IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 9, 2021, accepted November 3, 2021, date of publication November 8, 2021,
date of current version November 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3126582

A Predictive Multi-Tenant Database Migration and Replication in the Cloud Environment

AHMED E. ABDEL RAOUF, ALSHAIMAA ABO-ALIAN, AND NAGWA L. BADR
Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt
Corresponding author: Ahmed E. Abdel Raouf (ahmed_ezzat991@yahoo.com)

ABSTRACT With the rapid adoption of multi-tenant databases, the cloud provider consolidates multiple tenants' database on server machines, where the tenants share a common application and database instances. To ensure the quality of service (QoS) for the leased resources, both sides (i.e., the user and the provider) create a Service Level Agreement (SLA). Higher SLA violations result in high SLA contractual penalties and increase the possibility of losing the tenant. In addition, the unusual workload patterns of each tenant transactions require seamless adjustments due to the sudden burden changes and variability. As a result, to satisfy simultaneously availability and performance tenant requirements, it is necessary to perform reliable tenant migration and replication to distribute the workload to a flexible set of sites and avoid SLA violations. In this research, a cluster-based multi-tenant database management system (CB-MT DBMS) is proposed, which takes the migration and replication decisions in advance by monitoring and acting before the violation of the SLA occurs. In addition, a dynamic proactive multi-tenant database migration and replication MTDB-MR algorithm is proposed to reduce collisions and inconsistencies between migration and replication decisions for a group of violated tenants. Experimental results show that the proposed MTDB-MR algorithm is the ideal candidate for migration and replication of the violated multi-tenant databases, as it minimizes the total number of SLA violations, the number of multi-tenant clients SLA violations, client sites average response time and total execution time of each multi-tenant client site as compared to the previous algorithms.

INDEX TERMS Cloud computing, data migration, data replication, service level agreements SLA, TPC benchmarks.

I. INTRODUCTION

In a multi-tenant SaaS architecture, the tenants subscribe to a shared database to store their data. Thus, different performance can be achieved depending on the design of the data layer. Previously, various designs have been proposed for the multi-tenant data [5], [17]. The main difference between these designs is the level of separation of tenant data. Regardless of the layout of the data layer used in a multi-tenant structure, service providers face a difficult daily scenario. The tenants require strict guarantees for the performance and availability of the rental services, known as performance service level agreements (SLAs) [7], [11], [18]. Performance Service Level Agreement (SLA) is an agreement between tenants and service providers, which sets the minimum performance and availability requirements for a rental service. It also defines penalties if the SLA is violated. On the other hand, for service providers to make a profit, they must reduce operating costs and utilize most of their hardware and software resources [33], [42]. However, in multi-tenant environments [19], each tenant needs only a small portion of a single node's resources. So, the degree of multi-tenant synchronization for each node is very high, which makes ensuring SLA agreements difficult and a crucial issue.

Consequently, the cloud service provider should have an intelligent multi-tenant data storage system, which has efficient mechanisms for allocation, migration, and replication of the multi-tenant data. The main requirement for the multi-tenant data storage system is to provide a reliable Quality of Service (QoS) for the multi-tenants by satisfying SLA agreements. Moreover, it should reduce the cloud service provider operational costs, maximize the utilization of their hardware and software resources.

However, designing such multi-tenant cloud intelligent data storage system has several critical challenges. The first challenge is satisfying Quality of Service (QoS) for the multi

The associate editor coordinating the review of this manuscript and approving it for publication was Kaitai Liang.

tenants through meeting the SLAs. In other words, meeting SLA is an essential key for the cloud service provider to achieve a high overall quality of service by avoiding the SLA contractual penalties and the possibilities of losing tenant. The second challenge is the tenants have unusual patterns of workload, which require seamless adjustments due to the sudden burden changes and variability. Thus, it is necessary to perform reliable migration and replication of the multi-tenant data to distribute the workload to a flexible set of sites [41].

This paper contribution can be summarized by the following; firstly, a new clustered based multi-tenant database management system (CB-MT DBMS) is proposed to overcome the limitations of existing migration and replication techniques. The proposed system groups tenants' sites into disjoint clusters according to the least average communication cost between the tenant sites, to minimize the time required to execute the query transactions and perform the multi-tenant database migration and replication. Secondly, a new dynamic proactive provisioning multi-tenant database migration and replication (MTDB-MR) algorithm is proposed to handle the issues of irregular workload patterns of the tenant database transactions, which may lead to enormous SLA violations and their consequent contractual penalties.

The proposed MTDB-MR algorithm consist of four services.

- The first service is the Global Monitoring service which monitors all the tenant databases by collecting the response time and the type of each tenant's transaction.
- The second service is the Forecasting Service which utilizes a sophisticated system models for predicting the tenant's transaction response time. The forecasting service is built using three different prediction models: The Recursive Window Forecasting Autoregressive Integrated Moving Average (ARIMA) model, the Exponential Moving Average (EMA) model and the proposed Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells model to predict future window of 15 minutes of data.
- The third service is the Access Log Analysis service which takes the migration and replication decision in advance based on both the workload of the global monitoring service and the results of the forecasting service.
- The final service is the Tenant Weight Matrix service, which selects the optimal site to migrate or replicate the violated tenant database according to a set of proposed Migration and Replication Rules (MRR). The proposed MRR is designed to reduce collisions and inconsistencies between migration and replication decisions for a group of violated tenants. In other words, it selects the optimal sites for a group of violated tenants instead of choosing the optimal site for only one violated tenant as the previous work.

The rest of this paper is organized as follows; Section II overviews the related work. The proposed clustered based multi-tenant database management system (CB-MT DBMS) and its components are presented in Section III. Section IV presents the proposed multi-tenant database migration and replication (MTDB-MR) algorithm. The experimental evaluations are demonstrated in section V. Section VI discusses the performance evaluation and finally, conclusions and future research directions are given in Section VII.

II. RELATED WORK

The current literature can be categorized into two subsections. The first subsection discusses the multi-tenant data layer design and the advantages and disadvantages of each design. The second subsection discusses the current multi-tenant replication and migration techniques and their limitations.

A. MULTI-TENANT DATA LAYER DESIGN

In the multi-tenant SaaS architecture, three data layer designs were proposed and used in different application areas [5], [17]. The main difference between these designs is the level of separation of the tenants' data. The first design is called Independent Databases and Independent Database Instances (IDII) where each tenant has their own database system running on the server. However, this design violates the main idea of the multi-tenant SaaS architecture, where each tenant should share the same instance of software and hardware.

The second design is called Independent Tables and Shared Database Instances (ITSI), where each tenant has their own separate tables running on a shared database instance. This design uses a single shared database system, which decreases the maintenance costs compared to the IDII as it uses a single shared database system.

The final design is called Shared Tables and Shared Database Instances (STSI) where the tenants share the same database instance and tables. However, this design is more complex when it comes to customization. In addition, the security aspect of the tenants which share the same tables. If there are errors in the tenant application code, a tenant may access all other tenants' data.

B. MULTI-TENANT REPLICATION AND MIGRATION TECHNIQUES

Lately, data replication and migration techniques have gained much attention [14]. Whenever any change in tenant performance is discovered [3], multi-tenant migration and replication techniques are used to transfer the specified tenant under observation to another environment. These techniques are used to decrease the load imposed on the cloud host environment and to consolidate multiple tenants into the same host environment. Thus, exploiting multi-tenant migration and replication techniques aims to share the host resources with a reduced interference between tenants.

The authors of [20] provided an overview of the latest data replication technologies. The authors of [1], [6], [24], [28], and [29] have claimed that the migrations and replication strategies have various important issues to consider: The first issue is that the tenants have irregular workloads pattern,

which can affect the quality of services. The second issue is when to take the decision to migrate or replicate. The third issue is selecting which tenant to be the target of migration or replication. The fourth issue is choosing which operation (migration or replication) to be applied to the target tenant. The final issue is selecting the site where the tenant to be replicated or migrated. However, according to [26], controlling the number of replicas is also important. To handle the issue of tenant's irregular workloads patterns, dynamic provisioning techniques [22] are designed which take actions based on the observations of the workloads. There are two types of provisioning techniques: reactive and proactive [40]. Proactive techniques use prediction models to predict the future tenant access patterns then use the prediction results to take the migration or replication decisions to mitigate the crisis before happening. In contrast to this, reactive techniques detect and react to existing SLA violations by using a predefined threshold.

1) PROACTIVE TECHNIQUES

The authors of [16] and [22] proposed the PredRep approach, which analyzed the cloud database system workload. They stated that tenants have irregular workload patterns which affect quality of service guarantees, mainly due to the overlap between tenants. They used the SLA (response time) as the target objective for each tenant. However, the authors of [16] and [22] stated that, despite the satisfactory results, the proposed PredRep approach has several limitations: firstly, violated tenants with large size databases and irregular workloads patterns will not be suitable for replication due to the data backup overhead and the restore time. Secondly, the constant change in workload patterns can lead to inaccurate prediction results. Finally, large changes in workload patterns can cause interference to other tenants.

If the SLA of a tenant cannot be satisfied, it must migrate to a site where its quality can be guaranteed. As a result, they proposed an allocation strategy to reduce the provider penalties cost of SLA violations and improve performance. The aim of the allocation strategy is to decide whether the target tenant can be migrated to an existing VM or a new VM.

Given a tenant L in a site K to be migrated. For all tenants I in the selected site (J) to migrate the violated tenant L located in site (K), the authors defined MT_{IJ} as the mean execution time of the last M executed transactions. Where SLA_{IJ} is the performance service level agreement for tenant (I) in site (J) and the (N_J) is the number of tenants in site J. The allocation strategy sends the tenant to the site X that has a max MD_J' as shown in (1), if and only if the site X has a free disk space and $x \neq k$. However, if one or more tenants have violations in a site, then the MD' will not be calculated correctly.

$$MD'_J = \frac{\sum_{I=1}^{N_J} (MT_{IJ} - SLA_{IJ}) + (MT_{LK} - SLA_{LK})}{N_J + 1} \quad (1)$$

The work of [13], which is the extension of [30], showed that the AutoRegressive Integrated Moving Average (ARIMA) and the Exponential Moving Average (EMA) prediction models obtained similar accuracy for time series prediction. The authors of [25] introduced a data replication strategy that does not predict performance, but once the individual query arrives at the nodes, it estimates the response time of individual queries. Then it evaluates whether the response time can be satisfied or not. If response time cannot be satisfied, it creates a replica only if the creation is profitable. However, this approach can generate many replicas which result in a storage overhead.

RepliC is a cloud-based database replication strategy that supports quality of service, flexibility, and multi-tenancy [21]. The elasticity changes the system's capacity dependent on the current workload by adding and removing replicas. Extra resources are added if the monitored value does not match the set SLA. The author, however, did not specify the location of the newly added replica.

From a risk management perspective, the author of [36] approaches suggested the method of SLA violation detection and abatement. Following the establishment of an SLA, they offered a Risk Management-based Framework for SLA Violation Abatement (RMF-SLA), which includes SLA monitoring, violation prediction, and decision recommendation.

The authors of [37] offered FLAS (Forecasted Load Auto-Scaling), an auto-scaler for distributed services that combines the benefits of proactive and reactive techniques based on the scenario to determine the optimum scaling actions at any given time. The main novelties introduced by FLAS are firstly a predictive model of the high-level metrics trend that allows for the prediction of changes in the relevant SLA parameters (e.g. performance metrics such as response time or throughput) and secondly a reactive contingency system based on the estimation of high-level metrics from resource use metrics, reducing the required instrumentation.

The authors of [38] presented heuristic policies that use the recursive least squares method to forecast QoS parameters and compute the resources required in future intervals; however, the procedure of addressing SLA breaches when they are predicted by the system is not defined.

According to the evaluation results, the authors of [39] got an ideal prediction result by utilizing small intervals for prediction and the autoregressive integrated moving average (ARIMA) method.

2) REACTIVE TECHNIQUES

Some other authors have adopted reactive technology to detect and react after a Service Level Agreement (SLA) violation has occurred. The authors of [15] suggested an approach called SWAT which exchanged primary replica with one of its secondary replicas to balance the load. The proposed approach is very effective in terms of time and resources as it does not incur primary replica movement. However, the approach has two limitations; the first limitation is that the server can be overloaded until the approach takes a decision. The second limitation is the authors assumed that the secondary replica executes only the update queries relayed from

the primary replica and the read only queries has a zero load on secondary replica which may not be always valid in the real application.

The authors of [23] proposed two reactive solutions to balance the load in case any machine of the multi-tenant database system become overloaded. The first solution swaps the primary replica on the overloaded machines with one of its secondary replicas in any other machine. The second solution migrates the replicas one at a time from the most overloaded machine to the most underloaded machine until either the load of the machine gets balanced, or the algorithm migrates the maximum number of the database replicas. The algorithm moves the most active database first to mitigate the crisis with the minimum number of migrations. However, the two solutions have various limitations.

The first limitation is that the authors did not clarify the criteria based on which they choose the secondary replica to be swapped to mitigate the crisis. The second limitation is that the algorithm does not search for the source of the crisis as it always searches for any primary replica, or it selects the most active replica in the overloaded machine to swap or migrate it. The third limitation is that the algorithm does not consider the impact of the swapping/migration solution on the query access time. The fourth limitation is the reactive manner of the algorithm as it does not consider the distribution of the multi-tenant databases over the cloud machines in each region. More specifically, the algorithm runs every week in the dataset by moving the most active replica in the overloaded machine and then tests the effect of the migration on the next week.

Consequently, the machine can stay overloaded for one or more weeks until the algorithm selects the right replica to swap or migrate it, which can increase the number of SLA violations. Moreover, the authors of [22] stated that there is a limitation in the algorithm proposed in [23] for handling the workload changes. The limitation arises when multiple tenants are very active at the same time, then SLA violation may occur as only one database will be migrated at time.

The author of [12] presented a lightweight load balancing mechanism to solve the hotspot problems by exchanging the roles between tenants' primary replicas and secondary replicas. They mentioned that the existing load balancing work of [15] using replica exchange cannot be practically used. As their solution [15] assumed that each tenant has only one primary replica and one secondary replica used for fault tolerance. They assumed that the read-only queries that incur on the tenant secondary replicas has zero load and these queries just execute the update logs relayed from the tenant primary replicas. However, the authors of [12] mentioned that the load of the read only queries on the tenant secondary replicas should not be neglected in the practical use. Consequently, they [12] proposed a load balancing replica exchange strategy to select a suitable tenant secondary replica among multiple tenants' secondary replicas and exchange roles with the selected tenant primary replica as a result, it solves the first limitation of the previous work [23].

However, most of the aforementioned limitations of [23] still exist in the algorithm [12] as it does not search for the source of the crisis since it always searches for a primary replica to swap it with one of its secondary replicas. Also, the authors did not mention what happens if the algorithm cannot find a suitable secondary replica. Additionally, the algorithm does not consider the distribution of the multi-tenant databases over the cloud machines in each region while applying the migration processes. Finally, the algorithm does not consider the case of having more than one overloaded host and how to take the optimal solution for each host while reducing the interference between the migration and replication decisions in each host.

Based on the preceding discussion, it is obvious that although various techniques for cloud multi-tenant replication and migration approaches have been proposed in the literature, not all of them assist the service provider on the allocation, replication, and migration decisions necessary to mitigate the SLA violation.

Table. 1. compare different multi-tenant approaches on the five criteria required for mitigating SLA violations; namely the ability to predict possible SLA violations in proactive approaches (i.e., Proactive), the ability to detect and react after a SLA violation has occurred in reactive approaches (i.e., Reactive), the ability to migrate the violated tenant to mitigate the crisis (i.e., Migration), and the ability to replicate the violated tenant to mitigate the crisis (i.e., Replication), and finally the ability to best allocate all the violated tenant in case there are more than one violated tenant (i.e., Placement).

III. THE PROPOSED MULTI-TENANT DATABASE MIGRATION AND REPLICATION (MTDB-MR) ALGORITHM

The primary goal, as stated in the literature review, is to develop a multi-tenant migration and replication algorithm capable of selecting the best solution for several violation tenants based on their irregular workload patterns, instead of generating it for only one violated tenant, as in earlier studies [12], [15], [16], [22], [23], which could result in massive SLA violations and a contractual penalty. Additionally, the amount of overlap between migration and replication decisions in each host should be reduced, and SLA violations should be avoided. As a result, we propose a Multi-Tenant Database Migration and Replication (MTDB-MR) algorithm, which consists of Six services: Global Monitoring Service, Query Coordinator Service, Forecasting Service, Clustering Service, Access Log Analysis Service, and Tenant Weight Matrices Service.

The proposed MTDB-MR algorithm works proactively as follows: Firstly, it uses the forecasting service to detect the violated tenant's replica, which solve the issue of the irregular workloads pattern which may lead to enormous SLA violations and a consequent contractual penalty. Secondly, it uses the proposed access log analysis service to take one of two decisions; either replicate or migrate the violated tenant replica for a group of violated tenants, not for just only one tenant as the previous works [12], [15], [16], [22], [23].

TABLE 1. Comparison of reactive and proactive approaches and their limitations.

Paper	Proactive	Reactive	Migration	Replication	Placement	Limitations
[21]	Y	N	N	Y	N	• If the monitored value is not in accordance with the defined SLA, more resources are added. • The author did not determine the location of the new added replica. • The authors created a new VM with the slave replica.
[16]	Y	N	Y	N	N	The authors stated that the ARIMA prediction method generates some errors. They suggested studying the behavior of the signals using other prediction methods.
[22]	Y	N	N	Y	N	• Tenants with large databases and irregular workload patterns would not be suitable for replication due to data backup overhead and the time required to restore. • Constant change in workload patterns can lead to inaccurate forecasting results as well as cause interference with other tenants.
[25]	Y	N	N	Y	N	This approach may generate many replicas.
[30]	Y	N	N	N	N	• Flexibility is not considered. • No replication mechanisms are used for the addition of replicas.
[13]	Y	N	N	Y	N	
[36]	Y	N	N	N	N	
[38]	Y	N	N	N	N	The procedure of addressing SLA violations when they are predicted by the system is not defined.
[15]	N	Y	N	N- Swap	N	• The server can get overloaded before the approach makes a choice. • The authors believed that the secondary replica only performs update queries relayed from the primary replica, and that read-only queries have no load on the secondary replica, which may not necessarily be true in practice. • The existing load balancing work based on replica exchange could not be implemented in practice [12].
[23]	N	Y	Limited Migrations	N	Y	• The authors did not specify the criterion by which they chose the secondary replica to be exchanged to reduce the crisis. • The algorithm always looks for any main replica or chooses the most active replica in an overloaded machine to swap or move, instead of looking for the source of the problem. • The algorithm ignores the influence of the swapping/migration solution on the query access time. • The algorithm does not account the distribution of multi-tenant databases across cloud machines in each region. As a result, the system may remain overloaded for one or more weeks until the algorithm finds the appropriate replica to swap or migrate, thus result in increasing the number of SLA violations. • When numerous tenants are very active at the same time, SLA violations can occur since only one database is moved at a time [22].
[12]	N	Y	N	N- Swap	N	• The algorithm retains most of the limitations of [23], as it does not look for the source of the crisis because it constantly looks for a primary copy to swap with one of its secondary copies. • The authors did not indicate what occurs if the algorithm is unable to locate a suitable secondary replica. • When implementing migration operations, the algorithm did not consider the distribution of multi-tenant databases among cloud sites. • The approach ignores the scenario of multiple overloaded hosts and how to find the best answer for each one while minimizing interference between the migration and replication decisions in each host.

Finally, it determines the optimal site to migrate or replicate the violating tenants using the tenant weight matrices service and clustering service, which reduce collisions and inconsistencies between migration and replication decisions for a group of violated tenants.

A. GLOBAL MONITORING SERVICE

The purpose of the global monitoring service is mainly used to store the complete information for each tenant transaction. The collected information consists of the transaction response time and the transaction type. This collected information in addition to the location of each tenant and its replica are stored in a database called global catalog log database.

B. QUERY COORDINATOR SERVICE

The query coordinator service is used to redirect the tenant transaction to the closest suitable host to execute the transaction. As a result, the target tenant or its replicas must be placed on a host that is closest to the sites imposing the most amount of the queries. After executing the transaction, a complete transaction information will be stored on the global catalog log database which consists of the tenant's

transaction response time, the transaction type, client location and tenant location, which will be used later to take either replication or migration decisions.

C. FORECASTING SERVICE

As any tenant demands the stringent guarantees for the performance and availability of the rented services, which are known as performance service level agreements (SLAs), the forecasting service uses the information in the catalog log database to predict the behavior of the tenant transaction in advance.

However, in case the forecasting service detects an SLA violation, the proposed MTDB-MR algorithm should have sufficient time to take an action before the violation occurs and thus avoid SLA violation and its contractual penalties. We developed the proposed forecasting service previously [34] using three different prediction models: The Recursive Window Forecasting Autoregressive Integrated Moving Average (ARIMA) model, the Exponential Moving Average (EMA) model and the proposed Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells model to predict future window of 15 minutes of data.

The aim of building the proposed forecasting service using three different prediction models is to compare the accuracy of different prediction model using the same monitored data and in the same multi-tenant environment to select the optimal prediction model.

Our previous experiment results show that the forecasting service built using the proposed RNN prediction model is the ideal candidate for solving multi-tenant forecasting problem. To evaluate the accuracy of the proposed three prediction models, Mean Absolute Percentage Error (MAPE) and Root mean square error (RMSE) are used for computing the value of error during the training process. The value of RMSE and MAPE determine the difference between the predicted values and the actual values. As a result, lower values of RMSE and MAPE indicate higher accuracy. Our previous experiment results firstly proved that the proposed RNN prediction model reduces RMSE and MAPE compared to ARIMA and EMA models. Secondly, it proved that the proposed RNN prediction model is superior to their counterparts for detecting SLA violation values and windows using different SLA values.

D. CLUSTERING SERVICE

Clustering service is usually accomplished by determining the similarity between the items depending on their characteristics [31], [32]. Based on our literature review, there are two efficient ways to cluster the distributed database sites. The simplest way is clustering the distributed database sites using the region fields of the database [27], where the sites in the same region are assigned to the same cluster. However, this way of clustering will not work properly in case of all the sites belonging to the same region or each site belonging to a different region. The second way to cluster distributed database sites is to utilize the communication cost between database sites where sites are grouped into disjoint clusters according to the least average communication cost between network sites [1], [2], [8].

This clustering process depends highly on the communication cost range (CCR) value. If the communication cost between two sites is less than the CCR then the two sites are grouped into one cluster, which depends on how much time is allowed for the sites of the same cluster to transmit or receive their data.

The main advantages of this strategy are that it minimizes the time required for query processing and data allocation and it can be implemented in different environments even if the sites are enormous. However, it depends on the value of the CCR, which is considered as trivial drawback as it can be determined easily by the network admin. Consequently, we implement this strategy in multi-tenant environment to minimize the time required to execute the query transactions and multi-tenant database migration and replication.

E. ACCESS LOG ANALYSIS SERVICE

As the proposed MTDB-MR algorithm is used for online transaction processing (OLTP) purposes, there are two types of the tenants in the multi-tenant architecture: primary replica and secondary replica. Primary replicas allow read/insert/delete/update operations whereas secondary replicas allow only read operations. As a result, to ensure the SLA guarantees, the target tenant or its replicas must be placed on the closest host to the most amount of the queries.

Consequently, the proposed access log analysis service uses the forecasting service results to detect the violated tenants. After that, it uses the violated tenant monitored information which is collected by the monitored service and stored on the global catalog log database, to take one of two decisions (i.e., replicate the violated tenant or migrate the violated tenant). That decision is taken based on the number and the types of violated tenants' transactions as shown in a flowchart in Fig. 1.

F. TENANTS' WEIGHTS MATRICES SERVICE

To select the optimal sites for a group of violated tenants and not for just only one violated tenant as the previous works [12], [15], [16], [22], [23], the tenant site weight matrix (TSWM) is proposed, which is used as the basis for the assignments of the tenants or its replicas to the sites. The TSWM is a table constructed between the target tenants and the target sites. It is used to represent the weight of each tenant or its replica to be assigned to a specific site. In the TSWM, the highest weight, in each violated target tenant row, represents a convenient site to migrate or replicate the violated tenant to mitigate the crisis. The TSWM is constructed from the proposed migration and replication rules (MRR) which are a set of rules designed to reduce collisions and inconsistencies between migration and replication decisions for a group of violated tenants.

The proposed MRR rules are the tenant mix rule, the tenant replacement rule, the tenant communication cost rule, and the tenant violation rule. The TSWM (T_I, S_J) is determined

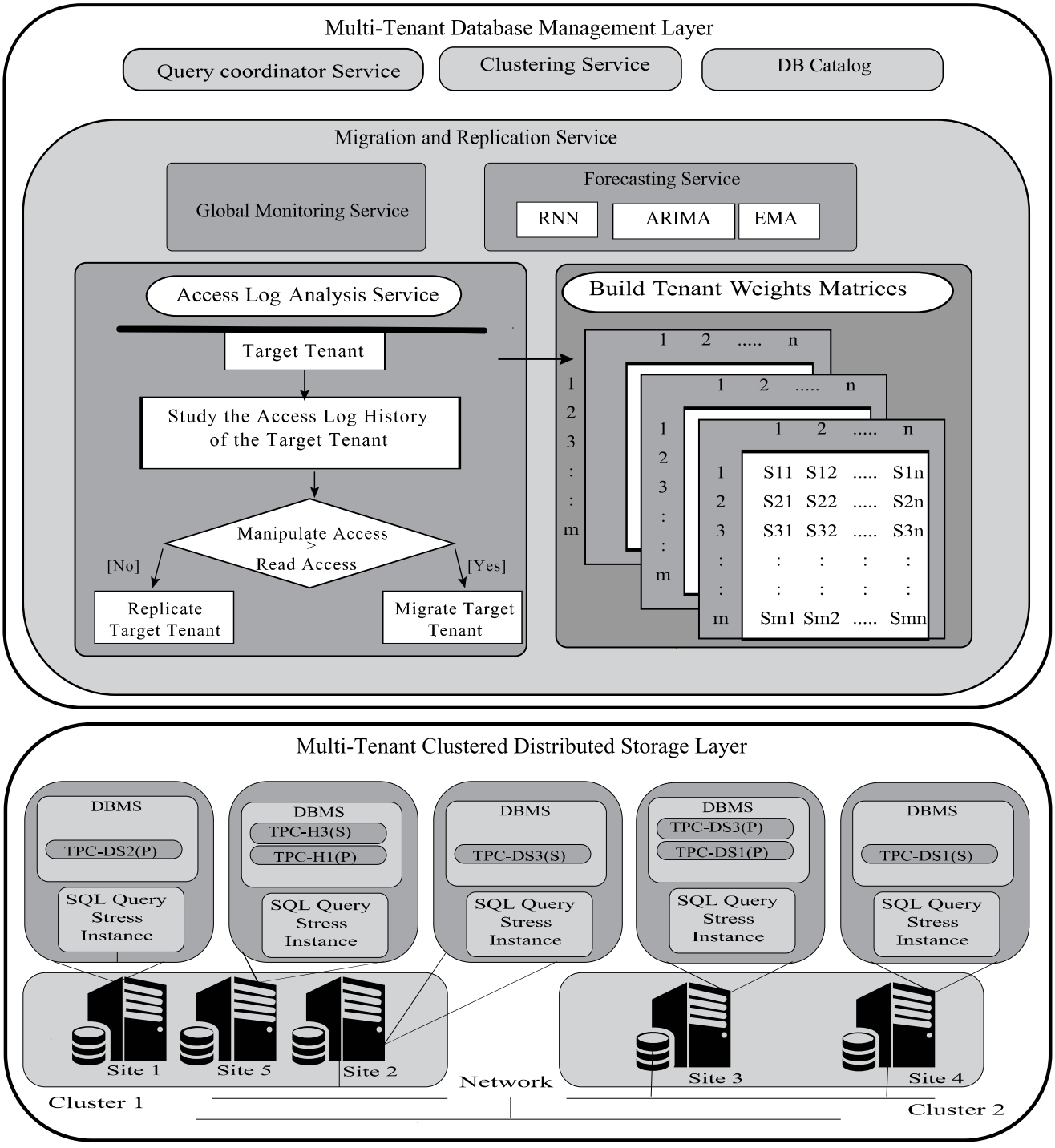

FIGURE 1. Clustered based multi-tenant database management system (CB-MT DBMS).

by (2). Where: the TMM (T_I, S_J) is the Tenant Mix Matrix between target tenant T_I and site S_J. The TSM (T_I, S_J) is the Tenant Swap Matrix between target tenant T_I and site S_J. The TECM (T_I, S_J) is the Tenant Execution Cost Matrix between target tenant T_I and site S_J. The TSVM (T_I, S_J) is the Tenant Site Violation Matrix between target tenant T_I and site S_J. Where W_I represents the impact degree of each rule on the assignment of the target tenants. The list of notations used are illustrated in Table. 2.

$$\begin{aligned}&\mathrm{TSWM}\,(T_I, S_J)\\&\quad = (W_1 * +\mathrm{TMM}\,(T_I, S_J)) + (W_2 * +\mathrm{TSM}\,(T_I, S_J))\\&\qquad + (W_3 * \mathrm{TECM}\,(T_I, S_J)) + (W_4 * \mathrm{TSVM}\,(T_I, S_J))\end{aligned} \tag{2}$$

TABLE 2. List of abbreviations.

TSWM	Tenant Site Weight Matrix
TMM (T_I, S_J)	Tenant Mix Matrix between target tenant T_I and site S_J
TSM (T_I, S_J)	Tenant Swap Matrix between target tenant T_I and site S_J
TECM (T_I, S_J)	Tenant Execution Cost Matrix between target tenant T_I and site S_J
TSVM (T_I, S_J)	Tenant Site Violation Matrix between target tenant T_I and site S_J
W_I	Importance of each rule on the assignment of the target tenants
RQ(S_J,T_I)	Number of read operations of site S_J to tenant T_I
MQ(S_J,T_I)	Number of read operations of site S_J to tenant T_I
AVG(RQ(LW))	Average of read operations in the last widow
AVG(MQ(LW))	Average of read/insert/delete/update operations in the last widow
CC(S_I, S_J)	Communication cost between site S_I and site S_J
TS	Violated tenant site
NP(TS)	Number of primary replicas in tenant site TS
NS(S_J)	Number of secondary replicas in site S_J
SR(T_I)	Secondary replica of tenant T_I
T(T_I)	Type of replica T_I (primary P, secondary S)

1) TENANT COMMUNICATION COST RULE (TCCR)

To reduce the time required to carry out tenant transactions, the tenant or its replica should be moved closer to the client sites from where the largest number of transactions are made.

Thus, the main purpose of the proposed tenant communications cost rule is to predict the total cost of executing all tenant transactions when the violating tenant is migrated or replicated to a particular site. Thus, the ideal location for migrating the tenant or its copy is that results in the largest reduction in transactions execution time for all client sites and not just one site as it is currently in the previous works. To implement the TCCR rule, the proposed MTDB-MR algorithm utilizes both the tenant access history (that is, collected using the Global Monitoring Service and stored in the catalog log database) and the results of the access log analysis service. The proposed MTDB-MR algorithm predicts the total cost of access to execute queries of all client sites during the last window before the migration or replication decision is implemented.

The proposed Tenant Communication Cost Rule (TCCR) is used to generate the proposed Tenant Execution Cost Matrix (TECM) which is a table constructed by placing the target tenants as rows and the target sites as columns. The value of TECM (T_I, S_J) represents the predicted reduction weight on the tenant total execution cost if the target tenant T_I is migrated or replicated to site S_J. The proposed TECM is generated using one of two matrices, which are Tenant Migration Execution Cost (TMEC) or Tenant Replication Execution Cost (TREC), depending on the type of violating tenant (primary tenant, secondary tenant).

In case the tenant type is primary replica, the Tenant Migration Execution Cost (TMEC) is a table constructed by placing the target tenants as rows and the target sites as columns. It represents the predicted total execution cost if the target tenant T_I is migrated to site S_J.

Similarly, in case the tenant type is secondary replica, the Tenant Replication Execution Cost (TREC) is a table constructed by placing the target tenants as rows and the target sites as columns. It represents the predicted total execution cost if the target tenant T_I is replicated to site S_J.

Assuming we have a violating tenant (I) located in site (TS). If the violating tenant type is Primary Replica, then the proposed algorithm iterates over the sites of the multi-tenant system to predict the total execution cost for all sites transactions when the violated tenant (I) is migrated to the new site (J), using the communication cost between the sites and the average of read and manipulate transactions in the last window, using TMEC (T_I, S_J) determined by (3).

After that, for each violating tenant in each site, the proposed algorithm calculates the TECM as the predicted weight of increase or decrease in the total execution cost when the violated tenant (I) is migrated to the new site (J) compared to the original total execution cost of the violated tenant (I) in the original violating site (TS).

Similarly, if the violating tenant type is Secondary Replica, the predicted the total execution cost for all sites transactions when the violated tenant (I) is replicated to the new site (J), are calculated using TREC (T_I, S_J) determined by (4), using the communication cost between the sites and the average of read transactions in the last window.

The pseudocode for generating the proposed TECM matrix is shown in Fig. 2. The complexity of generating the TECM

Algorithm: Generate Tenant Execution Cost Matrix (TECM)

```
Generate Tenant Execution Cost Matrix (TECM)
Input:
    a) Type T for each target tenant, where T(TI) is the type of tenant TI
       (primary P, secondary S)
    b) TS, the target tenant site.
    c) RQ (SJ, TI), the number of read queries of site SJ to tenant TI.
    d) MQ (SJ, TI), the number of read/insert/delete/update operations of
       site SJ to tenant TI.
    e) AVG(RQ), the average of read operations in the last window.
    f) AVG(MQ), the average of read/insert/delete/update operations in the
       last window.
Output: Tenant Execution Cost Matrix (TECM)
for (I in 1:N) Where N is the Number of Target Tenants do
    for (J in 1:M) Where M is the Number of Sites do
        if T(TI) = P   # tenant type is Primary Replica
            for (F in 1:K) Where K is the Number of Sites do
                TMEC (TI, SJ) +=
                    RQ (SF, TI) *(AVG(RQ) + CC(SJ,SF)) + MQ (SF, TI)
                    *(AVG(MQ)+ CC(SJ,SF))
            endfor
            TECM(TI,SJ) =1-(TMEC(TI,SJ) ÷ TMEC(TI, TS))
        else if T(TI) = S   # tenant type is Secondary Replica
            for (F in 1:K) Where K is the Number of Sites do
                TREC (TI, SJ) += RQ (SF, TI) *(AVG(RQ) + CC(SJ,SF))
            endfor
            TECM(TI,SJ) =1- (TREC(TI,SJ) ÷ TREC(TI, TS))
        endif
    end for
end for
return TECM
```

FIGURE 2. Pseudocode for generating tenant execution cost matrix (TECM).

is O (T * M^2) where T is the number of violated tenants and M is the number of multi-tenant system sites.

$$\begin{aligned} TMEC(T_I, S_J) = & \sum_{F}^{\text{Number of Sites}} RQ(S_F, T_I) \\ & * (AVG(RQ(LW)) + CC(S_J, S_F)) \\ & + MQ(S_F, T_I) * (AVG(MQ(LW)) \\ & + CC(S_J, S_F)) \end{aligned} \quad (3)$$

$$\begin{aligned} TREC(T_I, S_J) = & \sum_{F}^{\text{Number of Sites}} RQ(S_F, T_I) \\ & * (AVG(RQ(LW)) + CC(S_J, S_F)) \end{aligned} \quad (4)$$

2) TENANT VIOLATION RULE (TVR)

As any multi-tenant cloud provider faces the issue of the irregular workloads pattern which may lead to enormous SLA violations and a consequent contractual penalty, the service provider must have effective strategies to allocate, migrate, and replicate the tenant's databases to reduce their operating costs and maximize the use of their hardware and software resources while minimizing the SLA violations and their consequent contractual penalties.

Consequently, the proposed Tenant Violation Rule (TVR) is used to measure the impact of migration or replication decisions of the violated tenants' replicas on the number of SLA violations for all violated tenant client's sites.

The proposed Tenant Violation Rule (TVR) is used to generate the proposed Tenant Site Violation Matrix (TSVM) which is a table constructed by placing the target tenants as rows and the target sites as columns.

The value of TSVM (I, J) represents the weight of decreasing or increasing the number of violation when tenant (I) is migrated or replicated to site (J) as compared to the weight of violation at the violated tenant site.

The TSVM is generated using the Tenant Transaction Violation Matrix (TTVM) which is a table generated for each target tenant and constructed by placing the tenant's client sites as rows and the multi-tenant database system sites as columns. The value of TTVM (I, J) represents the predicted number of violations of tenant clients site transactions (I) when the tenant is migrated or replicated to site (J).

Assuming we have a violating tenant (T) located in site (TS). The value of TTVM (I, J) is calculated as following: the proposed algorithm iterates over the multi-tenant sites (J) to predict the number of SLA violations of each client site (I) transactions in case the predicted response time is greater than the corresponding value stated in the SLA, when the violated tenant (T) is allocated to site (J), as shown in step (6 to 12) in the pseudocode for generating the TTVM matrix shown in Fig.3. The predicted transaction (F) response time of client site (I) is calculated using (5).

$$\begin{aligned} \text{Predicted Transaction (F)} = & \text{ Transaction (F)} - CC(S_I, S_{TS}) \\ & + CC(S_I, S_J) \end{aligned} \quad (5)$$

Algorithm: Generate Tenant Transaction Violation Matrix (TTVM)

```
1   Generate Tenant Transaction Violation Matrix TTVM (Tenant T)
    Input:
      a) TS is the target tenant site.
      b) CC is Communication cost matrix.
      c) List of each client site transactions of tenant (T).
    Output: Tenant Transaction Violation Matrix TTVM (Tenant T)
2   for (I in 1:N) Where N is the Number of Tenant (T) Client sites do
3     for (J in 1:M) Where M is the Number of Sites do
4       if Site(J) ≠ TS
5         TTVM (I, J)= 0
6         for (F in 1:K) Where K is the Number of transactions of client
7           site (I) do
8           Transaction (F) = Transaction (F) - CC(SI, STS) + CC(SI,SJ)
9           if (Transaction (F) > SLA(Tenant T))
10            TTVM (I, J)= TTVM (I, J) + 1
11          endif
12        endfor
13      endif
14    end for
15  end for
16  return TTVM (Tenant T)
```

FIGURE 3. Pseudocode for generating tenant transaction violation matrix TTVM (Tenant T).

After that, for each violating tenant in each site, the proposed algorithm calculates the TSVM (Violated tenant (I), Site (J)) as the predicted weight of increase or decrease in the total number of SLA violations when the violated tenant (I) is migrated or replicated to the new site (J) compared to the original total number of SLA violations of the violated tenant (I) in the original violating site (TS) as shown in step (2 to 9) in the pseudocode for generating the TSVM matrix shown in Fig. 4.

Algorithm: Generate Tenant Site Violation Matrix (TSVM)

```
1   Generate Tenant Site Violation Matrix TSVM
    Input:
      a) TS(T) for each the target tenant
      b) Tenant Transaction Violation Matrices TTVM(T) of each target
         tenant
    Output: Tenant Site Violation Matrix TSVM
2   for (I in 1:N) Where N is the Number of Target Tenants (T) do
3     Transaction Tenant Violation TTV(TS(I)) = Sum of TTVM(, TS) of
4       tenant (I)
5     for (J in 1:M) Where M is the Number of Sites do
6       TTV(J) = Sum of TTVM(, J) of tenant (I)
7       TSVM(I, J) = 1- (TTV(J) ÷ TTV(TS(I)))
8     end for
9   end for
10  return TSVM
```

FIGURE 4. Pseudocode for generating tenant site violation matrix TSVM.

The pseudocode for generating the TTVM matrix is shown in Fig. 3 whereas the pseudocode for generating the TSVM matrix is shown in Fig. 4. The complexity of generating the TTVM is O (T * S * M * K) where T is the number of violated tenants, S the number of the client sites, M is the number of multi-tenant system sites and K is the number of transactions of clients. In addition, the complexity of generating the TSVM is O (T * M) where T is the number of violated tenants and M is the number of multi-tenant system sites.

3) TENANT MIX RULE

As the functionality of the tenant primary and secondary replicas are different in the multi-tenant database environ-

ments, the cloud service providers prefer to host a mix of tenant's primary and secondary replicas on each server to balance the load across all servers [4].

Consequently, the proposed tenant mix rule is used to ensure that each host has a mix of tenants with primary and secondary replicas to balance the load across all servers using the proposed Tenant Mix Matrix (TMM). TMM is a table constructed by placing the target tenants as rows and the target sites as columns. The value of the TMM (T_I, S_J) is determined by (6). The value of TMM (T_I, S_J) is equal to 1 if the target tenant T_I can be allocated to the site S_J. To balance the load across all servers, the value of the TMM can be set to 1 in two cases; the first case is when the violated tenant type is a secondary replica and the number of primary replicas in the selected site is greater than 0. The second case is when the violated tenant type is a primary replica and the number of secondary replicas in the selected site is greater than 0.

The complexity of the generating TMM is O (T * M) where T is the number of violated tenants and M is the number of multi-tenant system sites.

$$\text{TMM}(T_I, S_J) = \begin{cases} 1 & \text{If } (T(T_I) = S \text{ and } NP(S_J) > 0) \\ & \textit{OR If } (T(T_I) = P \text{ and } NS(S_J) > 0) \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

4) TENANT SWAP RULE

The authors of [12] and [23] stated that a performance crisis can be mitigated by exchanging the roles between tenants' primary replicas and secondary replicas. Whenever the performance crisis is detected on a specific site, their scheduling algorithm selects a subset of the tenants in the affected site to swap them with their secondary replicas.

However, this algorithm has many limitations. The first limitation is that a server that has a failure will stay overloaded until the algorithm can select the right tenant and perform the swap. The second limitation is the algorithm always searches for the primary replica to swap it with one of its secondary replicas. Moreover, the crisis will not be mitigated if the algorithm does not find a suitable secondary replica. Finally, it also violates the proposed tenant mix rule, which states that the host must have a mix of tenant's primary and secondary replicas with the goal of balancing the load across all servers.

To solve these limitations as well as reducing the number of migrations, the proposed MTDB-MR algorithm imposes the Tenant Swap Rule to generate the proposed Tenant Swap Matrix (TSM), which is a table constructed by placing the target tenants as rows and the target sites as columns. The value of the TSM (T_I, S_J) is determined by (7). The value of TSM (T_I, S_J) is equal to 1 if the site S_J has a secondary replica of the target tenant T_I and both the tenant site and the selected site must not violate the tenant mix rule. The complexity of generating the TSM is O (T * M) where T is the number of violated tenants and M is the number of multi-tenant system sites.

$$\text{TSM}(T_I, S_J) = \begin{cases} 1 & \text{If } S_J \text{ has SR } (T_I) \text{ and NS } (S_J) > 1 \\ & \text{and NP(TS)} > 1 \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

IV. EVALUATION

To evaluate the proposed MTDB-MR algorithm in a multi-tenant environment, it is important to use an appropriate standard criterion. But according to [21], there is no standard benchmark for assessing multi-tenant environments. Therefore, to simulate a multi-tenant environment, we provide a full multi-tenant environment with different databases in our evaluation.

These databases were provided by the OLTPBenchmark framework [35]. This framework provides different benchmarks such as TPC-DS and TPC-H. The OLTPBenchmark has over a decade's worth of experience in providing industry standard workloads, which is designed to produce the variable mixture, the variable rate load against any relational database.

The TPC-H Benchmark is a decision support benchmark that consists of a set of business queries and synchronized data modifications.

The queries and data in the TPC-H Benchmark database have been chosen to have remarkable importance at the industry level. The TPC-DS is a standard benchmark for measuring the performance of decision support systems solutions including, but not limited to, Big Data systems.

The evaluation section is divided into six subsections. The creation of an assessment environment for the multi-tenant database workload utilizing two distinct TPC benchmarks, as detailed in the first subsection.

In subsections 2, 3, 4, and 5, we use the proposed MTDB-MR algorithm in conjunction with the most recent migration models (i.e., Swap algorithm [12], Step algorithm [23], Allocation strategy of [16] and [22], and the basic strategy) to mitigate the crisis of the two violated tenants (TPC-DS1 and TPC-H2).

The final subsection compares the performance of the proposed MTDB-MR method to the performance of the following state-of-the-art migration models: swap algorithm [12], step algorithm [23], allocation strategy [16], [22], and basic strategy for mitigating the crisis of the two violated tenants (TPC-DS1, TPC-H2).

A. BUILDING THE SIMULATED MULTI-TENANT DATABASE ENVIRONMENT

The performance of the proposed MTDB-MR algorithm is studied in a simulated environment. The simulated environment consists of 8 Fujitsu esprimo-P556 sites clustered into 3 clusters as shown in Fig. 5. Each site has a Core I7– 3.40 GHz processor and 8 GB DDR3 MEMORY using the SQL Server as DBMS.

The clustering technique presented in [8] is used to cluster the sites into three clusters as shown in Fig. 5, based on the

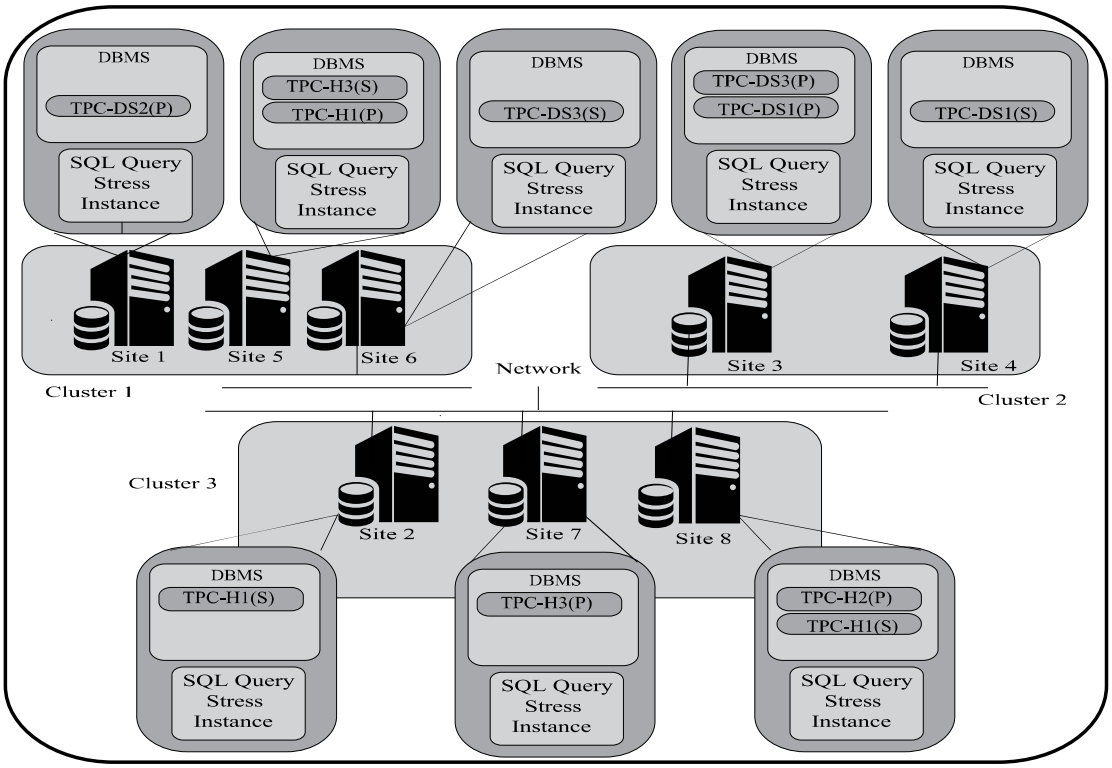

FIGURE 5. The simulated environment.

communication cost assumptions between sites in milliseconds, as shown in Table. 3. The communication costs between sites are randomly assumed so that the clustering algorithm can be applied to group the sites into three clusters based on the communication cost between the sites.

As illustrated in Fig. 5, each site has one DBMS with different tenant databases of different types (primary and secondary). In addition, each site runs a SQLQueryStress Performance Testing Tool [9], which simulates the multi-tenant client's connections.

We use 6 instances of TPC benchmarks were used: TPC-DS and TPC-H with its secondary replicas, which are distributed randomly to the sites of the multi-tenant system to study the ability of the proposed algorithm to take an optimal decision for a group of violated tenants and not for just only one tenant as the previous migration algorithms. The distribution of the tenants is shown in Fig. 5. As mentioned before and based on the merits and demerits of different multi-tenant data layer designs, the data layer is built using the Independent Tables and Shared Database Instances (ITSI) data layer design, where each tenant has their own separate tables running on a shared database instance which consequently decreases the maintenance cost when compared to other designs.

TABLE 3. Communication cost between sites in milliseconds.

Site #	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7	Site 8
Site 1	0	69	57	70	10	12	75	77
Site 2	69	0	85	69	80	75	8	14
Site 3	57	85	0	12	53	63	87	82
Site 4	70	69	12	0	75	72	72	80
Site 5	10	80	53	75	0	10	78	85
Site 6	12	75	63	72	10	0	70	79
Site 7	75	8	87	72	78	70	0	12
Site 8	77	14	82	80	85	79	12	0

SQLQuersyStress is a tool designed to test query loads. It allows to specify the number of virtual users (i.e., up to 200 virtual users) and the number of iterations, which specifies the number of times a test query must be executed simultaneously by each virtual user to simulate the workload. However, in our proposed MTDB-MR algorithm, the access history of each violated tenant should be studied to take an appropriate decision for the violated tenants. As a result, the SQLQueryStress tool is edited to record the query type and the response time for each executed iteration.

To simulate the multi-tenant database workload in each tenant site, a procedure for each TPC benchmark is built which contains a list of benchmark samples' queries identified by QueryID [10]. However, if the TPC benchmark procedure was tested in a loop with the same QueryID, each run after the first would be faster, due to the data caching, which could affect the accuracy of the test. To fix this problem, SQLQueryStress needs to send a different QueryID every time to each virtual user to solve the data caching issue.

To verify the quality of the proposed MTDB-MR algorithm, a violation in two different tenants (TPC-DS1 and TPC-H2) located in different sites in different clusters were simulated. To make a violation, the rate of queries and the number of virtual users that access the tenant simultaneously were increased. The distribution of the queries, number of virtual users, and the type of the queries submitted by each multi-tenant client site to each target tenant (TPC-DS1, TPC-H2) is shown in Table. 4.

TABLE 4. The distribution of the queries submitted to each target tenant.

Sites	TPC-DS1			TPC-H2		
	Queries	Virtual Users	Query Type	Queries	Virtual Users	Query Type
Site 1	30000	1	Read/ Insert/ Delete/Update	7000	2	Read
	5000	20	Read/ Insert/ Delete/Update			
Site 2	20000	10	Read	7000	2	Read
Site 3	--	--	--	7000	2	Read
Site 4	30000	1	Read/ Insert/ Delete/Update	7000	2	Read
Site 5	30000	1	Read/ Insert/ Delete/Update	7000	2	Read
	5000	20	Read/ Insert/ Delete/Update			
Site 6	30000	1	Read/ Insert/ Delete/Update	7000	2	Read
	5000	20	Read/ Insert/ Delete/Update			
Site 7	20000	10	Read	7000	2	Read
Site 8	20000	10	Read	7000	2	Read

B. APPLY SWAP ALGORITHM

In order to mitigate the crisis for the first violated tenant (TPC-DS1)(P), the Swap algorithm [12] solves the crisis by searching for a primary replica in the violated site and swapping it with one of its secondary replicas in different site. According to the tenant distribution shown in Fig. 5, the violated site 3 contains only one primary replica. As a result, the swap algorithm swaps it with its secondary replica located in site 4.

Regarding the second violated tenant (TPC-H2)(P), swap algorithm [12] searches for a primary replica in the violated site. However, the violated site has only one primary replica TPC-H2 which is the source of violation. As a result, the swap algorithm [12] is not able to solve this case as it does not consider the case when a tenant does not have any secondary replicas in any other sites.

C. APPLY STEP ALGORITHM

In order to mitigate the crisis for the first violated tenant (TPC-DS1)(P), Step algorithm [23] proposed two reactive solutions to mitigate the crisis. The first solution leads to the same results of the swap algorithm [12], which swap the primary replica on the overloaded machines with one of its secondary replicas in any other machine. The second solution moves the most active database from the overloaded site to the least active site to mitigate the crisis with minimum number of migrations.

According to the tenant distribution shown in Fig. 5, the least active site is site 4. However, site 4 contains a secondary replica of the target tenant TPC-DS1. As a result, the second solution of the step algorithm [23] migrates the tenant to site 2 which is the second least active site.

Regarding the second violated tenant (TPC-H2)(P), the Step algorithm [23] sends the violated TPC-H2 tenant to site 4 which is the least active site.

D. APPLY THE ALLOCATION STRATEGY

In order to mitigate the crisis for the first violated tenant (TPC-DS1)(P), allocation strategy of [16] and [22] uses the SLA (response time) as the target objective for each tenant.

Therefore, the tenant TPC-DS1 will be migrated to site 7 which has the max average free time. Regarding the second violated tenant (TPC-H2)(P), The allocation strategy of [16] and [22] migrates the tenant TPC-H2 to site 1 which has the second maximum average free time. Meanwhile, the first violated tenant TPC-DS1 is sent to the first average free time site (i.e., site 7).

E. APPLY THE PROPOSED MTDB-MR ALGORITHM

The proposed MTDB-MR algorithm is designed to reduce collisions and inconsistencies between migration and replication decisions for a group of violated tenants. In other words, it makes optimal decisions for a group of violated tenants instead of choosing the optimal solution for only one violated tenant.

Based on the tenant's access history which is stored in the global catalog log database, the access log analysis service analyzes the access history of each target tenants (TPC-DS1 and TPC-H2) transactions to take one of two decisions (i.e., replicate the target tenant or migrate the target tenant). Based on the results of the access log analysis service, tenant TPC-DS1 should be migrated to another site, while the tenant TPC-H2 should be replicated to another site to mitigate the crisis. To select the optimal sites to replicate and migrate the violated tenants, the proposed MTDB-MR algorithm construct the TSWM from rule-based weights matrices: tenant mix matrix TMM, tenant swap matrix TSM, tenant execution cost matrix TECM and tenant site violation matrix TSVM.

The generated TSWM is shown in Table. 5 which is used as a basis for the assignment of the tenant to the sites. Now, the main objective is to optimally assign the tenants to the site while reducing the interference between the migration and replication decisions for each target tenant. Consequently, from the TSWM, the highest weight in each violated target tenant row, represents a convenient site to migrate or replicate the violated tenant to mitigate the crisis. As a result, TPC-DS1 will be migrated to site 6 and TPC-H2 will be replicated to site 3.

F. EVALUATE THE ACCURACY OF THE PROPOSED MTDB-MR AND ALL PREVIOUS ALGORITHMS

In this part, the proposed MTDB-MR for multi-tenant database migration and replication will be compared to

TABLE 5. The generated tenant site weight matrix TSWM.

	S1	S2	S3	S4	S5	S6	S7	S8
TPC-DS1	0.909793	0.024744	0	1.370798	1.863522	1.985906	-0.9827	-0.30005
TPC-H2	1.139763	0.040457	1.204637	0.170559	1.132015	0.117374	1.029391	1

prior works to demonstrate its superiority in dealing with multi-tenant migration and replication difficulties while reducing SLA violations. Clients may be lost if SLAs are violated at a higher rate. As a consequence, we evaluate the influence of the proposed MTDB-MR method, as well as all previous techniques, on the overall number of SLA violations.

As a result, in order to show the superiority of the proposed MTDB-MR algorithm, we compare its performance to that of prior algorithms (Swap algorithm [12], Step algorithm [23], Allocation strategy of [16] and [22], and basic strategy) on the following performance factors:

- The overall number of SLA violations.
- The number of SLA violations by multi-tenant clients.
- The average response time of each client site queries after applying the proposed MTDB-MR algorithm and the previous algorithms.
- The total execution time of each multi-tenant client site

The tenant evaluation findings for the first violated tenant, TPC-DS1, are shown in Fig. 6. To begin, Fig. 6 (A) shows that the proposed MTDB-MR reduces the number of SLA violations by 78.5 % to the basic strategy with no migration and replication, whereas the Swap algorithm [12], Step algorithm [23], and Allocation strategy of [16] and [22] increase the number of SLA violations because they do not select the source of the problem and do not determine the right site to migrate the violated tenant replica. Second, as shown in Fig. 6 (B), the proposed MTDB-MR decreases the overall number of violations at each client site when compared to the prior Swap algorithm [12], Step algorithm [23], and Allocation strategy of [16] and [22], and basic strategy.

Similarly, Fig. 6 (C) illustrates the average response time of each client site inquiry after employing the proposed MTDB-MR algorithm, as well as the prior Swap algorithm, Step algorithm, Allocation strategy, and basic strategy. It demonstrates that the proposed MTDB-MR decreases the average response time for multiple clients by an average 27.5 % to the basic strategy without migration and replication.

Fig. 6 (B) and Fig. 6 (C) demonstrate that the proposed MTDB-MR chooses the optimal site to move the TPC-DS1 violating tenant, resulting in significantly fewer SLA violations and average response time for multiple multi-tenant client sites.

Finally, Fig. 6 (D) illustrates that when compared to all prior algorithms, including the basic strategy, the proposed MTDB-MR algorithm results in a significant decrease in overall execution time for each client site completed transaction. It demonstrates that the proposed MTDB-MR reduces overall execution time for all transactions by 43.69 % to the basic strategy with no migration or replication. The assessment findings presented in Fig. 6 (B), Fig. 6 (C), and Fig. 6 (D) demonstrate that the basic strategy and swap algorithm appear to be superior to our MTDB-MR method in just one location (site 4).

The following are the reasons behind this: In the basic approach, TPC-DS1 tenant and transactions are on the same site (site 4), therefore all TPC-DS1 tenant transactions are local to that site.

Furthermore, in the swap method, the tenant is switched with their secondary replica located at site 3 in the same cluster, and all TPC-DS1 tenant transactions in site 4 are deemed local to the new site 3, which is also located in the same cluster.

The tenant evaluation findings for the second violated tenant, TPC-H2, are shown in Fig. 7. To begin, Fig. 7 (A) demonstrates that the proposed MTDB-MR reduces the number of SLA violations by 99.5 % to the basic strategy.

The swap algorithm [12], on the other hand, is unable to handle this issue since it does not take into account the scenario where a tenant has no secondary replicas in any other sites. As a result, both the swap algorithm and the basic strategy have the same number of violations.

When compared to the basic strategy, the allocation and step algorithms either have a little increase or decrease in the number of SLA violations since they do not determine the source of the problem and do not choose the optimal site to migrate the tenant to.

To summarize, current experiment findings demonstrate that the proposed MTDB-MR algorithm considerably decreases the overall number of violations when compared to all prior methods, including the basic strategy. According to the studies, the proposed MTDB-MR determines the optimal location to replicate the violating tenant, resulting in fewer SLA violations.

Second, considering Fig. 7 (B) which demonstrates that the proposed MTDB-MR significantly reduces the average response time for multiple multi-tenant client sites. It demonstrates that the proposed MTDB-MR reduces average response time for many clients by 45.875 % compared to the basic strategy without migration and replication. Finally, Fig. 7 (C) illustrates that the proposed MTDB-MR reduces total execution time for all transactions by 45.8 % to the basic strategy with no migration and replication.

V. DISCUSSION ON PERFORMANCE EVALUATION

Recently, the techniques of data allocation, replication and migration have received much attention [14]. Whenever a change in a tenant's performance is detected [3], allocation, replication and migration techniques are used to move

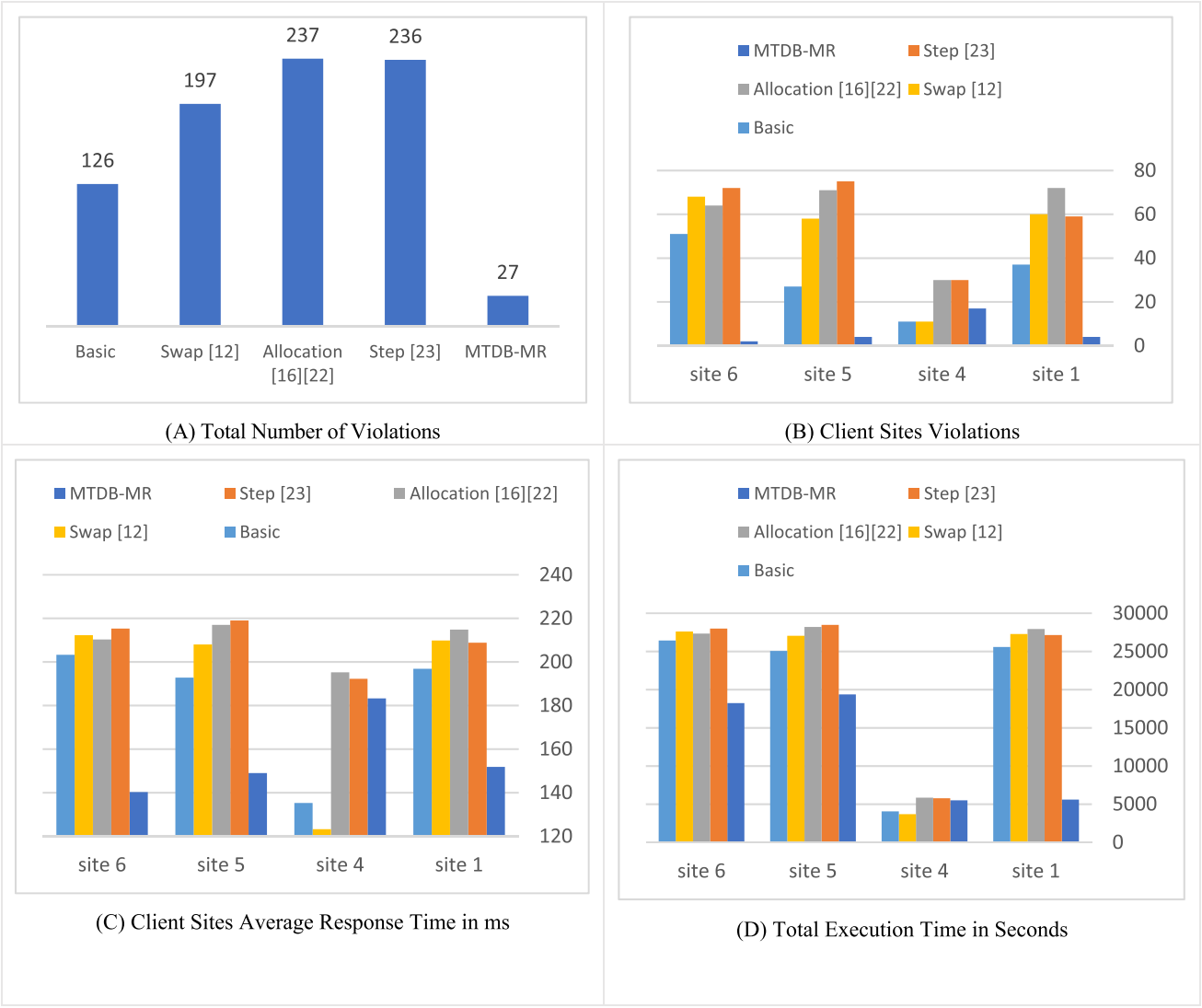

FIGURE 6. TPC-DS1 violated tenant evaluation result.

the specific tenant under violations to another environment. These techniques are used to reduce the load on a cloud host environment and to integrate multiple tenants into the same host environment.

However, designing such effective strategies for allocating, migrating, and replicating tenant databases has several critical issues. Firstly, different performance and availability requirements (SLAs) of all tenants should be considered. In other words, meeting a Service Level Agreement (SLA) is a vital key to achieving a high overall quality of service. High SLA violation rates indicate the potential for losing a tenant. Therefore, the number of SLA violations should be reduced to reduce costs from a cloud provider's perspective. As a result, the first task is how to ensure an SLA for a group of tenants according to the current workload while maximizing the utilization of hardware and software resources with small SLA violations. Secondly, tenants have irregular workload patterns, which require constant adjustments due to unexpected changes in workload and diversity. As a result, the second task is how to migrate and replicate the tenant databases on demand to distribute the irregular workload over a flexible set of devices. Thirdly, in case there are more than one violated tenant, an optimal allocation of the migrated and replicated tenant database could be very complex and requires good knowledge and experiences to reach. As a result, the third task is how to allocate the replicated and migrated tenant databases.

A new dynamic proactive multi-tenant database migration and replication (MTDB-MR) algorithm is proposed to handle the issues of irregular workloads and to meet the multi-tenant quality of service by avoiding service level agreement violations. To evaluate the performance of the proposed MTDB-MR algorithm in a multi-tenant environment, it is important to use an appropriate standard criterion.

FIGURE 7. TPC-H2 violated tenant evaluation results.

But according to [21], there is no standard benchmark for assessing multi-tenant environments. Consequently, multiple instances of TPC benchmarks: TPC-DS and TPC-H are used, to simulate a multi-tenant environment. But each tenant has irregular workload patterns which gracefully require adjustments due to the unexpected workload changes and variability.

Experiment results show that the proposed MTDB-MR algorithm is the ideal candidate for migration and replication of the violated multi-tenant databases, as it detects the source of the problem and selects the optimal operation to apply based on the results of the access log analysis services. It also selects the optimal site to migrate or replicate the violated tenant according to a set of rules which reduces the expected number of violations and reduces the expected execution time for all sites and not for just only one site as the previous works. Moreover, the proposed algorithm considers the tenant mix and swap rules to distribute the load of the queries and minimize the number of migrations.

VI. CONCLUSION AND FUTURE WORK

A multi-tenant database has a predominant role in hosting multiple tenants within a single DBMS with the active sharing of resources enabled. Providing these performance goals is a challenge for cloud service providers as they must balance the performance they can provide to their tenants and the operating costs. Additionally, tenants may have erratic workload patterns that negatively impact the SLA guarantees. Therefore, a promising solution for service providers is to replicate and migrate the tenants databases, which is beneficial to service availability, performance, flexibility and quality. In this paper, a new clustered based multi-tenant

database management system (CB-MT DBMS) is proposed. In addition, a dynamic proactive multi-tenant database migration and replication MTDB-MR algorithm is proposed which uses prediction results to enhance the anticipated need for migration and replication of multi-tenant data and to meet the multi-tenant quality of service by avoiding service level agreement violations. Experimental results show that the proposed MTDB-MR algorithm results in a significant reduction in the average response time by average 36.68% and total number of violations by average 89% for more than tenant client site and also results in a significant reduction in the total execution time by average 44.74% when compared to previous basic strategy. As a future work, the proposed MTDB-MR algorithm will be extended to use different prediction models in the forecasting service.

REFERENCES

[1] A. E. Abdel Raouf, N. L. Badr, and M. F. Tolba, "Dynamic data reallocation and replication over a cloud environment," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 13, Jan. 2018, Art. no. e4416.
[2] E. A. A. Raouf, L. N. Badr, and M. F. Tolba, "Distributed database system (DSS) design over a cloud environmen," in *Multimedia Forensics and Security*, vol. 115, A. Hassanien, M. M. Fouad, A. Manaf, M. Zamani, R. Ahmad, J. Kacprzyk, Eds. Cham, Switzerland: Springer, 2017, pp. 97–116.
[3] S. Barker, Y. Chi, H. J. Moon, H. Hacigümüş, and P. Shenoy, "'Cut me some slack': Latency-aware live migration for databases," in *Proc. ACM Int. Conf. Proc. Ser.*, New York, NY, USA, 2012, pp. 432–443.
[4] P. A. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, D. B. Lomet, R. Manne, L. Novik, and T. Talius, "Adapting Microsoft SQL server for cloud computing," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 1255–1263.
[5] J. Blad, "Analysis and performance evaluation of the data-layer in a multi-tenant architecture," M.S. thesis, School Comput. Sci. Commun. (CSC), KTH, Stockholm, Sweden, 2016.
[6] R.-S. Chang and H.-P. Chang, "A dynamic data replication strategy using access-weights in data grids," *J. Supercomput.*, vol. 45, no. 3, pp. 277–295, Sep. 2008.
[7] A. Floratou and J. M. Patel, "Replica placement in multi-tenant database environments," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2015, pp. 246–253.
[8] I. Hababeh, "Improving network systems performance by clustering distributed database sites," *J. Supercomput.*, vol. 59, no. 1, pp. 249–267, Jan. 2012.
[9] *SqlQueryStress Performance Testing Tool*. Accessed: Nov. 9, 2021. [Online]. Available: https://github.com/ErikEJ/SqlQueryStress/wiki
[10] *Benchmark Samples' Queries*. Accessed: Nov. 9, 2021. [Online]. Available: https://github.com/hortonworks/hive-testbench
[11] Y. Ji, Z. Lin, and T. Rong, "AdaptiveSLA: A two-stage scheduling framework for SLA profit maximization in multi-tenant database," *J. Phys., Conf. Ser.*, vol. 1187, no. 5, Apr. 2019, Art. no. 052002.
[12] T. Liu, Q. Li, L. Kong, L. Liu, and L. Cui, "Optimizing replica exchange strategy for load balancing in multienant databases," in *Web-Age Information Management* (Lecture Notes in Computer Science), vol. 9659, B. Cui, N. Zhang, J. Xu, X. Lian, and D. Liu, Eds. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-39958-4_34.
[13] C. S. Marinho, O. L. Moreira, F. E. Coutinho, S. J. C. Filho, F. R. Sousa, and C. J. Machado, "LABAREDA: A predictive and elastic load balancing service for cloud-replicated databases," *J. Inf. Data Manage.*, vol. 9, no. 1, p. 94, Jun. 2018.
[14] R. Mokadem and A. Hameurlain, "A data replication strategy with tenant performance and provider economic profit guarantees in cloud data centers," *J. Syst. Softw.*, vol. 159, Jan. 2020, Art. no. 110447.
[15] H. J. Moon, H. Hacigümüş, Y. Chi, and W.-P. Hsiung, "SWAT: A lightweight load balancing method for multitenant databases," in *Proc. 16th Int. Conf. Extending Database Technol. (EDBT)*, 2013, pp. 65–76.
[16] L. O. Moreira, V. A. E. Farias, F. R. C. Sousa, G. A. C. Santos, J. G. R. Maia, and J. C. Machado, "Towards improvements on the quality of service for multi-tenant RDBMS in the cloud," in *Proc. IEEE 30th Int. Conf. Data Eng. Workshops*, Mar. 2014, pp. 162–169.
[17] J. Ni, G. Li, L. Wang, J. Feng, J. Zhang, and L. Li, "Adaptive database schema design for multi-tenant data management," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2079–2093, Jun. 2013.
[18] S. Sakr and A. Liu, "SLA-based and consumer-centric dynamic provisioning for cloud databases," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 360–367.
[19] W. Sellami, H. H. Kacem, and A. H. Kacem, "Dynamic provisioning of service composition in a multi-tenant SaaS environment," *J. Netw. Syst. Manage.*, vol. 28, no. 2, pp. 367–397, Apr. 2020.
[20] S. Souravlas and A. Sifaleras, "Trends in data replication strategies: A survey," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 34, no. 2, pp. 222–239, Mar. 2019.
[21] F. R. C. Sousa and J. C. Machado, "Towards elastic multi-tenant database replication with quality of service," in *Proc. IEEE 5th Int. Conf. Utility Cloud Comput.*, Nov. 2012, pp. 168–175.
[22] F. R. C. Sousa, L. O. Moreira, J. S. Costa Filho, and J. C. Machado, "Predictive elastic replication for multi-tenant databases in the cloud," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 16, Aug. 2018, Art. no. e4437.
[23] R. Taft, W. Lang, J. Duggan, A. J. Elmore, M. Stonebraker, and D. DeWitt, "STeP: Scalable tenant placement for managing database-as-a-service deployments," in *Proc. 7th ACM Symp. Cloud Comput.*, Oct. 2016, pp. 388–400.
[24] M. Tang, B.-S. Lee, C.-K. Yeo, and X. Tang, "Dynamic replication algorithms for the multi-tier data grid," *Future Gener. Comput. Syst.*, vol. 21, no. 5, pp. 775–790, May 2005.
[25] U. Tos, R. Mokadem, A. Hameurlain, T. Ayav, and S. Bora, "A performance and profit oriented data replication strategy for cloud systems," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 780–787.
[26] Z. Wang, T. Li, N. Xiong, and Y. Pan, "A novel dynamic network data replication scheme based on historical access record and proactive deletion," *J. Supercomput.*, vol. 62, no. 1, pp. 227–250, 2012.
[27] D. D. I. G. Amalarethinam, "A study on performance evaluation of peer-to-peer distributed databases," *IOSR J. Eng.*, vol. 2, no. 5, pp. 1168–1176, May 2012.
[28] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid," in *Proc. Int. Workshop Grid Comput.*, Berlin, Germany: Springer, 2001, pp. 75–86.
[29] J. Zhang, B.-S. Lee, X. Tang, and C.-K. Yeo, "A model to predict the optimal performance of the hierarchical data grid," *Future Gener. Comput. Syst.*, vol. 26, no. 1, pp. 1–11, Jan. 2010.
[30] C. S. Marinho, E. F. Coutinho, J. S. C. Filho, L. O. Moreira, F. R. Sousa, and J. C. Machado, "A predictive load balancing service for cloud-replicated databases," in *Proc. SBBD (Short Papers)*, Uberlândia, MG, Brazil, 2017, pp. 210–215.
[31] M. I. Roushdy and M. A.-B. Salem, "Intelligent clustering technique based on genetic algorithm," *Int. J. Intell. Comput. Inf. Sci.*, vol. 21, no. 1 pp. 19–32, 2021.
[32] A. Awad, M. I. Roushdy, R. A. E. ElGohary, and I. F. Moawad, "An interactive tool for extracting low-quality spreadsheet tables and converting into relational database," *Int. J. Intell. Comput. Inf. Sci.*, vol. 21, no. 1, pp. 1–18, 2021.
[33] K. Ibrahim, M. Aborizka, and F. Maghraby, "Prediction of users charging time in cloud environment using machine learning," *Int. J. Intell. Comput. Inf. Sci.*, vol. 18, no. 2, pp. 39–57, Apr. 2018.
[34] A. E. A. Raouf, A. Abo-Alian, and N. L. Badr, "A predictive replication for multi-tenant databases using deep learning," *Concurrency Comput. Pract. Exper.*, vol. 33, no. 13, Feb. 2021, Art. no. e6226.
[35] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "OLTP-bench: An extensible testbed for benchmarking relational databases," *Proc. VLDB Endowment*, vol. 7, no. 4, pp. 277–288, Dec. 2013.
[36] W. Hussain, F. K. Hussain, O. Hussain, R. Bagia, and E. Chang, "Risk-based framework for SLA violation abatement from the cloud service provider's perspective," *Comput. J.*, vol. 61, no. 9, pp. 1306–1322, Sep. 2018.
[37] V. Rampérez, J. Soriano, D. Lizcano, and J. A. Lara, "FLAS: A combination of proactive and reactive auto-scaling architecture for distributed services," *Future Gener. Comput. Syst.*, vol. 118, pp. 56–72, May 2021.
[38] V. Cardellini, E. Casalicchio, F. Lo Presti, and L. Silvestri, "SLA-aware resource management for application service providers in the cloud," in *Proc. 1st Int. Symp. Netw. Cloud Comput. Appl.*, Nov. 2011, pp. 20–27.
[39] W. Hussain, F. Hussain, and O. Hussain, "QoS prediction methods to avoid SLA violation in post-interaction time phase," in *Proc. IEEE 11th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2016, pp. 32–37.

[40] W. Hussain, F. K. Hussain, O. K. Hussain, E. Damiani, and E. Chang, "Formulating and managing viable SLAs in cloud computing from a small to medium service provider's viewpoint: A state-of-the-art review," *Inf. Syst.*, vol. 71, pp. 240–259, Nov. 2017.
[41] M. Liaqat, A. Naveed, R. L. Ali, J. Shuja, and K.-M. Ko, "Characterizing dynamic load balancing in cloud environments using virtual machine deployment models," *IEEE Access*, vol. 7, pp. 145767–145776, 2019.
[42] S. Mustafa, K. Sattar, J. Shuja, S. Sarwar, T. Maqsood, S. A. Madani, and S. Guizani, "SLA-aware best fit decreasing techniques for workload consolidation in clouds," *IEEE Access*, vol. 7, pp. 135256–135267, 2019.

AHMED E. ABDEL RAOUF received the M.Sc. degree in information systems from Ain Shams University, in 2017, where he is currently pursuing the Ph.D. degree with the Information Systems Department, Faculty of Computer and Information Science. He is also a Teaching Assistant with the Information Systems Department, Faculty of Computer and Information Sciences, Ain Shams University. His research and publication interests include cloud computing, distributed database, and database management.

ALSHAIMAA ABO-ALIAN received the Ph.D. degree in information systems from Ain Shams University, in 2016. She is currently a Lecturer with the Information Systems Department, Faculty of Computer and Information Sciences, Ain Shams University. Her research and publication interests include cloud computing, security, data mining, and database management.

NAGWA L. BADR received the B.Sc. degree in computer science in 1996 and the Ph.D. degree in software engineering and distributed systems from Liverpool John Moores University, U.K., in 2003. She had done postdoctoral studies at the University of Glasgow, U.K. She is currently a Professor and the Dean with the Faculty of Computer and Information Sciences, Ain Shams University. She is also the head of the committee that contributed in research projects funded by national and international grants of information systems, bioinformatics, business analytic, and health informatics (i.e., http://www.heal-plus.eu/). Her current research interests include software engineering, cloud computing, big data analytics, social networking, Arabic search engines, and bioinformatics.