Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Advanced consistency management of highly-distributed transactional database in a hybrid cloud environment using novel R-TBC/RTA approach

Jasmina Dizdarevic*†, Zikrija Avdagic†, Fahrudin Orucevic† and Samir Omanovic†

Abstract
This paper examines possibilities for improving the existing strategies of consistency management for highly-distributed transactional database in a hybrid cloud environment. With a detailed analysis of the existing consistency models for distributed database and standard strategies including Classic, Quorum and Tree Based Consistency (TBC), it is concluded that an improved advanced model of so-called visible adaptive consistency needs to be applied in a highly-distributed cloud environment, as necessary and sufficient degree of synchronization of all replicas. Along with the proposed model, research and development of an advanced novel strategy for consistency management Rose TBC (R-TBC) approach has been conducted, by improving standard TBC approach. Regarding implementation, a specific agglomerative Rose Tree Algorithm (RTA) has been developed, based on Bayesian hierarchical clustering and Graph Partitioning Algorithm - Multidimensional Data Clustering (GPA-MDC) intelligent partitioning of transactional Cloud Database Management System (CDBMS). The final result is constructed R-TBC model that changes in accordance with dynamic changes of entire heterogeneous CDBMS environment.

Keywords: Consistency management, Rose Tree Based Consistency model, Cloud Database Management System, Intelligent partitioning, Hybrid cloud

Introduction

One of the most important aspects related to management of complex, highly-distributed transactional database systems in a heterogeneous cloud environment is application of well-known ACID (Atomicity, Consistency, Isolation and Durability) rules, in particular its consistency rule. Furthermore, it is crucial to preserve ACID rules without degrading the key features of the cloud platform: scalability, availability and reliability [1, 2]. The higher the number of replicas in the cloud environment, the more difficult is to achieve a desired degree of data consistency in Cloud Database Management System (CDBMS). Maintaining all the replicas simultaneously up-to-date, results with significant degradation in performance, as well as increased number of unsuccessfully executed transactions in the entire distributed Database Management System (DBMS) [3, 4]. However, all these mentioned problems are exactly in accordance with the Brewer's CAP (Consistency, Availability, Partition-Tolerance) theorem [5, 6] claiming that in a distributed environment it is not possible to achieve all three of the key features simultaneously, and thus give guarantees on consistency, availability and network partitioning of a highly-distributed system. Thus, in order to achieve targeted system performance and contracted Quality of Service (QoS) of Data as a Service (DaaS)/Database as a Service (DBaaS) launched from the cloud platform, it is possible to achieve only two of

*Correspondence: jasmina.dizdarevic@etf.unsa.ba
†Jasmina Dizdarevic, Zikrija Avdagic, Fahrudin Orucevic and Samir Omanovic contributed equally to this work.
University of Sarajevo, Faculty of Electrical Engineering, Computer Science and Informatics Department, Zmaja od Bosne bb, Sarajevo, Bosnia and Herzegovina

three selected properties [5]. Since network partitions in a distributed environment are common and logical, the application of this theorem in practice results in a compromise between two remaining properties: consistency and availability. Realizing that the unavailability of a service in cloud environment is in fact unacceptable since it affects its basic functionality, thus consistency property is left open for further consideration and rationing in accordance with the dynamic changes of system, user requirements and other relevant cloud DBMS environmental factors.

Standard strategies for preserving and managing the consistency of a highly-distributed transactional database in a hybrid cloud environment were analyzed and discussed in "Standard strategies for consistency management of a highly-distributed transactional DBMS database in cloud environment" section of this paper, in order to achieve the main goal of undertaken research dynamic and adaptive consistency management of the entire distributed system. The effects of application of proposed novel consistency management strategy or advanced Rose Tree Based Consistency (R-TBC) model approach are thoroughly observed and analyzed specifically for energy sector companies, collaborating within heterogeneous hybrid cloud sharing common highly-distributed transactional database.

Standard strategies for consistency management of a highly-distributed transactional DBMS database in cloud environment

In order to achieve a primary goal of dynamic consistency management for transactional DBMS within a highly-distributed cloud environment, extensive research of the current state of domain was conducted, including existing standard models and strategies for maintaining consistency with the application in distributed environments [7–9]. Regarding consistency management of distributed database systems, several fundamental consistency models exist, including: Strong [10], Weak [11] and Eventual Consistency (EC) [12] models, as well as other variations of weak consistency model (casual, monotonic reads/writes, session, and so on). Strong Consistency or Linearization is the strongest consistency model. Each operation must appear committed immediately, and all clients operate over the same valid data state. Strong Consistency leads to a high-level consistency system, but it compromises scaling by decreasing availability and network partition tolerance. Although Strong consistency is the ideal requirement for transactional DBMS, it deeply compromises horizontal-scalability which is very important property of highly-distributed cloud environments since it enables higher throughput and replication of data across distinct database nodes [10]. Weak Consistency model, as the name implies, weakens the consistency of distributed database system. It states that a read operation does not guarantee the return of the latest value written [11]. The most commonly used model in cloud environment is actually the Eventual Consistency model which is half-way a Strong consistency model and a Weak consistency model. It states that all replicas gradually become consistent and tend to converge to the same data state if no write operation occurs. This means that all user inquiries or requests addressed to a database become consistent only after some time has passed. While this convergence process runs, it is possible for read operations to retrieve an older version instead of the latest one [12].

In this regard, at the very beginning of this research the conventional strategies for consistency management of DaaS/DBaaS services were considered and thoroughly examined, including the Classic [6], Quorum [11, 12] and Tree-Based Consistency (TBC) [13, 14] strategy, then further explored the possibilities for generalizing these existing standard approaches and strategies, with potential for their improvements.

In the Classic strategy or approach [6] for consistency management of a highly-distributed CDBMS environment, one replica node (master or root - usually the environment controller) is selected to be responsible for communicating with all other replica nodes and notifying them of the latest system updates. Thus, the root node is responsible for monitoring the execution and distribution of update operations to all replicas nodes of the environment. It means that all replicas must be up-to-date, before the next read operation of the distributed database data is started with its execution. Consequently, this increases the response time of the system or DaaS/DBaaS service. The first layer of replicas and direct descendants of the root node will result in a reduction in workload, since they only process issued write/update operations of the distributed database. Conversely, the secondary layer of replica nodes will result in an increase in the workload volume, since all read (but also write) operations of the distributed database are forwarded directly to these nodes.

Figure 1 illustrates an example of the execution of read ('Read', R) and write ('Write', W) operations within the Classic consistency management approach for distributed cloud DBMS environment. According to the Classic model [6], the data value is not returned and forwarded for use to the next read or write operation, until all replicas of the distributed environment return the same value. This means that when the execution of update operation on the distributed database is required, then all replicas of the environment must be up-to-date in order for the next request to be taken into consideration and released for the execution within the system. Figure 1 shows the general principle of the Classic consistency management approach, on the example of the execution

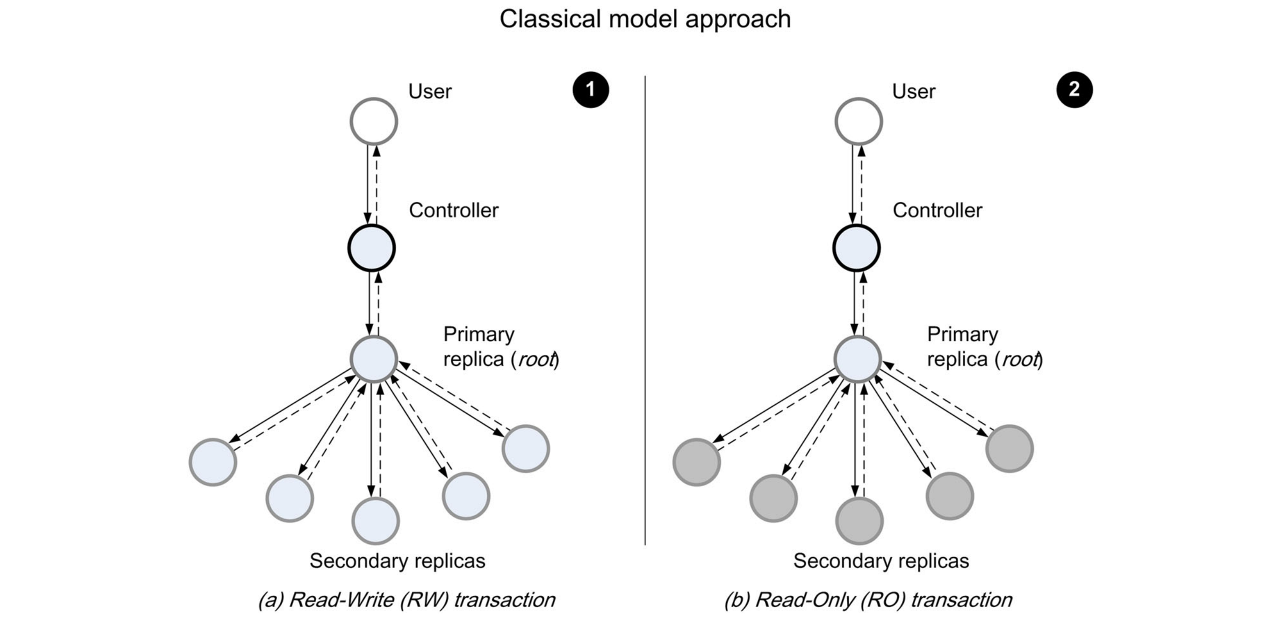


Fig. 1 Standard approach for consistency management - an example of the execution of read and write operations within the Classic approach [13]

of write ('Write', W) and read ('Read', R) operations within the distributed environment. The presented cluster configuration consists of six (6) replica nodes, along with the primary node or root replica. The entire established distributed environment of replica nodes is managed by a controller - a special node that performs, among other multiple functions, forwarding requests issued by system users to the root node (root) or its successors depending on current requirements and the degree of synchronization of the entire distributed environment [6, 13]. In this way, implemented replication within distributed environments based on the Classic strategy supports a Strong consistency model [10].

In practice, the Classic approach [6, 10] proved to be very suitable for serving distributed applications that include a lot of read-only (RO) operations and a small number of write (RW) operations. Thus, only in these cases, the system shows satisfactory response time and overall performance, which is not the general case with the Classic approach and represents one of its major disadvantages.

The next standard approach or strategy for consistency management of a highly-distributed CDBMS environment is the Quorum replication [11, 12], which is based on the consensus quorum. In this approach, received user requests are processed through an established replica quorum, before the results of the request are returned back to the system user. The quorum Q is defined as the group of the most nodes of the environment so that the relation ($Q > N/2$) is valid, where N is the total number of replicas (or nodes) and Q are the members of the quorum. The general principle of this protocol or quorum replication is that all operations are processed and approved by the most node replicas, members of the quorum Q, before returning the results to the end user, thus giving guarantees of access to the latest updated data items of CDBMS.

The main algorithm for implementation of the Quorum consistency model [13] is based on allowing the client to update any of the replica(s), and then this same data or replica update information is passed on and distributed in the background to other replica nodes, using so-called gossip protocol (based on the principle of "gossip spreading"). Since uncommitted updates can arrive at different replicas in different order of operations, this requires the implementation of an effective conflict-resolution mechanism that may occur during the execution of operations. Thus, during the actual voting of the quorum members, a decision can be made only when the majority of the members of the elected quorum agree with a certain decision. Accordingly, a distributed system based on the Quorum consistency management model [13] requires more than half of the replicas of the members (servers) to complete read or write operations, before the distributed database data items are available for the execution in the next read or write operation, as shown in Fig. 2. Also, it is important to note that there is no specifically designed and selected primary or secondary replica in Quorum-based systems. Namely, each replica can operate as a primary, and read or write operations are periodically sent to each replica node of the established distributed environment, for the execution and further synchronization. In this

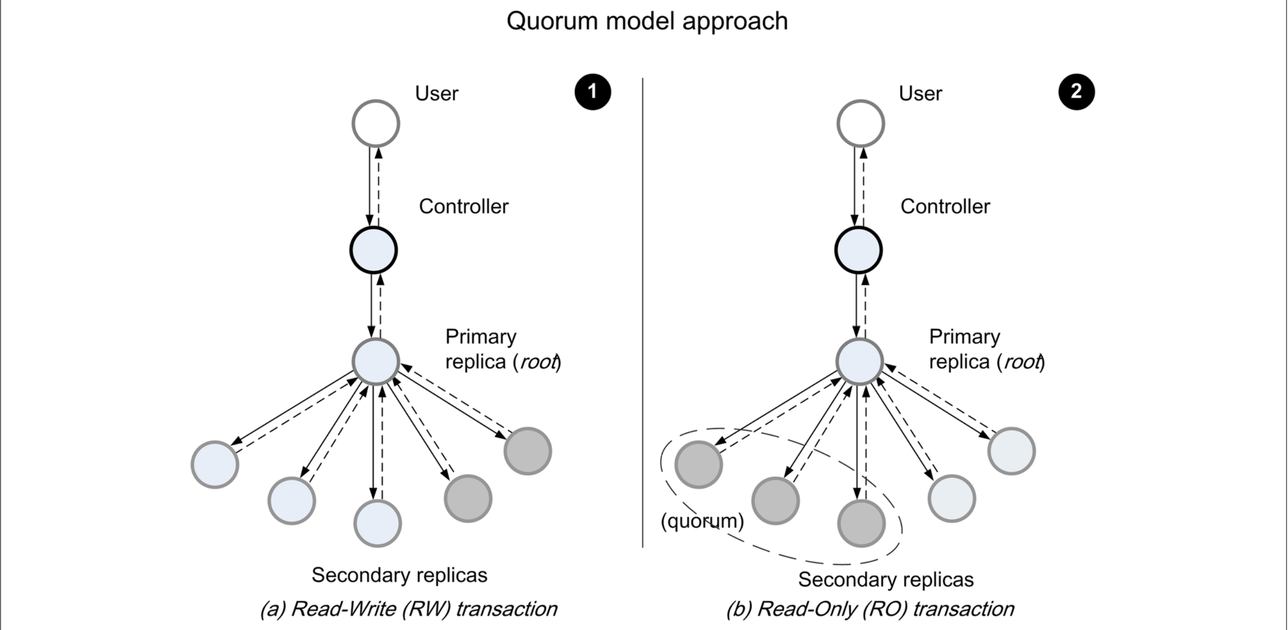


Fig. 2 Standard approach for consistency management - an example of the execution of read and write operations within the Quorum approach [13]

way, implemented replication within distributed environments based on the Quorum strategy supports a Weak consistency model [11, 12].

Also, it is important to note that the Quorum approach shows a better response time compared to the Classic approach, specifically in applications and distributed systems that perform a large number of write (RW) operations [13]. However, the requirement for obtaining the consent or quorum consensus from all included replicas, when executing each individual DB operation, can significantly slow down the overall system performance and thus increase the response time of the DaaS/DBaaS services.

To achieve a degree of the Eventual Consistency [12] in cloud environment, most of today's commercial DaaS/DBaaS solutions use the Quorum protocol or strategy for maintaining and preserving data consistency of distributed database. However, the Quorum protocol proved to be unsatisfactory and in most scenarios results in a significantly longer response time comparing to the Classic protocol. With detailed elaboration of this model and research of its applications, it was concluded that using the most exploited model for consistency management and preservation cannot provide necessary and sufficient levels of the consistency for entire distributed system. This is especially manifested through often conflicting situations of multiple user queries and requests addressed to a CDBMS, as well as evident inaccuracy and invalidity of data affected by the same issued queries [12].

Conversely, the TBC approach [13, 14] provides optimized performance for a highly-distributed transactional database in cloud environment, and also a significant advantage over other conventional approaches (Classic, Quroum) [6, 11] for managing and preserving the consistency of a distributed system. TBC is a structural, tree-based approach that promotes an adaptive model in which the consistency changes dynamically throughout the TBC tree and actually declines from a Strict or Strong [10] to a Eventual Consistency [11, 12] manifested on the leaves of the tree. This leads to the concept of a more advanced model in the form of so-called “visible” or Apparent Consistency (AC) as a necessary and sufficient degree of synchronization of all replicas and related nodes of the transactional cloud DBMS database. From the side of the end user of a cloud web application or DaaS/DBaaS service based on cloud transactional DMBS, “visible” consistency is quite sufficient since the highest levels or layers, i.e. synchronized database replicas are always consistent with a complete, integral transactional cloud database. The end user, in fact, does not even know about the temporary inconsistency of invisible nodes or layers of replicas of the distributed environment. This is primarily because it remains hidden in the lower parts of the entire system hierarchy with replicated nodes. This temporary inconsistency is automatically and procedurally processed throughout background synchronization technique.

In practice, it is shown that maintaining the consistency of the cloud environment, using the TBC approach, is determined mostly by the number of replicas in the system, but also by delays in communication links and

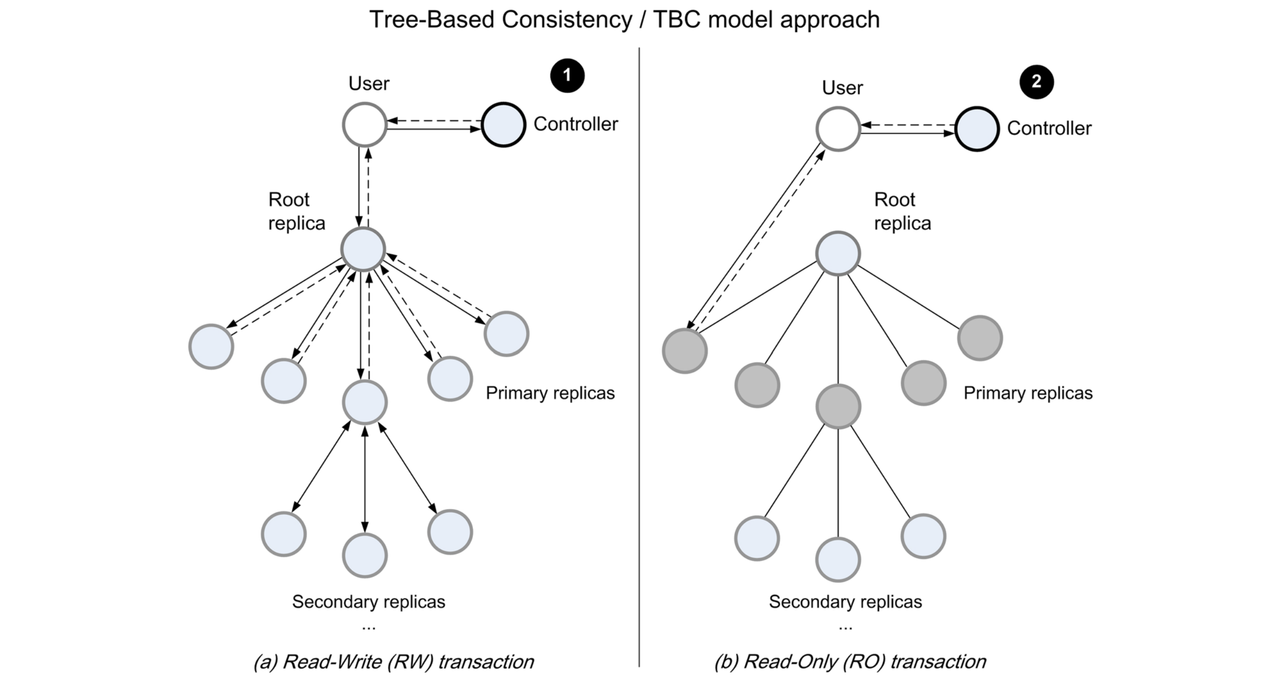


Fig. 3 Standard approach for consistency management of a highly-distributed environment - an example of the execution of read and write operations within the TBC approach [13]

the overall system workload. As can be seen from Fig. 3, within the TBC approach for consistency management of the distributed environment, all paths that replicas use to communicate with each other and propagate user requests with update operations are defined and established first. Also, one of the nodes is designated as the primary or root node of the environment. In particular, upon receipt of a request with read or write operations over data items in the distributed database, a replica should send a notification to its own and assigned subset of replica nodes. Thus, upon receipt of a request with update or write ('Write', W) operations over some data items Y, the root node notifies its immediate (or direct) descendants i.e. child nodes in the TBC tree structure, in order to update their own replicas or assigned copies of distributed CDMBS with the same values of data items Y. Replicas proceed with the execution of these updates, and upon completion of operations send a confirmation or 'ACK' to the root node or primary replica of the environment, on successfully completed update operations on data items Y. After that, the next read or write operations are taken into consideration and further execution within the distributed environment, as shown in Fig. 3.

Regarding the execution of the user requests with read ('Read', R) operations over data set Y, only the upper layers of the TBC tree are referred, i.e. the primary replica (root node) or set of its immediate (direct) successors, and returned the value of the requested data items Y to the end user. Thus one of these nodes (descendants) sends an acknowledgment 'ACK' to the root node or primary replica of the environment, to inform it on the successfully completed updates over data items Y, as the same update operations are propagated throughout the TBC tree. This is a signal to the primary node that it can return the requested values of data items Y back to the end user, and that the requested data is guaranteed to be consistent. All subsequent write operations require the root node to notify its immediate successors again in order to complete the write operations. Read operations are not propagated through the TBC tree, but are retained or executed at the first two (highest) levels or layers of the tree (root node and first descendants), so it is guaranteed return of consistent and valid data. Specifically, at these highest levels of TBC tree, guarantees are given for the Strong or Strict consistency [10] on data items of the replicated CDBMS in cloud environment. Therefore, a faster system response time is expected when it comes to read operations, and a slightly slower response time for write operations on data items of the distributed database.

Results show that the TBC approach has much better response time when compared to the other two leading conventional approaches (Classic, Quorum), regardless of the various relevant performance factors of the entire cloud environment, such as: the arrival rate of issued operations in the system, read-write (RO/RW) distribution ratio of operations, variations in targeted data selection, database size, etc. Also, the TBC approach reduces the interdependency between replica servers of the entire cloud and highly-distributed heterogeneous database system, which results in a shortened response time of cloud database, while generally maximizing the performance of launched user applications and services. Furthermore, application of the TBC approach causes a significant decrease of so-called "Inconsistency Window" (IW) as the key characteristic concerning consistency management of a highly-distributed system [13]. Finally, surveys in the relevant literature [14, 15] show that maintaining the consistency of cloud environment with the application of TBC approach is determined not only by the number of system replicas but also delays on communication links and overall system or network load. Therefore, in order to reduce the effect of replica servers interdependency on overall system performance, TBC sets a limit on the maximum number of allowed "children" for each replica server or "parental" node within the TBC tree structure of entire distributed environment. It is necessary to take into account the fact that increasing the number of "children" per replica server requires each "parent" node to wait until all its "children" complete their current updates on copies or assigned partitions (fragments) of distributed cloud database. This, in effect, means that in the TBC approach replica servers are interdependent relating execution of write operations (RW) or updates on the cloud database, which certainly has a significant impact on overall system performance and launched DaaS/DBaaS services. Consequently, this fact promotes the Modified TBC (MTBC) approach [13, 14] that is designed specifically to reduce the Inconsistency Window (IW) of the distributed transactional DBMS in cloud environment. Thus, using the MTBC approach the effect of the IW for entire distributed cloud system is practically minimized.

Furthermore, a detailed analysis of Performance Evaluation Metric (PEM) [16] for highly-distributed transactional CDBMS as well as thorough identification of key PEM network parameters were conducted, and then the role of these parameters in maintaining the consistency of cloud database for all three conventional approaches (Classic, Quorum, TBC/MTBC). Since Classic and Quorum strategies have not shown sufficient performances in managing and preserving consistency but also the other PEM parameters of highly-distributed transactional DBMS in cloud environment, a further research is focused exclusively on TBC strategy or Tree-Based Consistency approach for managing and preserving the consistency of cloud transactional database.

Development of advanced strategy for consistency management of highly-distributed transactional database

The existing TBC approach proposed in [13–15] represents one of the most advanced standard approaches for managing and preserving the consistency of a highly-distributed transactional database. It provides the necessary and sufficient guarantees on data consistency as well as high performance of the entire cloud environment. This approach is based on the construction of the TBC tree of consistency as optimally organized structure of replica nodes for the entire cloud environment. Thus, the consistency dynamically changes throughout the TBC tree, and actually decreases from the Strong consistency [6] on the upper layers to the Eventual consistency [12] that is evident on the leaves of the tree. This results in introducing a new and advanced consistency model-so-called "visible" or Apparent Consistency (AC). Also, the TBC approach is designed specifically for network environments, taking into account the PEM metrics and relevant factors or network parameters while in the same time reducing the interdependency and necessary communication between replica nodes. But most importantly, the TBC approach preserves and maintains the cloud transactional database in a continuously consistent state. Taking into account the specifics of the TBC approach, the implementation must be performed with the application of a wide set of different tools and complementary mechanisms, protocols, algorithms, etc. in order to achieve the targeted levels of AC for CDBMS. TBC approach with data categorization ensures dynamic management and rationalization of consistency across the entire CDBMS, with a degree of variation of this property depending on business needs, transactional workload, network bandwidth, type of transactions, ABC data categories [17], and many other identified PEM factors [16].

As already mentioned, the TBC approach [13–15] has a significantly better response time than conventional consistency management approaches (Classic, Quorum), regardless of the arrival rates of user requests, the ratio of read-write operations, variations in data selection and preference, database size, and other PEM network factors. In this regard, it is very important to identify relevant environmental factors that affect the overall performance of the system, but also the construction of TBC tree. Thus, when propagating an update operation within distributed database, several PEM network factors of cloud DaaS/DBaaS service are taken into account, including: disk update time, replica node workload, network workload, bandwidth and network reliability, link speed, network traffic and many others. These factors are especially taken into account during the construction of TBC tree with replica nodes for cloud highly-distributed environment and generally play the most important role in this process.

In this way, the levels or degrees of consistency within TBC tree are variable and in direct correlation with the dynamic changes of the relevant PEM network parameters [16] of the entire environment. This means that a change in the environment is the cause of a change in the degree of consistency within the TBC tree. Therefore, the consistency levels of the cloud DBMS represent a real reflection of the actual state and parameters for the formed cloud transactional database environment. Also, the PEM environmental parameters determine the level of consistency for the transactional cloud database items. It is impossible for entire database to have high degree or level of consistency on data items if certain preconditions or most important environmental factors are not satisfied. Therefore, the possible lack of consistency of the distributed database first should be addressed at the root of the problem - key PEM factors or performance parameters of the transactional DBMS cloud. Therefore, lower levels of consistency indicate that something has changed or degraded in the structure and organization of the entire cloud network, the number of nodes or quality of server replicas like reliability, hardware characteristics, etc., then quality and speed of network connections, average percentage of packet loss, degree of network congestion, user requirements in the form of an unplanned, highly increased input transactional load, etc.

In particular, it should be taken into account that the cloud infrastructure is usually built using heterogeneous systems and network components, which leads to a great diversity of embedded systems, data formats and transfer standards, communication protocols, etc. Also, it is often the case that some communication links are slower or unreliable, while other parts (or segments) of the network are much faster. Therefore, in some parts of the cloud heterogeneous network, there may be more frequent bottlenecks and congestion occurrences, for example due to heavy loads in network traffic, and thus the formation of the requests queues with write/update operations on distributed database and likewise. On some other parts of the cloud network, there may be periodical outages, e.g. communication links or server units (or even clusters of replicas), which all together lead to an increase in response time and consequently decrease the performance of a DaaS/DBaaS service and entire distributed system. This degradation in the performance of the cloud transactional database or DaaS/DBaaS service has a very significant impact on the strategy of maintaining the consistency of a highly-distributed cloud DBMS environment. The proposed advanced R-TBC/RTA approach, which represents an improvement and extension of the existing standard TBC approach, solves a number of these mentioned problems. This is primarily because it is designed for application within highly-distributed, networked environments and its intensive exploitation of advanced hierarchical data structures such as Bayesian BRT Rose tree [18–20] within entire cloud environment. Therefore, the proposed advanced R-TBC/RTA approach takes into account and relies on the foundation of the PEM network metrics with the most relevant network parameters, which is more discussed and analyzed in "PEM metric for managing the structure of the R-TBC tree with replica servers of cloud environment" section. Thus, this advanced approach achieves minimal performance degradation while maintaining the desired degree of consistency for transactional cloud DBMS database and highly-distributed heterogeneous network [21–23].

Standard TBC approach is based on the application of the Modified Dijkstra (MD) Shortest Path (SP) algorithm, with a limitation to a maximum number of two children for each single node of the final tree structure. The application of the MD SP algorithm for the construction of TBC tree is based on search for the shortest path between server replicas of the environment. The problem of the shortest path in the weighted interconnection graph means the search for the path between the two selected vertices with the lowest weight (i.e. weight coefficients). The algorithm detects the shortest path from a specific vortex, denoted as 1, to all other vertices, denoted as 2,3...n in the graph G. Modified Dijkstra algorithm is implemented in the main control node of the environment, and shows a rather satisfying performance within the standard TBC tree consistency management approach.

Hierarchical data structures in the TBC/R-TBC tree construction process

Formed binary TBC tree of "visible" consistency is shown as an infused and illusory data structure, since it does not usually reflect the real state and inherent structures and models contained in the background data sets. Usage of the Bayesian tree structure model based on the Bayesian Rose Tree (BRT) model and hierarchical clustering [18–20], with an arbitrary number of nodes children in the whole structure, results with the formation of an optimized structures. Limitation with the number of allowed children for parent node is similar as in the process of constructing the TBC tree of consistency. Produced data models are with much greater likelihood of representation of analyzed data sets and with better distribution in the search space. In this paper, an advanced Rose Tree approach (R-TBC/RTA), based on the R-TBC model of consistency and intelligent partitioning of highly-distributed transactional database in a hybrid cloud environment, is proposed and presented as an improvement compared to the existing consistency strategies.

Figure 4 shows comparison of the two leading models of hierarchical clustering - Bayesian Hierarchical Clustering (BHC) Binary model and Bayesian BRT Rose Tree model. It is very obvious that Bayesian hierarchical structure is optimized, better distributed in the search space, with better utilization of memory and other computing resources and capabilities, and with a much smaller resultant set

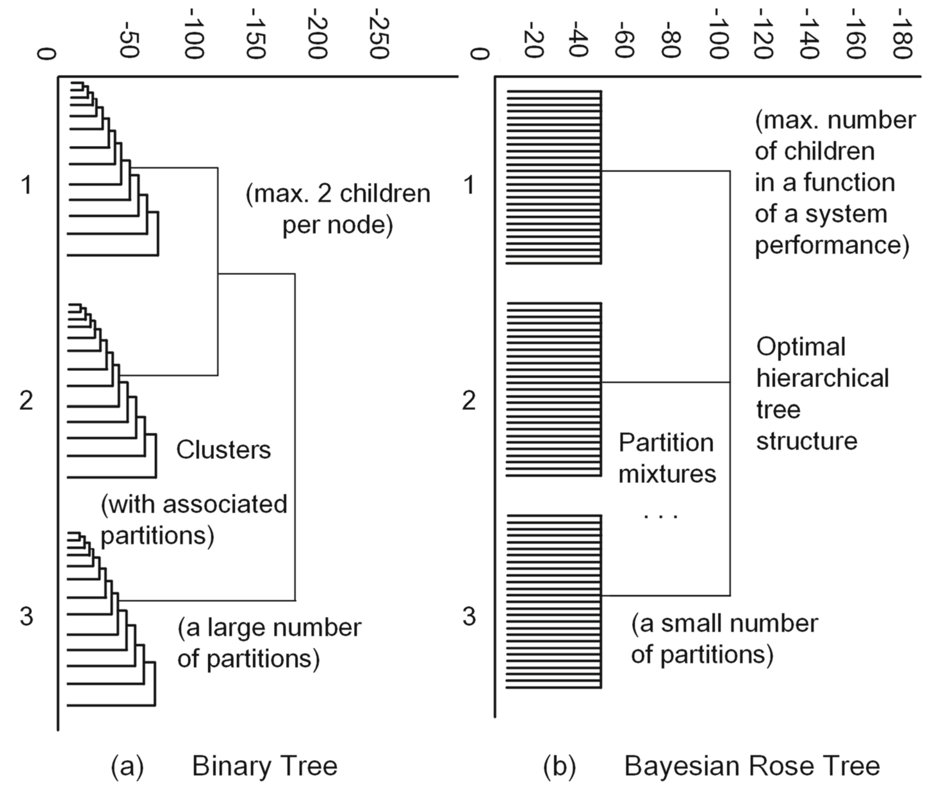

0
-50
-100
-150
-200
-250
1
2
3
(max. 2 children per node)
Clusters
(with associated partitions)
(a large number of partitions)
0
-20
-40
-60
-80
-100
-120
-140
-160
-180
(max. number of children in a function of a system performance)
Optimal hierarchical tree structure
Partition mixtures
. . .
(a small number of partitions)
(a) Binary Tree
(b) Bayesian Rose Tree

Fig. 4 Binary hierarchical data structure (left) vs. BRT Bayesian Rose Tree structure (right) for the same synthetic data set [20]

of generated partitions of the final Rose Tree, comparing to Binary BHC. In general, research [20] shows that the resulting partition sets in the BHC and the BRT models significantly differ in size, in most applications and scenarios of their implementation. Thus, the first and fundamental difference between the existing, standard TBC and proposed advanced R-TBC approach for managing and preserving consistency of transactional cloud database is based on the exploitation and application of different data structures in the execution of the TBC/R-TBC tree construction process.

The next essential difference, but also improvement in the proposed advanced R-TBC, relates to the steps or sequence of the RTA Rose Tree Algorithm for the construction of R-TBC consistency tree, as well as the domain of its application. Specifically, as can be noticed in the standard TBC MD SP-based approach [13, 14], the construction of the TBC tree primarily refers to cloud replica servers, taking into account PEM network performance factors [15, 16] and calculated metrics for each of the included replica servers in the final tree-structure. On the other hand, advanced R-TBC approach, based on the Rose Tree and the agglomerative RTA algorithm [18], constructs hierarchical tree structure not only with the replica servers of entire cloud environment, but also with the cloud transactional database and all its fragments (partitions). Thus, there is a certain parallelisam between the standard TBC/MTBC and the proposed advanced R-TBC/RTA consistency approach since both perform construction process of the fundamental tree structure with main difference in scope of the process, as shown on upper part of Fig. 5. However, the standard TBC approach does not deal with the partitioning and data placement (distribution) of generated partitions of the transactional database across the replica servers of the environment, as shown on lower part of Fig. 5. This actually represents the main advancement of the proposed advanced R-TBC/RTA consistency management approach. Relating the standard TBC/MTBC consistency approach, the process of partitioning and autoscaling of the cloud DBMS environment is only generally introduced. Thus, the existing, standard TBC approach is limited to the formation of the TBC tree with the replica servers of the cloud environment, but it does not deal with cloud transactional database and its fragments. On the other hand, advanced R-TBC approach, based on the Rose Tree and the agglomerative RTA algorithm [18], constructs hierarchical tree structure not only with the replica servers of entire cloud environment, but also with the cloud transactional database and all its fragments (partitions).

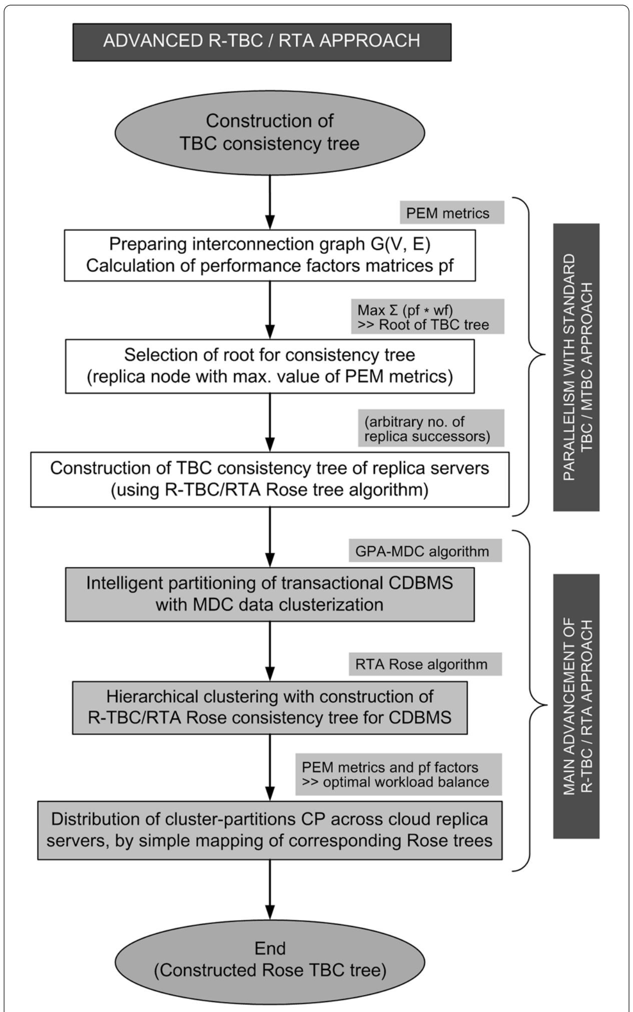

ADVANCED R-TBC / RTA APPROACH
Construction of TBC consistency tree
PEM metrics
Preparing interconnection graph G(V, E) Calculation of performance factors matrices pf
Max Σ (pf ∗ wf) >> Root of TBC tree
Selection of root for consistency tree (replica node with max. value of PEM metrics)
(arbitrary no. of replica successors)
Construction of TBC consistency tree of replica servers (using R-TBC/RTA Rose tree algorithm)
PARALLELISM WITH STANDARD TBC / MTBC APPROACH
GPA-MDC algorithm
Intelligent partitioning of transactional CDBMS with MDC data clusterization
RTA Rose algorithm
Hierarchical clustering with construction of R-TBC/RTA Rose consistency tree for CDBMS
PEM metrics and pf factors >> optimal workload balance
Distribution of cluster-partitions CP across cloud replica servers, by simple mapping of corresponding Rose trees
MAIN ADVANCEMENT OF R-TBC / RTA APPROACH
End (Constructed Rose TBC tree)

Fig. 5 Proposed advanced R-TBC/RTA approach and specific sequence of steps in the process of construction the R-TBC Rose Tree of consistency for a highly-distributed transactional database in a hybrid cloud environment

Generally, main differences between BHC Binary Hierarchical Clustering and BRT Bayesian Rose tree structural data model are presented in the Table 1. It is concluded that BRT data model shows significant advantages comparing to standard BHC data model - used in the most database applications and todays implementations [18, 20].

The process of construction the R-TBC tree using the proposed advanced R-TBC/RTA consistency management approach

The following Fig. 5 presents the main contribution of the research presented in this paper, the advanced R-TBC/RTA approach with specific sequence of steps in the process of constructing the Rose Tree of consistency, as well as the differences and improvements in relation to the standard TBC (Tree-Based Consistency) approach for managing and preserving consistency of the transactional cloud database:

Furthermore, the resulting R-TBC Rose Tree includes not only cloud DBMS data items but also includes other components of the entire environment (for example: services and application software, organizational units, and so on). In general, the proposed advanced R-TBC/RTA approach and the Rose Tree extend to the entire cloud heterogeneous hybrid environment as a unique and representative hierarchical network model of all its internal structures and elements. It is already mentioned that there is a certain parallelism between the standard TBC/MTBC and the proposed advanced R-TBC/RTA approach, as shown on Fig. 5. More precisely, both approaches have in common the first three steps related to the construction and formation of the Rose Tree which consists of replica servers of the cloud environment. Obviously, different algorithms were used, but the essence of the process is identical in both approaches.

At the very beginning, an interconnection graph G (V, E) representing the entire distributed cloud environment is prepared, and then PEM metric of performance factors (pf) is calculated for each of the replica servers [16]. The obtained PEM metric represents the input for the next step - root node selection for the final structure of the R-TBC tree. This node represents the replica server with the highest total for obtained PEM performance metrics. Therefore, it is very logical for this node to be selected as the root node of the R-TBC tree. After that, going through the other algorithmic steps and the main program loop, leads to the formation of the final R-TBC tree with replica servers of the cloud environment. This results with formation of a binary tree of replica servers (in the case of the TBC/MTBC algorithm) and a BRT tree [20] with replica servers of the same environment (in the case of the advanced R-TBC/RTA algorithm). With completing first three steps and exiting the main loop ends the process of executing the standard TBC/MTBC algorithm.

On the other hand, the R-TBC/RTA algorithm [18] continues its execution, and this is precisely the main enhancement of this innovative approach for managing the consistency of the heterogeneous cloud environment of a highly-distributed transactional database. Specifically, the following three steps perform construction and formation of the Rose Tree of consistency for transactional database and all its partitions, generated as a result of the process of intelligent partitioning and the application of the Graph Partitioning Algorithm - Multidimensional Data Clustering (GPA-MDC) algorithm, as shown on Fig. 5. The generated partitions and clusters of data serve again as input to the RTA, and are building blocks of the final Rose Tree of consistency for the cloud database. Upon completion of the construction and formation of

Table 1 Main differences between BHC Binary Hierarchical Clustering and BRT Bayesian Rose tree structural data model

	BHC Binary Hierarchical Clustering structure	**BRT Bayesian Rose Tree structure**
Applicability of structure	specialized	general
Quality of data structure	poor/good	very good
Emptiness of data structure	supported	not supported
Basic DML operations	supported	supported
Number of children	maximum two	arbitrary
Degree of node (in/out)	limited	unlimited
Number of subtrees	mainly two (left/right)	zero or many
Height of tree	log2 N (where N is the number of nodes)	logM N (where M is the order of tree)
Data order	ordered	unordered/ordered
Order criteria	single	zero or multiple
Resulting partition set	large	small
Performance requirements	performed when data is loaded in the RAM	performed when data is loaded in the disk and/or RAM
Execution time	logarithmic	exponential
Scope of appliance	mostly in coding and code optimizations	in DBMS internal structures and distributed environments

the R-TBC tree for the transactional database, the process of distribution the generated cluster-partitions (CP) over the replica servers of the cloud environment is initiated, shown as the last step on sequence diagram of Fig. 5. In simple terms, mapping and integration of appropriately formed Rose Tree occurs, for replica servers of the cloud environment and a transactional database. This leads to the construction of the final R-TBC consistency tree, which is a specific hierarchical structure, composed of both replica-servers of the entire environment with assigned responsibilities for partitions (or fragments) of the transactional database, as well as other essential components of the cloud environment.

Thus, the third important difference between the existing, standard TBC and this novel, advanced R-TBC/RTA approach consists in the execution process of the RTA algorithm [18] that have been substantially expanded with additional steps and thus functionally improved. After forming the R-TBC tree of consistency with replica servers of the environment, through the first three (3) steps of the algorithm, start the process of cloud database intelligent partitioning with Multidimensional Data Clustering (MDC), as shown on Fig. 5. In this step, first ABC categorization of data on sets of different consistency levels takes place [17]. At the same time comes horizontal data partitioning with generating larger range partitions, and then vertical partitioning by key attributes or data dimensions. This phase of the process is also called dimensionalization of cloud database. Thus, Hybrid data partitioning strategy that consists precisely of the efficient combination of horizontal with vertical partitioning, with the application of MDC, is introduced. More specifically, the intelligent data partitioning is based on horizontal cuts or cross sections of the R-TBC tree structure and then grouping of related table data from a backbone of highly-distributed cloud database. That produces partitions or fragments (rose subtrees) of groups i.e. clusters of related data. Vertical cuts or sections of these generated rose subtrees are then applied on groups of related and bound data, thereby generating sets of smaller (rose) subtrees or fragments (partitions) with even stronger interconnections, common attributes, and other features of the covered data groups. The MDC technique facilitates the management of complexity of the CDBMS environment by introducing dimensioning of contained data, based on previously defined relevant dimensions or data categories. Also, multidimensional clustering of background CDBMS data further shortens the execution time of queries and issued transactions over CDBMS data and contributes to improving the overall performance of the cloud environment.

So, this whole process results in the formation of cluster-partitions or rose subtrees (rosettes) that are linked or merged together by the hierarchical clustering into the construction of the R-TBC tree for the transactional cloud database. Finally, at the last phase of the process (see Fig. 5), the distribution of cluster-partitions of formed R-TBC tree across replica servers of the environment is performed simply by overlapping the corresponding rose trees (replica servers vs. database). Logically, the conclusion is that clusters of replica servers of a higher degree of PEM performance factors correspond to rose trees or CDBMS cluster-partitions of higher degree of consistency. There are many other advantages of the innovative R-TBC/RTA approach, but these three described differences represent the main improvements to the existing, standard TBC approach.

Pseudocode 1. presents steps within the sequential diagram with key phases in the process of executing the RTA algorithm for construction of the Rose tree of consistency on which advanced R-TBC/RTA approach is based (see Fig. 6):

In the sequence of steps of the RTA algorithm for the construction of optimal structure of hierarchical clustering performed on given data set i.e. the R-TBC Rose Tree (as it is shown in Pseudocode 1.), used parameters are defined as follows:

- $D = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ represents an input set of data items or nodes of the initial graph of the interconnection G that is modeled into the Rose Tree R of the optimal hierarchical structure,
- k represents the total number of clusters or individual rose subtrees T_i identified in the input data set D and generated as a result of applying a specific GPA-MDC intelligent partitioning algorithm. This completes the first phase of initialization of the entire construction process of the R-TBC tree,
- p_M or $p(x|\theta)$ represents a probability model of the Rose Tree and (input probabilities) of clusters of rose subtrees,
- p_H or $p(\theta|\beta)$ represents a priori probability of a hyperparameter beta for adjusting and tuning the final structure of the Rose Tree, as well as maximizing the probability of the R-TBC/RTA model,
- The main part of the algorithm is a recursive process with the operation of merging with the greatest probability model ratio p_M for selected rose subtrees T_i and T_j into an optimally merged (inter)structure of the resulting rose subtree T_m, with the optimal value of a hyperparameter beta probability p_H for adjusting and tuning the final structure of the R-TBC tree, using appropriate merge operation: 1. join, 2. absorption, 3. collapse,
- The final result of the algorithm is constructed Rose Tree with the optimal hierarchical structure T_m (T/R^T).

```
PSEUDOCODE 1: RTA( D, $p_M$, $p_H$ )
# Algorithm is performed over data items or nodes of interconnection graph G,
# taking into account probabilities and structure of the Rose Tree model
k ← n;   # Initialization of total number of rose subtrees $T_i$
$T_i$ ← $\{x^{(i)}\}$, where i=1,2,...,n;
$T_m$ ← {};   # Initialization of merged R-TBC tree

while k >1 do
    # Select pair of rose subtrees $(T_i, T_j)$ for merge operation m
    findRsubtrees(D, $(T_i, T_j)$);

    # Calculate the Tree model ratio of the relevant probabilities

        L($T_m$) = $\frac{p(leaves(T_m)|T_m)}{p(leaves(T_i)|T_i)p(leaves(T_j)|T_j)}$;

    # Perform merge operation m with the greatest probability model ratio $p_M$
    mergeRsubtrees($(T_i, T_j)$, $T_m$, $p_M$, $p_H$);   # automatic context based merging

    # Delete processed rose subtrees $T_i$ and $T_j$
    deleteRsubtrees($(T_i, T_j)$, D);

    # Decrease the counter of clusters
    k ← k - 1;

end

# Return constructed and optimally merged R-TBC Rose Tree
return $T_m$;
```

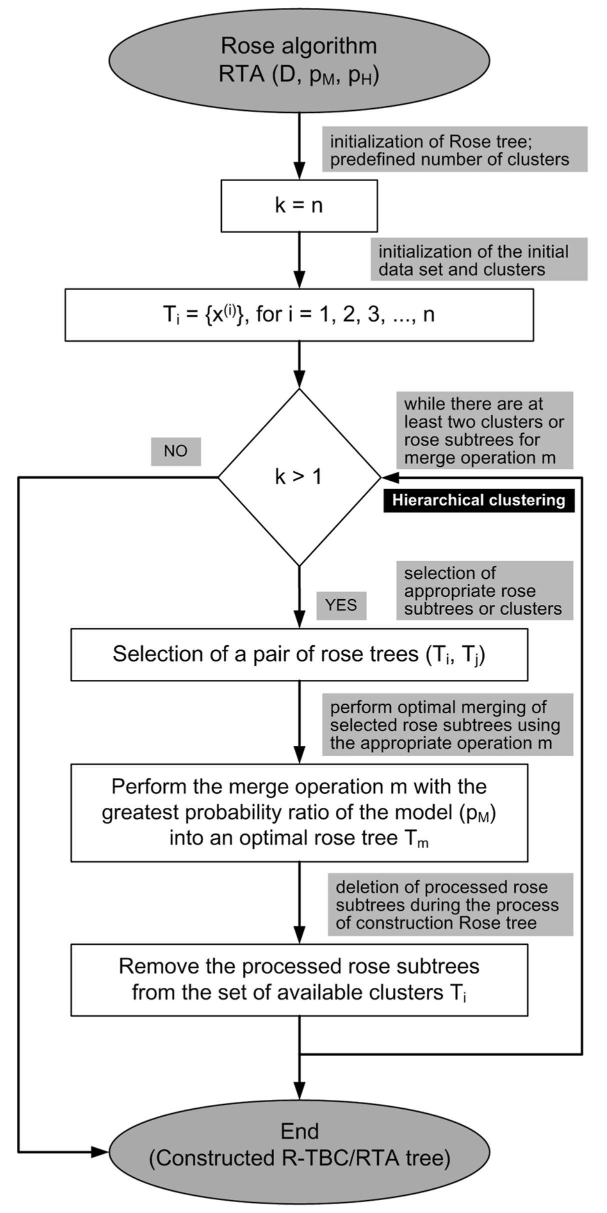

Rose algorithm
RTA (D, p_M, p_H)
initialization of Rose tree; predefined number of clusters
k = n
initialization of the initial data set and clusters
$T_i = \{x^{(i)}\}$, for i = 1, 2, 3, ..., n
while there are at least two clusters or rose subtrees for merge operation m
NO
k > 1
Hierarchical clustering
selection of appropriate rose subtrees or clusters
YES
Selection of a pair of rose trees (T_i, T_j)
perform optimal merging of selected rose subtrees using the appropriate operation m
Perform the merge operation m with the greatest probability ratio of the model (p_M) into an optimal rose tree T_m
deletion of processed rose subtrees during the process of construction Rose tree
Remove the processed rose subtrees from the set of available clusters T_i
End
(Constructed R-TBC/RTA tree)

Fig. 6 Rose Tree Algorithm for construction of the R-TBC consistency tree with replica servers of heterogeneous transactional cloud DBMS environment

Input into the RTA represents clusters or rose subtrees generated as a result of execution of the GPA-MDC intelligent partitioning algorithm on a transactional database. Pseudocode 2. presents general steps of MDC algorithm used within GPA process of intelligent partitioning of background data set of a highly-distributed cloud database and the R-TBC/RTA approach for construction of the final Rose Tree (see Fig. 7).

In the sequence of steps of the GPA-MDC intelligent partitioning algorithm (as it is shown in Pseudocode 2.) for the construction of the MDC multidimensional cube of the R-TBC tree, used parameters are defined as follows:

- $C = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ represents the initial set of data items i.e. nodes of the interconnection hypergraph H of a highly-distributed cloud database,
- k represents the total number of input clusters by the ABC-categorization process with Hybrid partitioning of transactional cloud database,
- $\mathcal{D} = \{d^{(1)}, d^{(2)}, \ldots, d^{(m)}\}$ represents a vector of relevant dimensions or key data attributes of hypergraph elements (initialization phase and dimensioning of CDBMS) with the priority of dimensions in accordance with their position,
- $f(minDT)$ is a goal function of the minimum of distributed DT transactions while preserving the predetermined degree of "visible" apparent consistency AC for transactional CDBMS (input into the main loop of algorithm with MDC),
- MDC_{ij} represents the multidimensional cube with sorted elements of resulting clustering process,
- The main part of the algorithm is a recursive process with the operation of merging or clustering selected clusters C_i and C_j (or $MDC_{i-1,j-1}$) into optimal cluster C_ij (or MDC_{ij}) with the greatest value of goal function f_c,
- The final output or the result of the GPA-MDC intelligent partitioning algorithm is optimally constructed MDC multidimensional cube R-TBC of the rose tree for transactional cloud database.

Therefore, a summary of the main differences between existing, standard TBC and novel, advanced R-TBC/RTA approach, with proposed improvements for managing and preserving the consistency of transactional database within a hybrid cloud environment includes the following:

- exploitation and application of different data structures in the process of constructing the TBC/R-TBC consistency tree,
- approach in modeling the entire heterogeneous cloud environment including the replica servers and other components of the entire cloud environment as well as virtual organizations, with a primary focus on modeling of highly-distributed transactional cloud database into the structure of the R-TBC consistency tree,
- extending the sequence or execution steps of the RTA algorithm for constructing the R-TBC consistency tree, including intelligent partitioning of cloud database with MDC and, finally, optimized and balanced distribution of generated cluster-partitions (CPs) across replica servers of heterogeneous hybrid cloud environment.

```
PSEUDOCODE 2: GPA-MDC( D, $\mathcal{D}$, $f_{minDT}$ )
# Algorithm is performed over data items or nodes of interconnection hypergraph H
# relating a highly-distributed transactional cloud DBMS database
k ← n;  # Initialization of total number of clusters k
$C_i$ ← $\{x^{(i)}\}$, where i=1,2,...,n;
$\mathcal{D}_j$ ← $\{d^{(j)}\}$, where j=1,2,...,m;  # Initialization of data dimensions vector

# Goal function for merge operation as minimum distributed transactions
# with predefined level of apparent consistency for CDBMS data items
$f_c$ ← goalFunc(D, $\mathcal{D}$, minDT, AC);

# Initialization of PEM metrics (price) for merging each pair of clusters
$P_{ij}$ ← ∅, where i=1,2,...,n, j=1,2,...,m;

while k >1 do
  # Calculate the price (minDT) of merging between each pair
  # of elements from selected clusters $C_i$ and $C_j$ (or $MDC_{i-1,j-1}$)
  $P_{ij}$ ← calcPrice($f_c$, ($C_i$, $C_j$)) or calcPrice($f_c$, ($C_i$, $MDC_{i-1,j-1}$));

  # Select the best scored pair of clusters ($C_i$, $C_j$) for merge operation m
  findRsubtrees(D, $P_{ij}$, ($C_i$, $C_j$));

  # Perform merging of selected clusters into optimal cluster $C_{ij}$
  # within the final $MDC_{ij}$ multidimensional cube
  mergeRsubtrees(($C_i$, $C_j$), $C_{ij}$) or
  mergeRsubtrees(($C_i$, $MDC_{i-1,j-1}$), $MDC_{ij}$);

  # Sort elements of resulting cluster into MDC cubes
  # in accordance with predefined dimensions of elements
  $MDC_{ij}$ ← sortMDCelements(D, ($C_{ij}$, $MDC_{ij}$));

  # Delete processed clusters $C_i$ and $C_j$ from initial clusters set C
  # for the construction of the final MDC cube of the R-TBC tree
  deleteRsubtrees(C, ($C_i$, $C_j$));

  # Decrease the counter of clusters
  k ← k - 1;
end

# Return optimally constructed MDC R-TBC Rose Tree cube
return $MDC_{ij}$;
```

The final output or the result of the GPA-MDC intelligent partitioning algorithm is optimally constructed MDC multidimensional cube R-TBC of the Rose tree for transactional cloud database.

Final construction of the R-TBC tree of consistency for transactional database in a hybrid cloud

The process of construction of the R-TBC tree of consistency for transactional cloud database consists of several key phases including following: initialization of data sets, initialization of clusters, background data dimensioning, and multiphased (multidimensional) hierarchical clustering until formation of the final structure of the Rose Tree, as shown on Fig. 8. So, the beginning of the formation of the R-TBC tree starts from the available heap of data of the background cloud database, visually represented by hypergraph H, which is first clustered into groups of related data sets or rose subtrees (rosettes). In addition, through the classification and categorization process, the data included in these rose subtrees (and

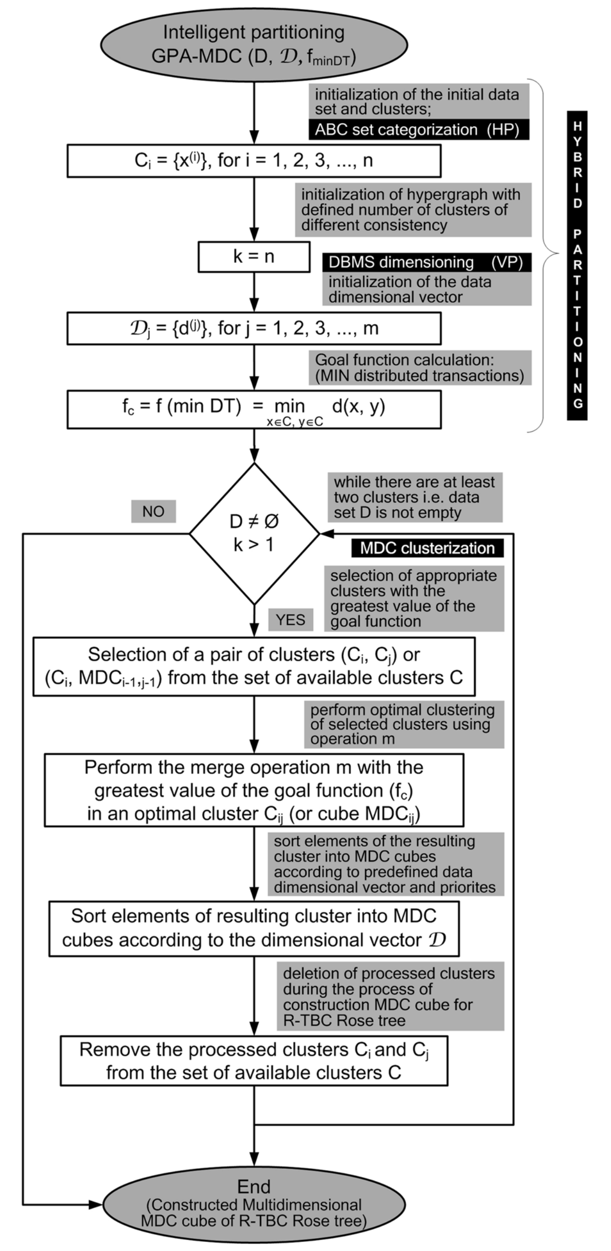

Intelligent partitioning
GPA-MDC (D, $\mathcal{D}$, f_{minDT})
initialization of the initial data set and clusters;
ABC set categorization (HP)
$C_i = \{x^{(i)}\}$, for i = 1, 2, 3, ..., n
initialization of hypergraph with defined number of clusters of different consistency
k = n
DBMS dimensioning (VP)
initialization of the data dimensional vector
$\mathcal{D}_j = \{d^{(j)}\}$, for j = 1, 2, 3, ..., m
Goal function calculation: (MIN distributed transactions)
$f_c = f(\min DT) = \min_{x \in C, y \in C} d(x, y)$
HYBRID PARTITIONING
while there are at least two clusters i.e. data set D is not empty
NO
$D \neq \emptyset$
$k > 1$
MDC clusterization
selection of appropriate clusters with the greatest value of the goal function
YES
Selection of a pair of clusters (C_i, C_j) or (C_i, $MDC_{i-1,j-1}$) from the set of available clusters C
perform optimal clustering of selected clusters using operation m
Perform the merge operation m with the greatest value of the goal function (f_c) in an optimal cluster C_{ij} (or cube MDC_{ij})
sort elements of the resulting cluster into MDC cubes according to predefined data dimensional vector and priorites
Sort elements of resulting cluster into MDC cubes according to the dimensional vector $\mathcal{D}$
deletion of processed clusters during the process of construction MDC cube for R-TBC Rose tree
Remove the processed clusters C_i and C_j from the set of available clusters C
End
(Constructed Multidimensional MDC cube of R-TBC Rose tree)

Fig. 7 GPA-MDC algorithm of intelligent partitioning for transactional database in a heterogeneous cloud environment

corresponding DB tables) are labeled with the appropriate categories of ABC-consistency set [17]. Thus, in the clusters initialization phase, ABC-data categorization takes place producing the clusters of different degrees of guaranteed consistency. This takes into account the fact that not all transactional database data require the same level and guarantee of consistency as they do not have the same importance, volume, scope or price. In this way, the advanced R-TBC model allows the rationalization of the degree of consistency by individual replica servers, that is, the system nodes and other components. Therefore, the adaptive and dynamic rationalization of consistency implies primarily ABC-data analysis, which is carried out by different categories, data type, price and importance (priority) of the background CDBMS data as well as the other relevant factors. This process finishes with setting of flags with different degrees of consistency depending on the results of the performed ABC-analysis and data categorization. These flags refer primarily to different types of consistency (partial and complete), but also different data categories (A, B, C).

Further, the Hybrid partitioning process first performs horizontal partitioning of related clusters into appropriately recognized data ranges, and then vertical partitioning into dimensioning background CDBMS data (see Fig. 8). Thus, strongly linked columns and attributes, or even entire relational tables that exactly match these sets or ranges of related data, group together, and then form higher-order clusters.

With the primary objective of reducing the volume of distributed transactions, the advanced R-TBC approach with the implementation of the RTA algorithm [18] applies an efficient combination of horizontal and vertical partitioning of a transactional database, within an innovative intelligent partitioning technique for cloud DBMS environment. Also, this technique includes proactive organization of data within a back-end transactional database to maximize performance and balance the overall workload of the system and the entire cloud environment.

In the later stages of the RTA process, a multistage MDC of generated clusters is performed through all recognized, relevant data dimensions, taking into account the goal function and respecting the predefined probability model of the Rose Tree. It is important to note that in the process of executing MDC, objects and data items that are highly interconnected are placed in the same cluster (rose subtrees) or joint partition within the cloud network and the background transactional database. These objects can be: replica servers or nodes of the cloud environment and other network and hardware infrastructure, software applications, schemas and tables, and other objects of the background CDBMS, but also virtual organizations (VO), associated organizational structures and units, etc. Thus, in the particular scenario under consideration (this research is focused on scenario for energy companies), everything that constitutes a conceptual hybrid cloud can be part of the R-TBC tree construction and an input for the partitioning process.

Further, based on the dataset requirements of the cloud environment, as well as the connectivity of these objects represented by the appropriate network hypergraph, the objective function is defined, which is to be optimized when performing the partitioning process. In this

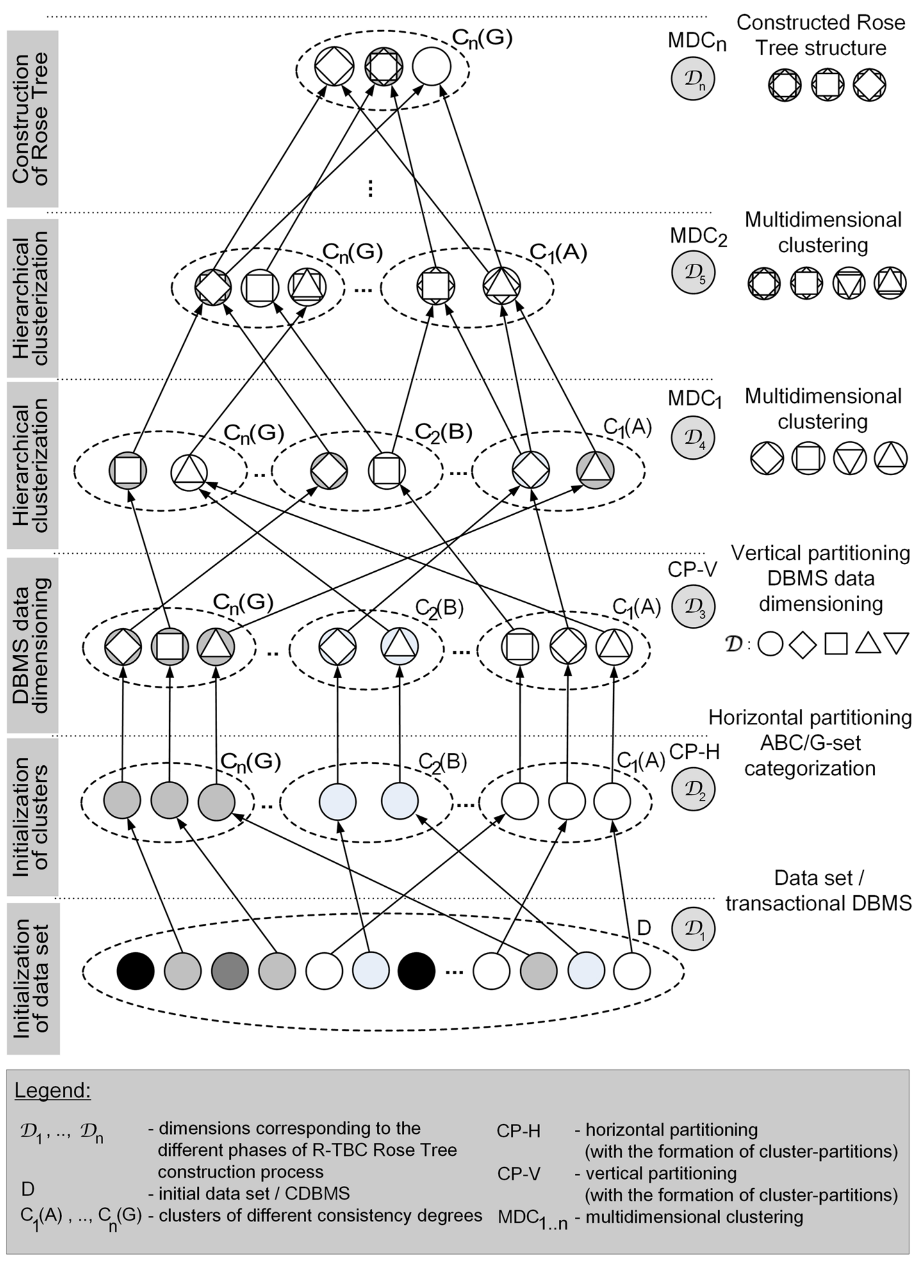

Construction of Rose Tree
Hierarchical clusterization
Hierarchical clusterization
DBMS data dimensioning
Initialization of clusters
Initialization of data set
$C_n(G)$
$C_1(A)$
$C_2(B)$
D
MDCn
Constructed Rose Tree structure
MDC2
Multidimensional clustering
MDC1
Multidimensional clustering
CP-V
Vertical partitioning DBMS data dimensioning
CP-H
Horizontal partitioning ABC/G-set categorization
Data set / transactional DBMS
$\mathcal{D}_1$ … $\mathcal{D}_n$
Legend:
$\mathcal{D}_1, .., \mathcal{D}_n$ - dimensions corresponding to the different phases of R-TBC Rose Tree construction process
D - initial data set / CDBMS
$C_1(A)$, .., $C_n(G)$ - clusters of different consistency degrees
CP-H - horizontal partitioning (with the formation of cluster-partitions)
CP-V - vertical partitioning (with the formation of cluster-partitions)
$MDC_{1..n}$ - multidimensional clustering

Fig. 8 Process of constructing the R-TBC tree of consistency for transactional database in a hybrid cloud environment

particular case, this is a minimization of the number of distributed transactions DT when executing user queries and requests, in order to shorten the response time of the launched DaaS/DBaaS service(s) and increase the overall system performance while maintaining the required adaptive level of consistency of the background CDBMS data items.

Figure 8 shows the whole process of construction of the R-TBC tree of consistency for the transactional cloud database, including all the key phases in the process of executing the GPA-MDC algorithm.

After the RTA algorithm [18] is executed over the initial set of CDBMS data items and recognized ABC-data clusters [17], as the final result of all this processing is formation and establishment of the optimal structure of the Rose Tree for the entire R-TBC hybrid cloud DBMS environment, as shown on Fig. 8.

Upon completion of the main RTA process, the R-TBC cloud environment controller initiates the auxiliary processes for distribution of just (re)constructed Rose Tree with all generated data clusters i.e. intelligent partitions and related DB fragments across entire heterogeneous cloud environment. This actually means initialization (or refreshing) of all included replica servers and network components, leading to the actualization of the final Rose Tree consistency structure for the entire cloud environment. It is important to note that the controller periodically initiates these repartitioning and redistribution processes throughout the structure, especially in cases of heavy system bursts and decrease or significant drop in system performance and launched DaaS/DBaaS services [24, 25]. Therefore, the formed Rose Tree is of the optimal dynamic consistency, properly partitioned, with classified and grouped related data items, appropriately dimensioned, connected and constructed in accordance with the given procedure, and finally supported by the backbone cloud infrastructure and replica servers of the whole environment. Also, the complete tree can be considered as a large MDC multidimensional cube which represents specific infinitesimal structure, customized and optimized for executing Data Manipulation Language (DML) operations and user queries, thus efficiently servicing of input RO/RW transactional workload.

It is important to emphasize that in the particular case and in the considered scenario of optimal organization or arrangement of the cloud environment for companies of energy sector, the application of the RTA hierarchical clustering algorithm [18] based on the R-TBC tree is very suitable as well as reasonable, since it significantly reduces the number of finally generated partitions and clusters (or rose subtrees) on the order of 10^3 and multiple times (relative to other hierarchical structures and algorithms [19]).

Organization of a hybrid cloud environment according to the rose tree model using proposed advanced R-TBC/RTA approach for energy sector companies

It is important to note that proposed advanced R-TBC/RTA approach is implemented in the main control node of the entire cloud environment - controller which manages the whole structure of the rose tree and performs its periodic reconstruction in accordance with the requirements of the cloud environment. This dynamic and adaptive management of the environment, through the control node, allows the formation of a layered and virtual hierarchical structure of the R-TBC tree which consists of replica servers originating from different organizations, in order to create a consolidated working environment (for energy sector companies, in this case scenario) [24]. Thus, the highest layers of the tree structure are replica servers of P2P ring, with the highest PEM performance factors, because they have the highest degree of ownership and responsibility over the partitions of a highly-distributed transactional database. Therefore, the key aspect in preserving the consistency of the conceptualized highly-distributed, hybrid cloud environment is the tree structure of the whole environment with contained virtual units, replicas, and other cloud components [25, 26]. Each virtual organization (VO) refers to a separately formed R-TBC consistency tree, as illustrated on the Fig. 9. By periodical construction and reconstruction of the R-TBC tree, the controller performs creation of a dynamic tree with the partitions or fragments of a distributed transactional database, taking into account the relevant PEM factors or parameters of the network metrics for the entire cloud environment.

In order to achieve high-performance of a hybrid cloud environment, it is necessary to form it having in mind the relevant PEM factors while simultaneously performing the fragmentation or partitioning [26] of the background transactional database. In this way, replica nodes of the environment will be assigned to corresponding responsibilities to the database fragments (or partitions) depending on their degree of importance and position, i.e. priority in the R-TBC Rose Tree of the entire cloud environment. PEM metrics discussed in following section were used to evaluate the replica servers performances as well as other crucial components of the environment [21–23].

PEM metric for managing the structure of the R-TBC tree with replica servers of cloud environment

In order to enable the controller to dynamically manage the R-TBC tree structure, it is necessary, on a regular basis, to perform the calculation of key performance parameters for the entire cloud environment and based on that continuously reconstruct (or restructure) the R-TBC tree. Thus, the R-TBC tree becomes an image or a reflection of the internal state of the cloud environment, in terms of relevant PEM network parameters [21]. PEM metric for highly-distributed environment include the following key factors:

- Workload of replica servers,
- Reliability of replica servers,
- The time required for forwarding messages,
- Network reliability,
- Network bandwidth,
- Network load,
- etc.

After the calculation of the current value of each of these factors, the controller performs an analysis of the impact of PEM factors on the total preservation of consistency for cloud distributed transactional database. Controller performs calculation of the PEM metric, based on n performance factors, according to the following formula:

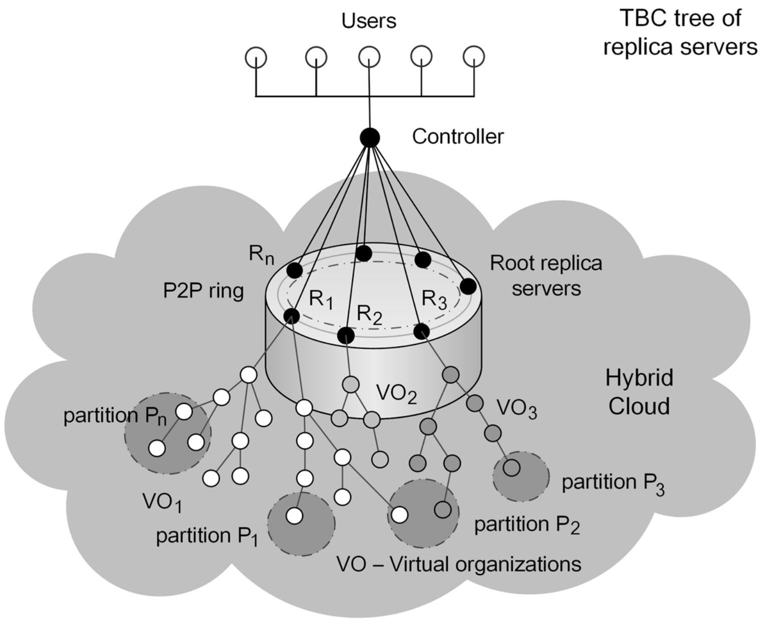


Fig. 9 The R-TBC tree structure of replica nodes for public companies or virtual organizations of the energy sector [21]

$$PEM = \sum_{i=1}^{n} (pf_i * wf_i) \quad (1)$$

, where pf_i is i-th performance factor, and wf_i is i-th weight factor, positive or negative real number.

After calculating the PEM metric and performance evaluation factors [21, 27], the environment controller prepares the R-TBC consistency tree according to the obtained results and parameter values using proposed approach.

As can be seen on the Fig. 10, in the proposed advanced R-TBC/RTA approach, the system is organized as a tree, where the controller manages entire cloud environment and primary replica is the root node. The tree defines a path that is used by the replica servers to propagate the update requests to other replicas (leaves).

The process of constructing the R-TBC consistency tree consists of the following mandatory steps:

(*i*) **Preparing the interconnection graph:** The controller first prepares a weighted interconnection graph G (V, E) where V represents a set of vertices, and E denotes a set of edges with assigned weights (or weight coefficients). Each replica server represents a vertex in the interconnection graph, and is therefore a member of the set V. All direct connections between replica servers are considered as edges, and so are members of the set E. A separate

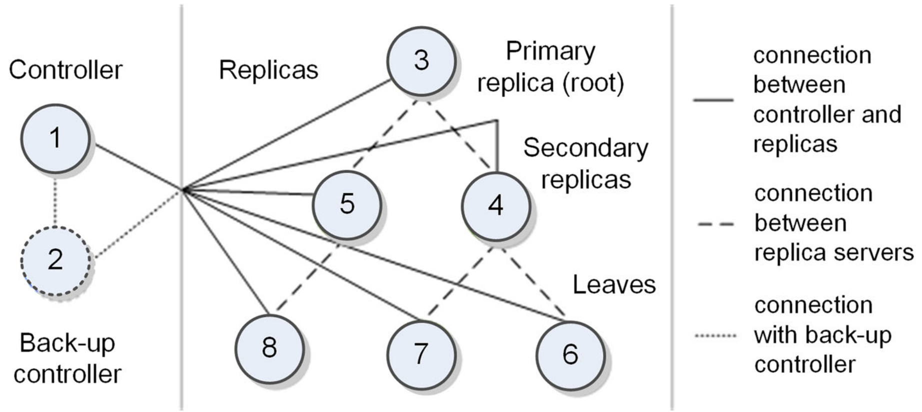
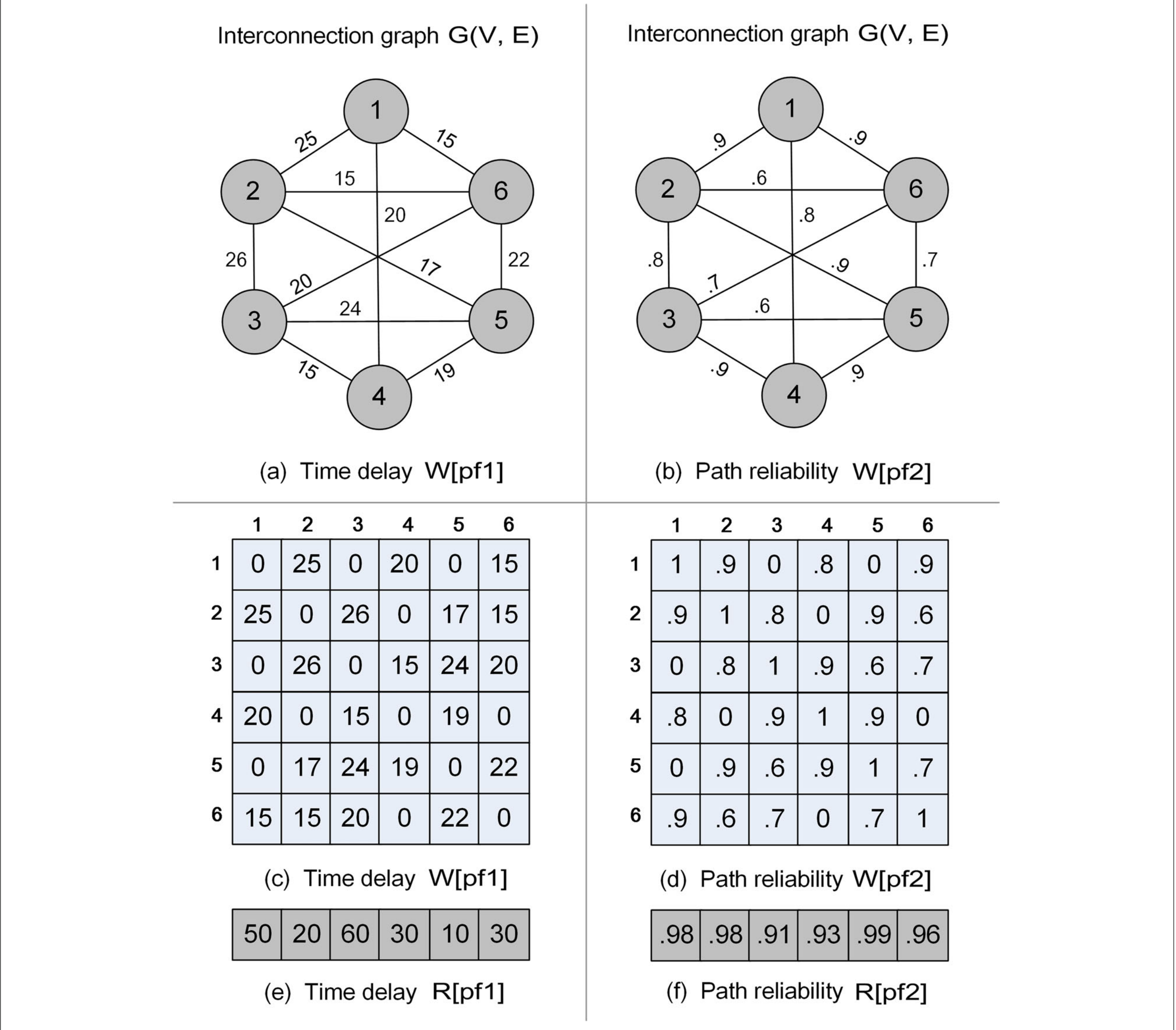

Controller
1
2
Back-up controller
Replicas
3
Primary replica (root)
Secondary replicas
5
4
Leaves
8
7
6
connection between controller and replicas
connection between replica servers
connection with back-up controller

Fig. 10 Communication connections of different components within the R-TBC tree structure [15]

matrix of performance factors for included replica servers is then prepared and calculated for any performance factor considered. It is important to emphasize that the obtained graph G represents so-called weighted graph of interconnections, with weight factors indicated. Figure 11 (*a*) illustrates an example of a graph of interconnections in the process of constructing the R-TBC consistency tree by applying the RTA algorithm [18] for highly-distributed cloud environment.

(*ii*) **Selection of root for the R-TBC consistency tree:** Based on the PEM metric obtained for all replica nodes in the environment, the controller selects the root node of the tree that will be the primary replica server. A primary server is a server that maintains direct connection with the end user. The controller then calculates the PEM value for each replica server based on the performance factors (pf) of the engaged replica servers, combining them together with the corresponding weight factors or coefficients (wf). As a result, the controller selects a root server or replica node that has the maximum PEM evaluation metric value.

Figure 11 (*a*), (*b*) illustrate weighted graphs of network interconnections for various performance factors, with pf1 representing time delay and pf2 path reliability. Figure 11 (*c*), (*d*), (*e*) and (*f*) illustrate the performance factors or adjacency matrices as well as the totals for the same example. Each matrix represents (weighted) graph G, which is a square n x n matrix. Both matrices are sym-


Interconnection graph G(V, E)
1
2
3
4
5
6
25
15
15
20
26
22
20
17
24
15
19
(a) Time delay W[pf1]
Interconnection graph G(V, E)
.9
.9
.6
.8
.8
.7
.7
.9
.6
.9
.9
(b) Path reliability W[pf2]

	1	2	3	4	5	6
1	0	25	0	20	0	15
2	25	0	26	0	17	15
3	0	26	0	15	24	20
4	20	0	15	0	19	0
5	0	17	24	19	0	22
6	15	15	20	0	22	0

(c) Time delay W[pf1]

50	20	60	30	10	30

(e) Time delay R[pf1]

	1	2	3	4	5	6
1	1	.9	0	.8	0	.9
2	.9	1	.8	0	.9	.6
3	0	.8	1	.9	.6	.7
4	.8	0	.9	1	.9	0
5	0	.9	.6	.9	1	.7
6	.9	.6	.7	0	.7	1

(d) Path reliability W[pf2]

.98	.98	.91	.93	.99	.96

(f) Path reliability R[pf2]

Fig. 11 PEM metrics calculation in a process of construction the R-TBC tree of consistency with replica servers of heterogeneous cloud environment [15]

metric. The cost of crossing the path from node Vi to node Vj is denoted as a member (i, j) of the presented matrices. In this case, the 'price' of the matrix specifically refers to network performance factors [21]. Node 5, which has optimal performance (in this case, the least delay on the path and the highest reliability of packet transmission along the same path, as well as other relevant PEM factors) was selected as the root node of the R-TBC consistency tree.

It is important to point out that the value 0 (often the symbol ∞) in the adjacency matrices indicates that there is no connection or link between the two replica servers or nodes in the interconnection graph, or that it is a connection of the node with itself i.e. cyclical bond (which does not exist in the specific case under consideration). As mentioned above, node 5 was selected for the root node since this node or replica server had the least delay (10ms) and the highest path reliability (0,99).

(*iii*) **Preparation of R-TBC consistency tree:** After selecting the root node by the environment controller, preparation of consistency tree from the weighted interconnection graph is started. The controller implements the RTA algorithm [18] for constructing a rose tree, which is an improvement comparing to the standard TBC approach. The root of the tree will then be selected as a single source or initial node of the tree. The RTA algorithm will then find the appropriate (best) path (or interconnection) to each replica server, to maximize system performance, with all paths or interconnections together forming a consistency tree. The algorithm also allows an arbitrary number of children or offspring for the parent node, since the R-TBC tree structure is based on Bayesian Rose Tree which has no limit on the maximum number of children allowed (as it is the case with Binary TBC tree). Further, the advanced RTA algorithm allows the use of negative weight coefficients (which is a difference comparing to the original TBC algorithm). Figure 11 (*c*) and (*d*) illustrate the performance factors used for the same example, where pf1 is a time delay and pf2 is a path reliability. Also, weight factors wf1 = -0,02 and wf2 = 1 were used in this example [21].

It is important to note that the RTA algorithm [18] for construction and optimization of the R-TBC tree shows good results of expected performances for the entire environment and cloud DBMS TBC replication structure. Also, it operates with a relatively small number of parameters, so it is easy to implement. This is because this algorithm is originally designed to manage consistency of a highly-distributed network environments, which is not the case with the other two standard consistency management algorithms (Classic, Quorum). Also, the RTA algorithm does not require significant computing capacities and resources to execute the program code.

This algorithm, as mentioned above, is embedded and implemented in the main node of the entire structure - the controller. For the purposes of processing of a request (or issued query), the user first addresses this node, under whose control a complete further procedure of executing transactions is automatically performed and supervised. Obviously, end user has no any knowledge of the execution processes in the background. Further, the controller has well-defined procedures in the case of some hazard situations like server failures or, even, crashes of parts of the R-TBC tree. It also manages autoscaling processes, i.e. the expansion and contraction of the entire cloud DBMS environment.

Figure 11 shows adjacency matrices for two performance factors and weighted network interconnection graphs in the PEM metrics calculation process for constructing the R-TBC consistency tree using an advanced RTA algorithm for a highly-distributed, cloud environment [18, 21]. As mentioned before, each factor has an appropriate performance weight factor which indicates the importance of this factor in relation to the others as well as its impact on overall system performance. The top layers of tree with replica servers show the best performance which decreases by moving down the tree to the leaves. In this way, the main objective of conducted research is achieved, that is dynamic and adaptive management of consistency as well as other performance factors of a highly-distributed transactional CDBMS environment.

Finally formed and constructed R-TBC consistency tree represents Bayesian Rose Tree, shown on the Fig. 12 (right). On the left side of Fig. 12 is shown an interconnection graph of a distributed system, and on the right is a final structure of the R-TBC consistency tree - node 5 of the initial interconnection graph is selected for the root node of the final R-TBC consistency tree:

It is important to note that the controller as the main component and key node of the entire environment has always an up-to-date copy of the integral cloud database as well as all the necessary information or knowledge of the nodes and other components of entire distributed network, thus having a complete picture of the general state of the heterogeneous cloud database system.

Essentially, PEM factors are assigned as weight coefficients to the hypergraph nodes (as shown on the Fig. 11) and thus participate in the goal function calculation during the phase of partitioning and redistribution of the MDC cluster-partitions (CPs) across replica servers of the entire cloud DBMS environment [21, 28]. The usual timing of the repartitioning and redistribution operation of the CDBMS (or the restructuring of the R-TBC Rose Tree of consistency) is immediately at the beginning of the workday, before starting normal, regular workflows (in this case within energy companies, sharing common

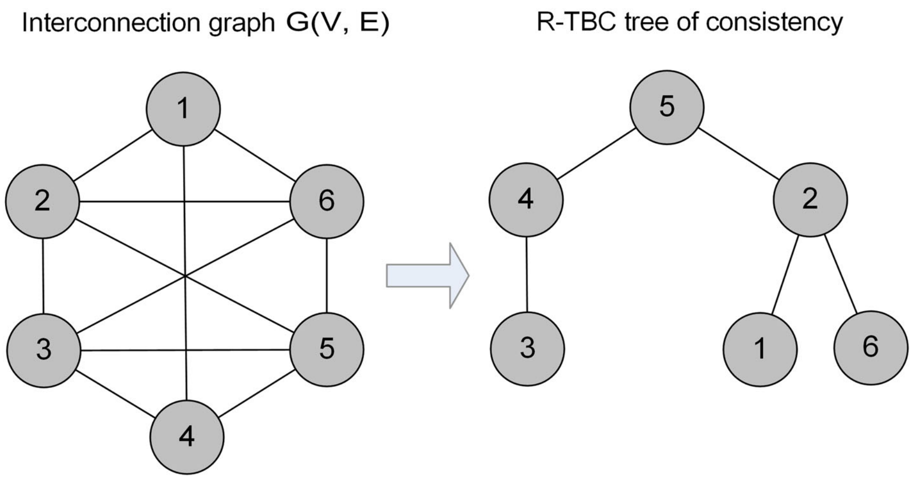


Fig. 12 Initial interconnection graph of highly-distributed system and final R-TBC consistency tree with the application of advanced RTA Rose Tree Algorithm [21]

working cloud DBMS environment, with built-in security options for the SLA service agreements of the cloud service network) [25, 26].

Analysis of experimental results

Experimental setup in this case scenario, relating the proposed advanced R-TBC/RTA approach, included implementation of a simple cluster with six replica servers of standard configuration, connected into cloud configuration as previously shown on Fig. 12 (right). Designed within simulation environment of MathLab framework, CDBMS is based on MySQL Server and highly-distributed transactional database for energy sector companies, resided on Amazon S3 (Simple Storage Service) open platform. MySQL supports several consistency model configurations [11, 14, 15] which is important requirement for this particular experimental setup. In each node i.e. replica server of established cluster configuration are implemented all three conventional approaches (Classic, Quorum, TBC/MTBC) as well as proposed advanced R-TBC/RTA approach for consistency management of a highly-distributed transactional database. A simulation program was developed to generate user requests with transactions sent to the database servers of the cloud DBMS. Default input workload ratio with read/write operations (RO/RW) is 70:30. In this experiment were used other ratios of read/write operations, as well. An arrival of a transaction requests is controlled by a Poisson distribution. The requests with queries contain basic DML SELECT/INSERT operations related to the tables of the common highy-distributed database of energy sector companies. Which tables and columns are going to be used depend on generated user queries and contained read/write opeations. These queries relate to data on energy consumption, user payments and debts, actual price of products and services, discounts, energy package reservations, subscriptions, etc. The goal is to make it easier for the end user to access the necessary data through the constantly available online cloud service and thus increase customer satisfaction with provided services and products. Therefore, a special testing DaaS/DBaaS cloud service is launched on the Java platform and represents a shared user program. Response time is the metric used in conducted experiments. The average time a server spends on processing an operation is calculated by summing the disk and CPU time, and then multiplying this amount by the appropriate percentage of read or write operations.

As already mentioned, six replica servers with a distributed, replicated database and assigned partitions, as well as a configured application server (AS) have been allocated and connected to the main implemented cloud DBMS database on the Amazon S3 platform. On the same servers, DaaS/DBaaS application or test service was launched, which users directly access, depending on the location and availability of the same. Therefore, users and other clients of the DaaS/DBaaS service access the same by using a standard browser and making active connection with AS application and DB replica servers, in the background.

The simulation environment and the conceptual architecture of cloud DBMS for the performed experiment and validation of the results as well as the effectiveness of the proposed advanced R-TBC/RTA approach for preserving the consistency of the environment are shown in the following Fig. 13 which illustrates MathLab simulation environment:

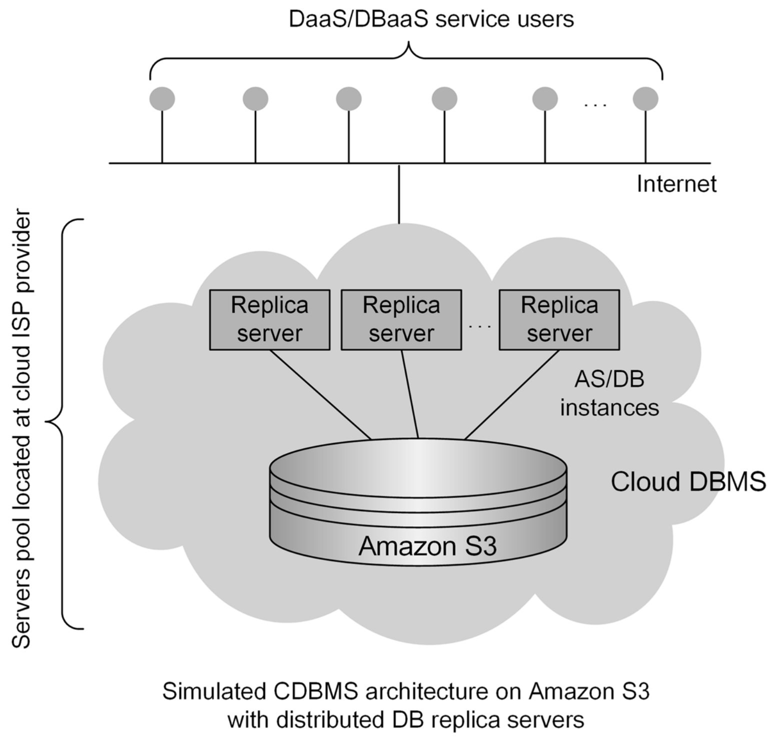

DaaS/DBaaS service users
Internet
Servers pool located at cloud ISP provider
Replica server
Replica server
Replica server
AS/DB instances
Cloud DBMS
Amazon S3
Simulated CDBMS architecture on Amazon S3 with distributed DB replica servers

Fig. 13 Simulation environment and cloud DBMS architecture for the performed experiment and validation of the results of the proposed advanced R-TBC/RTA approach

For the purpose of this experiment, a special simulation program was used to manage the read and write requests sent to the replicated CDBMS database. Each read request is represented by a query (SQL SELECT) against the MySQL database, and each write request is also represented by an insert operation (SQL INSERT) into a background, replicated transactional database. Thousands of user requests are generated by random distribution and then sent to the database to determine the average response time.

The final results for the proposed advanced R-TBC/RTA approach are compared to the results relating conventional approaches (Classical, Quorum, TBC) for managing and preserving the consistency of a highly-distributed transactional database, presented by A.Islam and S.Vrbsky [15]. As experimental results demonstrated the basic TBC approach shows the best overall results in comparison to the other conventional approaches (Classic, Quorum) for maintaining the consistency of the distributed transactional CDBMS, taking into account various aspects and relevant network parameters. As already mentioned, this is primarily because the TBC approach is specifically designed for highly-distributed network environments, which is not the case with the two other approaches. Although it might be assumed otherwise, Quorum approach has shown the worst total performance due to the mandatory participation of all active replica servers (quorum) in each operation of reading and writing data items (RO/RW) of the distributed transactional CDBMS. Also, the Classic approach [10] has not shown satisfactory performance when it comes to the transactional cloud DBMS database. Specifically, in the Classic strategy or approach for maintaining the consistency of a highly-distributed CDBMS environment, one replica node (master or root) is responsible for notifying all other replicas of the system on recent and actual updates. In this way, all replicas must be updated before starting the next read/write operation (RO/RW) within the distributed transactional database. Therefore, according to this approach, each write/update operation (RW) requires the participation of all replica nodes of the environment. Obviously, this increases the response time of the system or DaaS/DBaaS service(s). In addition, only a few direct descendants of the root node i.e. the 'first-level replicas' will result in a reduction of the response time, specifically for the update operations. On the other hand, there will be an increase in workload on the root node, as well as its direct descendants, since all the read/write operations (RO/RW) issued by the end-user using selected DaaS/DBaaS service are directed exactly to these nodes. This is very reasonable taking into account the fact that they are the most potent and updated nodes of the entire cloud environment. Therefore, the Classic approach [10] and related strong consistency management model are recommended for databases that are rarely updated, i.e. proanalytical, because it requires a long time until all replica servers of the entire environment become completely synchronized.

Furthermore, the main disadvantage of the Quorum approach or strategy for maintaining the consistency of DaaS/DBaaS services and system implementations is exactly the application of the eventual EC consistency model [11, 12]. As already mentioned, it represents a variation of the weak model and thus guarantees that the data of the back-end distributed database will only ultimately become consistent. This means that the service database ultimately converges to Strong consistency, but most of the time database items report Weak consistency. However, the Quorum-based and Eventual Consistency models give poor guarantees, and in a large number of data accesses, accuracy and consistency in fact cannot be guaranteed. All of this contributes to the increased generation of conflicts among user requests and queries issued to the highly-distributed cloud database, which is certainly not a satisfactory solution within heterogeneous, highly exploited and multi-user cloud environments. Therefore, the Quorum approach cannot be an adequate solution when it comes to consistency management of highly-distributed database systems, as in this particular case conceptualized and formed hybrid cloud of consolidated transactional database for energy sector companies [21].

The Classic and TBC approaches use a centralized entry point for write requests, while the Quorum approach uses a distributed entry point. As it is known, a centralized entry point could represent the ‘bottleneck’ of the system. Therefore, the calculation of system load for each of the conventional approaches (Classic, Quorum, TBC/MTBC) as well as the proposed advanced R-TBC/RTA approach will be performed in this section on the basis of related mathematical models.

When considering server load, it consists of two parts: load for disk operations and load due to CPU usage. It is assumed that the load due to disk operations actually represents the amount of time it takes to execute and complete the disk operations, while the load due to the use of the processor represents the time that the CPU has to spend to perform a read or write operation.

In the Classic approach, the primary server processes the input load with database write operations, while the secondary servers handle the load with read operations. Obviously, every secondary server must take part in every write operation in order to maintain a Strong consistency of highly-distributed CDBMS data, so the following equations apply [10, 15]:

$$PL = (D_{WP} + C_{WP}) * W \tag{2}$$

$$SL = (D_{WS} + C_{WS}) * W + (D_{RS} + C_{RS}) * R \tag{3}$$

, where D_{WP} is disk write time for primary server, and D_{WS} for secondary server. Analogly, D_{RP} is disk read time for primary server, and D_{RS} for secondary server. Also, C_{WP} is CPU usage time for write operation on primary sever, and C_{WS} is CPU usage time for write operation on secondary server. Similarly, C_{RP} is CPU usage time for read operation on primary server, and C_{RS} is CPU usage time for read operation on secondary server. Furthermore, the primary server load is denoted as PL, while secondary server load is denoted as SL.The number of write operations is denoted as W, while the number of read operations is denoted as R.

In the Quorum approach, servers participate in processing and executing each request with read and write operations on distributed database data items, while each replica server or node can take a role as the primary server with the remaining replica servers serving as secondary servers, so the following equations apply [11, 12, 15]:

$$PL = (D_{WP} + C_{WP}) * W + (D_{RP} + C_{RP}) * R \tag{4}$$

$$SL = (D_{WS} + C_{WS}) * W + (D_{RS} + C_{RS}) * R \tag{5}$$

In the TBC tree based approaches, the root node processes the input load with database write operations, while the direct descendants or immediate children of the root node processes the load with read operations. Each direct descendant of the root must participate in each writing operation to preserve the consistency of the TBC-based system. If we denote the primary or root node load as PL and the load of intermediate children or secondary nodes of the system as SL, then the following equations apply [13–15]:

$$PL = (D_{WP} + C_{WP}) * W \tag{6}$$

$$SL = (D_{WS} + C_{WS}) * W + (D_{RS} + C_{RS}) * R \tag{7}$$

Table 2 shows the measured values as a result of conducted experiments for each of the standard consistency management approaches (Classic, Quorum, TBC), including the primary and secondary replica servers, as well as for the proposed advanced R-TBC/RTA approach, taking into account 70:30 ratio of read and write operations as input workload. On average, for Classic approach, the primary server spends approximately 26 ms (25,8) per operation ((31 + 55) * 0,3), and the secondary servers spends approximately 24 ms (24,1) per operation ((15 + 7) * 0,7 + (21 + 8) * 0,3) = (15,4 + 8,7) = 24,1 ms, respectively. It is concluded that the Classic approach actually performs better when write requests represent a low volume (as it is case in this particular experimental setup). On average, all servers based on Quorum approach consume above 30 ms per operation, much more than Classic approach. Specifically, the Quorum technique is better to write requests in subsequent read or when write operations are high.

Table 2 The processing times (ms) of read/write operations (70/30 R/W ratio) for standard approaches (Classic, Quorum, TBC) and proposed advanced R-TBC/RTA approach for maintaining the consistency of the cloud DBMS

		Classic approach		**Quorum approach**		**TBC approach**		**R-TBC/RTA approach**	
		Read	**Write**	**Read**	**Write**	**Read**	**Write**	**Read**	**Write**
Primary server / root replica	Disc	-	31	16	17	-	16	-	11
	CPU	-	55	23	25	-	30	-	16
Secondary replica servers	Disc	15	21	16	17	18	19	9.3	9.6
	CPU	7	8	16	18	12	11	4	6.1
Average operation processing time(ms)	Primary replica:	-	25.8	27.3	12.6	-	13.8	-	8.1
	Avg (Disc+CPU)	**25.8**		**39.9**		**13.8**		**8.1**	
	Secondary servers:	15.4	8.7	22.4	10.5	21	9	9.3	4.7
	Avg (Disc+CPU)	**24.1**		**32.9**		**30**		**14**	

Finally, the TBC approach performs better in most cases than the previous two approaches (Classic, Quorum) regardless of the input workload. On average, a TBC tree root spends 14 ms (13,8) per operation ((16 + 30) * 0,3), and secondary replica servers spend 30 ms per operation ((18 + 12) * 0,7 + (19 + 11) * 0,3) = (21 + 9) = 30 ms, respectively. Finally, in this case scenario and conducted experiment for the proposed advanced R-TBC/RTA approach, results show that, on average, tree root spends 8 ms (8,1) per operation ((11 + 16) * 0,3), and secondary replica servers spend 14 ms per operation ((9,3 + 4) * 0,7 + (9,6 + 6,1) * 0,3) = (9,3 + 4,7) = 14 ms, respectively. On the basis of obtained data from performed experimental analysis, it can be concluded that Quorum servers spend significantly more time than other considered approaches (Classic, TBC-based). So when the write operations ratio is low compared to read operations (as in the experiment, the R:W ratio is 70:30), then Quorum servers spend more time on average per operation (since every operation that is performed within a Quorum-based system requires a consensus of all quorum members). Also, considering the fact that this analysis takes the average time to process operations, the arrival rate of DB requests has no effect on the complete analysis undertaken.

The results obtained prove that a unique, singular entry point for a write operation (Classic, TBC-based approaches) will not result in a 'bottleneck' scenario in a system with 70:30 ratio of read and write operations, as shown in the Table 2. Also, it can be concluded that the proposed advanced R-TBC/RTA approach shows better overall performances comparing to standard approaches (Classic, Quorum, TBC) taking into account the fact that it expands to entire cloud DBMS and not only to replica servers of the environment.

Also, it is important to note that 70:30 ratio of read and write operations is considered typical ratio for the most data management applications. In the second experiment, the effect of the percentage of read versus write requests on the response time for the standard consistency approaches (Classic, Quorum, TBC), as well as for the proposed advanced R-TBC/RTA approach, is measured for read intense applications. For that purpose, 90:10 ratio of read and write operations is taken into consideration and experimental runs were proceeded on the basis of previously presented mathematical models. A constant arrival rate $\lambda = 0.1$ is maintained for this experiment, meaning there is a 10% chance a request will arrive every 5 ms. Transmission time is taken for regular network, meaning that up to 10% of the packets are delayed due to network congestion.

Table 3 shows the measured values as a result of conducted experiments for each of the standard consistency management approaches (Classic, Quorum, TBC), including the primary and secondary replica servers, as well as for the proposed advanced R-TBC/RTA approach, taking into account 90:10 ratio of read and write operations as input workload. Measured values represent elapsed time in milliseconds. On average, for the Classic approach, the primary server spends approximately 8 ms (8,1) per

Table 3 The processing times (ms) of read/write operations (90/10 R/W ratio) for standard approaches (Classic, Quorum, TBC) and proposed advanced R-TBC/RTA approach for maintaining the consistency of the cloud DBMS

		Classic approach		**Quorum approach**		**TBC approach**		**R-TBC/RTA approach**	
		Read	**Write**	**Read**	**Write**	**Read**	**Write**	**Read**	**Write**
Primary server / root replica	Disc	-	28	17	18	-	12	-	8
	CPU	-	53	25	26	-	25	-	12
Secondary replica servers	Disc	17	18	17	19	17	15	8	7
	CPU	9	5	18	18	11	8	2	3
Average operation processing time(ms)	Primary replica:	-	8.1	28.8	4.4	-	3.7	-	2
	Avg (Disc+CPU)	**8.1**		**33.2**		**3.7**		**2**	
	Secondary servers:	23.4	2.3	31.5	3.7	25.2	2.3	9.9	1
	Avg (Disc+CPU)	**25.7**		**35.2**		**27.5**		**10.9**	

operation ((28 + 53) * 0,1), and the secondary servers spends 25,7 ms per operation ((17 + 9) * 0,9 + (18 + 5) * 0,1) = (23,4 + 2,3) = 25,7 ms, respectively. In this case, the load on the primary server significantly decreased due to the reduced volume of write operations which results in much faster response time. On the other hand, due to increase in load on secondary servers, the response time slightly increased since secondary servers participate in processing each request with read and write operations on cloud DBMS.

On average, for the Quorum approach, the primary server spends approximately 33 ms (33,2) per operation ((17 + 25) * 0,9 + (18 + 26) * 0,1) = (28,8 + 4,4) = 33,2 ms, and the secondary servers spends approximately 35 ms (35,2) per operation ((17 + 18) * 0,9 + (19 + 18) * 0,1) = (31,5 + 3,7) = 35,2 ms, respectively. Generally, it is concluded that the Quorum approach performs worse than the other consistency management approaches because of its higher response time in execution of read-only transactions. Further results show that performance of the TBC approach is better than the standard approaches. On average, a TBC tree root spends 3,7 ms per operation ((12 + 25) * 0,1), and secondary replica servers spend 27,5 ms per operation ((17 + 11) * 0,9 + (15 + 8) * 0,1) = (25,2 + 2,3) = 27,5 ms, respectively. Finally, in this case scenario with 90:10 ratio of read and write operations, performance results for the proposed advanced R-TBC/RTA approach show that, on average, tree root spends 2 ms per operation ((8 + 12) * 0,1), and secondary replica servers spend approximately 11 ms per operation ((8 + 3) * 0,9 + (7 + 3) * 0,1) = (9,9 + 1) = 10,9 ms, respectively. On the basis of obtained data from performed experimental analysis, it can be concluded that Quorum servers spend significantly more time than other considered approaches (Classic, TBC-based). So when the write operations ratio is very low compared to read operations (as in the experiment, the R:W ratio is 90:10), then Quorum servers spend more time on average per operation (due to requirement of the quorum). In this case, proposed advanced R-TBC/RTA approach shows 2-3x better response time comparing to the standard consistency approaches (Classic, Quorum, TBC) due to appropriate load balancing throughout the tree structure of entire cloud DBMS.

In the next experimental setup, 50:50 ratio of read and write operations is used representing write intense applicational workload. In this case, R/W workload is balanced which is especially suitable for TBC-based consistency approaches and satisfactory performance results in response time. Default network parameters were used as in previous experimental runs. All measured values represent elapsed time in milliseconds.

Table 4 shows the measured values as a result of conducted experiments for each of the standard consistency management approaches (Classic, Quorum, TBC), including the primary and secondary replica servers, as well as for the proposed advanced R-TBC/RTA approach, taking into account 50:50 ratio of read and write operations as input workload. On average, for the Classic approach, the primary server spends 49 ms per operation ((36 + 62) * 0,5), and the secondary servers spends

Table 4 The processing times (ms) of read/write operations (50/50 R/W ratio) for standard approaches (Classic, Quorum, TBC) and proposed advanced R-TBC/RTA approach for maintaining the consistency of the cloud DBMS

		Classic approach		**Quorum approach**		**TBC approach**		**R-TBC/RTA approach**	
		Read	**Write**	**Read**	**Write**	**Read**	**Write**	**Read**	**Write**
Primary server / root replica	Disc	-	36	18	19	-	19	-	13
	CPU	-	62	25	27	-	33	-	19
Secondary replica servers	Disc	14	28	18	19	17	22	7.2	13
	CPU	6	13	19	20	11	14	2	10
Average operation processing time(ms)	Primary replica:	-	49	21.5	23	-	26	-	16
	Avg (Disc+CPU)	**49**		**44.5**		**26**		**16**	
	Secondary servers:	10	20.5	31.5	3.7	14	18	4.6	11.5
	Avg (Disc+CPU)	**30.5**		**35.2**		**32**		**16.1**	

30,5 ms per operation ((14 + 6) * 0,5 + (28 +13) * 0,5) = (10 + 20,5) = 30,5 ms, respectively. In this case, it is concluded that the Classic approach performs worse than the other consistency management approaches because of its higher response time in execution of write operations. Also, the Quorum approach does not show satisfactory performance results as the same servers are involved in both the read and write operations. On average, for the Quorum approach, the primary server spends 44,5 ms per operation ((18 + 25) * 0,5 + (19 + 27) * 0,5) = (21,5 + 23) = 44,5 ms, and the secondary servers spends approximately 35 ms (35,2) per operation ((18 + 19) * 0,5 + (19 + 20) * 0,5) = (31,5 + 3,7) = 35,2 ms, respectively. Conversely, the TBC-based approaches show much better results comparing to the standard consistency approaches. Further results show that on average, a TBC tree root spends 26 ms per operation ((19 + 33) * 0,5), and secondary replica servers spend 32 ms per operation ((17 + 11) * 0,5 + (22 + 14) * 0,5) = (14 + 18) = 32 ms, respectively. Finally, in this case scenario with 50:50 ratio of read and write operations, performance results for the proposed advanced R-TBC/RTA approach show that, on average, tree root spends 16 ms per operation ((13 + 19) * 0,5), and secondary replica servers spend approximately 16 ms (16,1) per operation ((7,2 + 2) * 0,5 + (13 + 10) * 0,5) = (4,6 + 11,5) = 16,1 ms, respectively. So, it is obvious that balanced read/write input workload ratio will result in balanced distribution and processing time overall tree structure. In this case, proposed advanced R-TBC/RTA approach shows 2-3x better response time comparing to the standard consistency approaches (Classic, Quorum, TBC) due to appropriate load balancing and dynamic consistency management throughout the entire CDBMS tree structure.

In the next experimental setup, the response time of the DaaS/DBaaS service is measured based on different cluster configurations of input interconnection network graph as well as various density of tree structure. Performance results were carefully observed and measured relating presented cluster configurations, as shown on Fig. 14.

The results obtained prove that a different density of input tree structure for the interconnection graph plays an important role in final performance of standard TBC-based consistency approaches (TBC, MTBC), as well as proposed advanced R-TBC/RTA consistency approach. Different levels of density (high, medium, low) were taken into account while measuring elapsed response time in milliseconds. A typical data management application 70:30 read/write ratio was used in the experiment. As expected, the response time increases as the density of the tree increases [13].

On average, for standard TBC approach, the response time of service ranges from 134 ms for the sparse tree, to 166 ms for the medium trees, to 227 ms for the dense tree, as shown in the Table 5. Cluster configuration consists of 6 nodes connected into heterogeneous hybrid cloud DBMS environment. A sparse tree, as shown on Fig. 14 (*b*), results in a faster response time for the TBC approach, but has fewer immediate children of the root node and fewer updated replicas that are available for access by

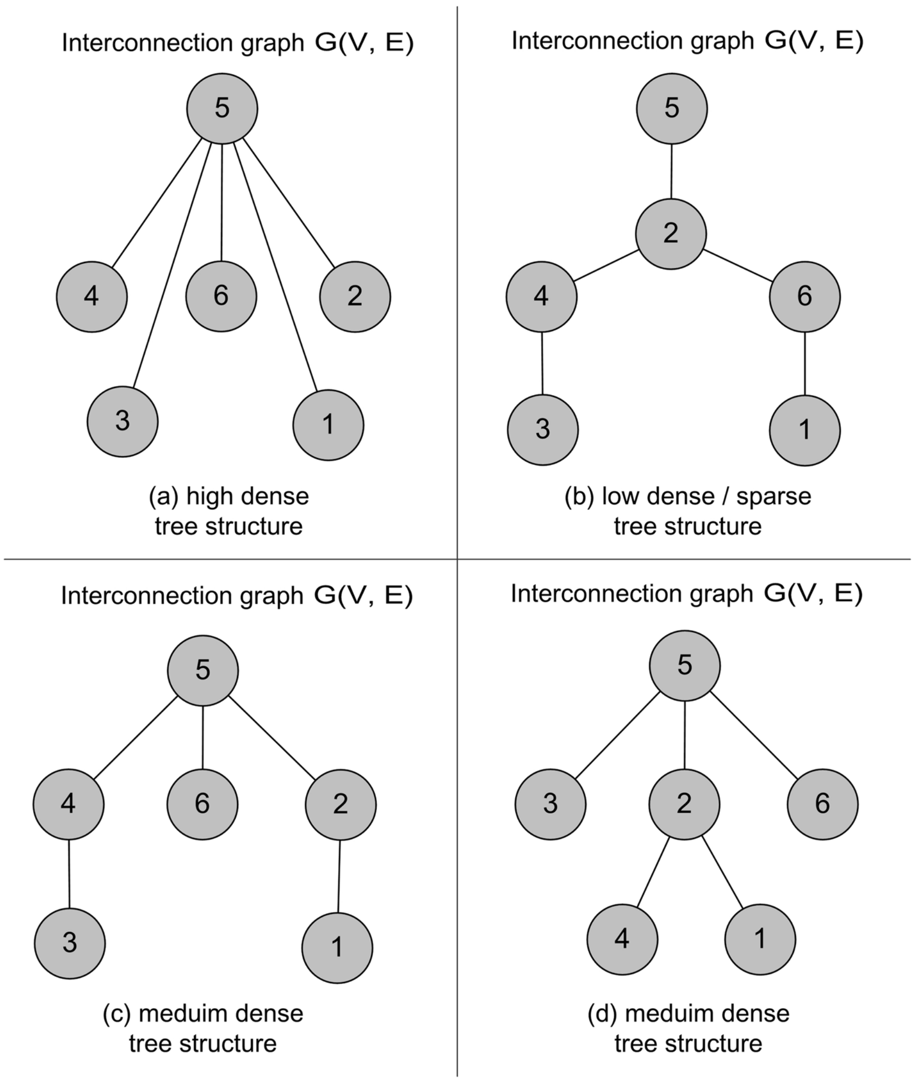

Interconnection graph G(V, E)
5
4
6
2
3
1
(a) high dense tree structure
Interconnection graph G(V, E)
(b) low dense / sparse tree structure
Interconnection graph G(V, E)
(c) meduim dense tree structure
Interconnection graph G(V, E)
(d) meduim dense tree structure

Fig. 14 Effect of tree density on standard TBC/MTBC and proposed advanced R-TBC/RTA consistency approaches in a hybrid cloud DBMS environment

the client. Conversely, a dense tree, as shown on Fig. 14 (*a*), has a slower response time, but more updated replicas available. On average, for the Modified Tree-Based strategy or MTBC approach, the response time of service ranges from 113 ms for the sparse tree, to 145 ms for the medium trees, to 192 ms for the dense tree. Since MTBC approach has a restriction on maximum allowed number of children for parent node, there is a higher rate of occurrence of conflicting operations over CDBMS data items. It is concluded that MTBC approach reduces total amount of response time for DaaS/DBaaS service due to restriction on maximum allowed number of children for parent

Table 5 Average response time (ms) of DaaS/DBaaS service with various density of tree structure using standard 70/30 ratio of read/write operations for standard TBC/MTBC approaches and proposed advanced R-TBC/RTA approach for maintaining the consistency of the cloud DBMS

		TBC approach	MTBC approach	R-TBC/RTA approach
Average response time of DaaS/DBaaS service (ms)	High density tree-structure	**227**	**192**	**144**
	Medium density tree-structure	**166**	**145**	**102**
	Low density tree-structure	**134**	**113**	**76**

node, but it increases the chances of possible conflicts and increases the complexity of addressing those conflicts. This is specifically the case for high and medium density tree structures, as shown on Fig. 14 (*a*), (*c*) and (*d*). Finally, on average, for the proposed advanced R-TBC/RTA consistency approach, the response time of service ranges from 82 ms for the sparse tree, to 104 ms for the medium trees, to 149 ms for the dense tree, as shown in the Table 5. From obtained experimental results, it is concluded that proposed advanced R-TBC/RTA approach provides a flexible framework that allows fast adaptation to different cluster configurations of input interconnection network graph as well as various density of tree structure. Thus it provides dynamic trading off the availability and consistency while preserving optimal levels of the performance and QoS of DaaS/DBaaS service delivered to the end user.

Finally, in the next conducted experiments, large number of issued transactions were processed, and specifically the response time of the DaaS/DBaaS service as well as the Inconsistency Window IW were carefully observed and measured (in milliseconds). The first experiment measuring read and write times show that the proposed advanced R-TBC/RTA approach represents a significant improvement comparing to standard approaches. Surprisingly, the Quorum approach did not performed as expected. According to Table 2, while the existing TBC approach shows 2x-3x faster processing time compared to the Classic and Quorum approaches, the proposed R-TBC/RTA approach improves the TBC approach for approximately 45%-55%, both on primary and secondary replica servers (in average weighted time).

Regarding the TBC approach and its modification MTBC approach, obtained average results from repeated experiments relating actual Inconsistency Window are shown on the Fig. 15. It is important to emphasize that the Inconsistency Window IW [16] is the key PEM performance factor that is in the focus of the overall analysis of results when comparing standard vs. advanced consistency management strategies. As already mentioned, the MTBC strategy [13, 14] as well as the proposed advanced R-TBC/RTA strategy are designed precisely with the goal to significantly reduce the Inconsistency Window as a key PEM factor of the entire distributed CDBMS system. Thus, the response time of the system was precisely monitored and measured in order to demonstrate the overall effects on system induced by decreasing the length of the Inconsistency Window IW (in milliseconds).

As shown on Fig. 15 the Classic approach has the Inconsistency Window of approximately 270 ms (268), while the Quorum approach has IW factor with the length of 380 ms, much longer than Classic (due to requirement for quorum replica set). On the other side, the TBC approach has very short Inconsistency Window of 144 ms, while the MTBC approach has IW factor with the length of 109 ms. The experiments demonstrate that the Modified TBC approach MTBC can noticeably reduce the length of the system IW Inconsistency Window, but on the other side significantly increase the chances for generating and

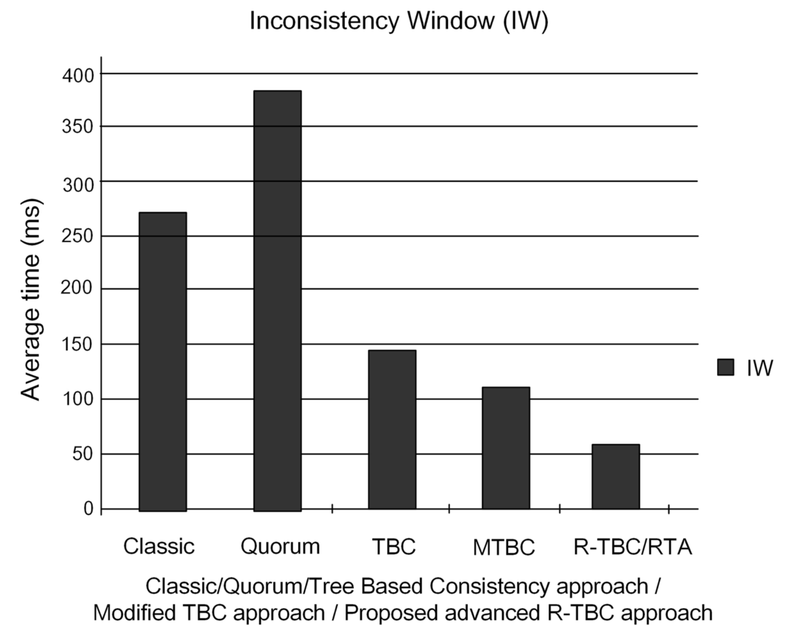


Fig. 15 Comparison of standard approaches (Classic, Quorum, TBC/MTBC) and proposed advanced R-TBC/RTA approach relating Inconsistency Window IW

causing conflicts over data items and different replicas of the environment, as well as the complexity of these conflict situations and their efficient resolutions. On contrary, in this case scenario and conducted experiment relating proposed advanced R-TBC/RTA approach, results in prove that this novel approach prevents occurrence of conflicts among data items of cloud transactional database as well as different replica servers of the entire cloud environment. Also, having IW factor with the length of approximately 60 ms (58), advanced R-TBC/RTA shows improved performances for about 55% comparing to Modified MTBC approach and for about 42% comparing to standard TBC approach. This reduction of IW parameter using novel R-TBC/RTA approach is significant due to minimized interdependency of replica servers within R-TBC consistency tree. Based on the obtained results from experimental analyzes, it is confirmed the high customization and the complete justification of using the proposed advanced R-TBC/RTA approach to maintain optimal levels of consistency for data items of a highly-distributed transactional cloud DBMS database, with application for energy sector companies cooperating within common heterogeneous cloud [21]. Also, this improvement in performances is achieved due to the implementation of intelligent partitioning of a highly-distributed transactional CDBMS database, and then appropriate distribution of data partitions across replica servers within entire cloud environment.

Relating failure recovery, the proposed advanced R-TBC/RTA approach has developed procedures for managing network failures of certain components of the cloud highly-distributed network. Managing failures ultimately comes down to the reconstruction of the R-TBC consistency tree. So if any replica server or communication link fails within the established consistency tree, then the responsibility for handling such a problematic situation is taken over by the environmental controller [28]. As it is already known, the controller maintains continuous communication with the primary (root) replica of the R-TBC consistency tree but also with the other replica servers of the environment. The controller's task is to keep the R-TBC consistency tree valid and communication links active. Also, it is responsible for managing the synchronization processes of the transactional cloud DBMS. If the primary (root) replica fails and the same server cannot be restored in the short term, then the controller starts the selection of the new root node for the R-TBC consistency tree, based on the regularly calculated PEM metric for replica servers of the environment. The server with the highest PEM performance metric is selected for the new root node of the consistency tree. Then, the controller establishes the network connections of the R-TBC tree and thus completely reconstructs it. Also, the interconnection graph is reconfigured with all available replica servers, and thus successfully built using described procedure. However, if some other replica servers or communication links fail within the R-TBC consistency tree, then the controller first tries to communicate with the same server or revive the network link. If this non-responsive behavior continues, then the controller starts the procedure of removing the same server and/or deactivating the communication link from the existing R-TBC consistency tree and immediately begins the process of reconstructing it. The connection graph is then reconfigured without that particular server or communication link and all servers are informed about the new tree structure. In the event the communication link is down, the controller can still communicate with the server via another link. The controller will then reconfigure the connection graph including that server and build the consistency tree.

From the aforementioned, it can be concluded that the network quality, i.e. the loss of packages and traffic congestion does not have a significant impact on the efficiency and performance of the proposed advanced R-TBC/RTA approach. Thus the R-TBC tree structural approach shows much better performances for managing network failures compared to the standard approaches of consistency management of transactional cloud DBMS (Classic, Quorum). Network quality has a very large impact on the response time of the Classic approach due to regular synchronization procedures of the entire cloud environment. Also, Quorum approach does not show satisfactory performances in managing network failures [28] due to the requirements on the quorum server set. Coversely, the proposed advanced R-TBC/RTA approach reduces the network and transactional failure risk due to reduced interdependency among replica servers and achieves high performances regardless of network load, bandwidth and other performance factors.

Furthermore, the comparison of standard TBC/MTBC approach and the proposed advanced R-TBC/RTA approach performed in continuation of the experiment, focused primarily on basic data structures exploited within these approaches, i.e. Binary BHC and R-TBC Rose Tree, also confirms these findings and conclusions. As it is already mentioned, adopting the Bayesian tree structure based on the BRT Bayesian Rose Tree model of hierarchical clustering [20], with an arbitrary number of node children into the TBC tree structure, leads to the formation of an optimized structures with much greater likelihood of representation of the analyzed data sets. Experiments show that Bayesian hierarchical structure is efficiently optimized, better distributed in the search space, with better utilization of memory and other computing resources and capabilities, and with a much smaller resultant set of generated partitions of the final Rose Tree. Also, the resultant partition sets in the BHC and BRT models significantly differ in size thus contributing to the BRT model preference over the BHC Binary model. This leads to a very clear and obvious conclusion regarding overall performance, applicability and primarily efficiency of the two leading models of hierarchical clustering, the Binary BHC and Bayesian BRT Rose Tree model. Consequently, the results demonstrated limitation in application of the standard TBC/MTBC approach based on BHC Binary Tree model when compared to the proposed advanced R-TBC/RTA Rose Tree model and novel consistency management approach. While TBC/MTBC data structure or Binary TBC tree of consistency is very often forced and illusory, the RTA data structure - Rose Tree of consistency - is natural and easily found in various analyzed data sets. Also, TBC/MTBC is more specific and limited to replica servers, while R-TBC/RTA is designed for general application within entire highly-distributed transactional cloud DBMS environments. When observing the execution time of R-TBC/RTA algorithm, it shows an exponential $O(n^k \log n)$ complexity referring the entire cloud environment, where k is the input set of clusters or rose subtrees (rosettes), and n is a total number of data items from initial data set D. On the other side, TBC/MTBC Dijkstra algorithm executes with $O(mn \log n)$ complexity, where m is a number of relevant performance factors, and n is a total number of replica servers in distributed system. Considering the fact that standard TBC/MTBC approach is limited to replica servers while advanced R-TBC/RTA approach relates to the entire cloud environment, these complexities contribute to the preference of the proposed advanced R-TBC/RTA consistency approach for management of highly-distributed cloud DBMS environments [21].

Obtained experimental results show that the proposed advanced R-TBC/RTA approach and "visible" consistency model, successfully handle input workload, distribute and balance it across entire cloud replica servers and thus achieve high performance levels and quality of service QoS while maintaining dynamic data consistency. This adapted quality of data QoD is achieved by relaxing and enforcing the consistency level on datasets as needed and in accordance with real requirements of cloud service users. Finally, all mentioned facts confirm that the proposed advanced R-TBC/RTA approach represent the reasonable choice for consistency management of a highly-distributed transactional CDBMS database and optimal model of hierarchical clustering for various data structures and components within heterogeneous cloud environment.

Discussion

By performing experiments, presented in the previous section, and on the basis of obtained results, the work hypothesis is successfully confirmed. Conclusions provide general picture related to the performance of the examined leading approaches for managing the consistency of highly-distributed transactional cloud DBMS environment, as well as leading algorithms for hierarchical clustering but also the efficiency of generated data structures (trees) using the appropriate models. Comparisons were made between standard, conventional consistency management strategies (Classic, Quorum, TBC) and the advanced TBC-based strategies (MTBC, R-TBC/RTA), along with comparing appropriate hierarchical clustering models, i.e. BHC Bayesian Binary and BRT Bayesian Rose Tree model [20], in relation to their representation in the analyzed data sets, degree of efficiency and optimization as well as the scope of their application. The obtained experimental results point to the fact that the effects of the Inconsistency Window IW on distributed system with the application of the MTBC approach are minimized [14, 15]. The reason for this is significantly reduced the interdependency of involved replica servers within the highly-distributed transactional CDBMS environment. On the other hand, as a consequence of this improvement, there is an evident increase in generating conflicts over data items and different cloud components as replica servers owning assigned fragments or CDBMS partitions. At the same time, this means increased complexity in resolving the same conflict situations and consequently searching for some more effective solutions. Therefore, opportunities have been explored to further improve the existing standard TBC and Modified TBC (MTBC) approach, in the form of the proposed advanced R-TBC/RTA approach, based on the optimized hierarchical structure of the Rose Tree of consistency for the transactional cloud DBMS environment. By analyzing the hierarchical clustering models (Bayesian Binary BHC and Bayesian Rose BRT), summarized findings support the conclusion that the BRT model of hierarchical clustering [18, 19] shows significantly better results and finds much better quality hierarchical structures (i.e. rose trees) than any other BHC algorithm (or its variations). This is because the latter are based on the illusory and forced data structure of Binary Tree which very often does not depict the actual condition of the analyzed data sets and primarily their internal structure with nodes and connections of all included data items. Therefore, based on the undertaken research can be concluded that the BRT model and Rose Tree [20] are proven to represent optimal clustering and hierarchical structure model for most application and modeling of background data sets within the highly-distributed transactional CDBMS environment. This is because they support inherent, the actual and already contained, existing data structures in analyzed data sets.

Finally, the application of innovative RTA/R-TBC Rose Tree Algorithm in the construction of CDBMS consistency tree, based on the BHC Bayesian Hierarchical clustering, shows significantly improved overall performances compared to conventional approaches (Classic, Quorum, TBC/MTBC). Generally, the proposed advanced RTA/R-TBC approach demonstrates a number of advantages over the existing, conventional approaches for maintaining the consistency of a transactional database within a hybrid cloud environment. While the standard TBC approach constructs minimum spanning tree of replica nodes by running a Modified Dijkstra's algorithm, the RTA/R-TBC approach builds a multi-way tree by using a clustering algorithm. The research carried out in this paper suggests that, using the proposed advanced R-TBC/RTA approach to preserve and maintain the targeted levels of dynamic (adaptive) consistency of the established hybrid transactional cloud DBMS environment, the necessary and sufficient consistency guarantees of the analyzed data sets are achieved. Thus, the ultimate goal of undertaken research is successfully reached by proving the hypothesis that Eventual Consistency [11, 12] model does not adequately satisfy the needs of end users of the cloud environment for the accurate and consistent data of the highly-distributed transactional CDBMS database, but rather novel adaptive consistency model AC Apparent Consistency.

Conclusion

The standard consistency management strategies for a highly-distributed transactional CDBMS environment have not demonstrated sufficient performances in all applications. Most of them are based on the eventual consistency model, which does not provide satisfactory levels of data consistency. Aim of this research was to explore the possibilities for improving them and developing new strategies for consistency management and preservation, with application in highly-distributed transactional cloud DBMS environments. Structural TBC Tree Based Consistency approach showed significantly better performance compared to the other considered standard approaches (Classic, Quorum). It represents the basis for further exploration and improvements in this domain. All the weaknesses and disadvantages of the TBC approach were thoroughly analyzed, resulting in the conclusion that the binary structure of the finally formed TBC consistency tree is not adequate for use in most problem situations and does not adequately model realistic background data sets. Thus, the application of the Bayesian hierarchical structure of the Rose Tree has been introduced, and the proposed advanced R-TBC/RTA approach is practically based on it. Also, the basic TBC approach algorithm (Modified Dijkstra Shortest Path MD SP) has been replaced by the innovative RTA and GPA-MDC algorithms for the construction of the Rose Tree for entire hybrid cloud environment together with intelligent partitioning of a highly-distributed transactional CDBMS database. The CDBMS fragments or partitions are controlled and distributed across the R-TBC tree organized cloud environment, and that way achieving optimal performances and values of the relevant parameters of PEM network metric. Finally, the ultimate goal of conducted research - a dynamic and adaptive management of the desired degree of data consistency for the transactional cloud database is achieved, that way promoting an advanced model of so called "visible" or Apparent Consistency as a necessary and sufficient degree of synchronization of all replicas of entire heterogeneous cloud environment.

Acknowledgements
There is no acknowledgement.

Authors' contributions
All authors read and approved the final manuscript. The author J.Dizdarevic proved the presentation of the research (proposed approach, made the implementation and testing), under the supervision of other co-authors. The contribution of the author F.Orucevic is the formulation of problem, analysis and presentation of the situation in the field, in collaboration with the first author. The contribution of the author Z.Avdagic is the structuring of the manuscript and the way of presenting the results and analysis, in cooperation with the first author. The contribution of the author S.Omanovic is related to helping the first author to present the proposed approach using diagrams and pseudocodes.

Authors' information
Jasmina Dizdarevic is a leading engineer for the information system, in Development sector Department of Information system and cadastre, KJKP Sarajevogas Ltd. Sarajevo. Actively took participation in implementation of a number of projects, programs and systems. Jasmina Dizdarevic received the Bachelor of Engineering degree in Computer Science in 2005 from University of Sarajevo/Faculty of Electrical Engineering, and the MS degree in Computer Science in 2013 from the University of Sarajevo. She is currently a Ph.D. student in Computer Science at the University of Sarajevo, Bosnia and Herzegovina. Her research interests include cloud computing, service management, highly-distributed databases and software engineering. Zikrija Avdagic is Bosnian scientist in the field of artificial intelligence and real-time systems. He was awarded a Fulbright scholarship to study in the field of genetic algorithms. He is a professor at the Faculty of Electrical Engineering, University of Sarajevo. His research interests include genetic algorithms, computer systems in real time, and methods and the application of artificial intelligence. Fahrudin Orucevic is a professor at the Faculty of Electrical Engineering, University of Sarajevo. Actively took participation in implementation of a number of projects, programs and systems. His research interests include cloud-based systems, service management and computer systems in real time. Samir Omanovic is a professor at the Faculty of Electrical Engineering, University of Sarajevo. Actively took participation in implementation of a number of projects, programs and systems. His research interests include software engineering, pattern recognition, image processing, genetic algorithms and artificial intelligence. University of Sarajevo, Faculty of Electrical Engineering, Computer Science and Informatics Department, Zmaja od Bosne bb, 71000, Sarajevo, Bosnia and Herzegovina.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 9 June 2020 Accepted: 19 January 2021
Published online: 16 April 2021

References

1. Agrawal D, El Abbadi A, Chin Ooi B, Das S, Elmore AJ (2012) The evolving landscape of data management in the cloud. Int J Comput Sci Eng 7(1):2–16
2. DeWitt DJ, Gray J (1992) Parallel database systems: The future of high performance database systems. Commun ACM 35(6):85–98
3. Abadi DJ (2009) Data management in the cloud: Limitations and opportunities. IEEE Data Eng Bull 32(1):3–12
4. Roe C (2012) ACID vs. BASE: The Shifting pH of Database Transaction Processing. http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing. Accessed 21 Nov 2020
5. Brewer E (2012) Cap twelve years later: How the 'rules' have changed. Computer 45(2):23–29
6. Bonomi S (2012) The CAP Theorem and the Design of Large Scale Distributed Systems: Part I. University of Rome La Sapienza, Great Ideas in Computer Science & Engineering, Rome
7. Thomas RH (1979) A majority consensus approach to concurrency control for multiple copy databases. ACM Trans Database Syst 4(2):180–209
8. Bernstein PA, Goodman N (1981) Concurrency control in distributed database systems. ACM Comput Surv 13(2):185–221
9. Steinke RC, Nutt GJ (2004) A unified theory of shared memory consistency. J ACM (JACM) 51(5):800–849
10. Miret LP (2014) Consistency models in modern distributed systems. an approach to eventual consistency. Masters thesis. PhD thesis, Universitat Politecnica de Valencia, Valencia, Spain
11. Jimnez-Peris R, Martnez MP, Alonso G, Bettina K (2003) Are quorums an alternative for data replication?. ACM Trans Database Syst (TODS) 28(3):257–294
12. Vogels W (2009) Eventually consistent. Commun ACM 52(1):40–44
13. Islam MA (2013) Database consistency in cloud databases. PhD thesis, The University of Alabama, Tuscaloosa, Alabama
14. Islam MA, Vrbsky SV, Hoque MA (2012) Performance Analysis of a Tree-Based Consistency Approach for Cloud Databases. International Conference on Computing, Networking and Communications, ICNC'12. ICNC. pp 39–44
15. Islam MA, Vrbsky SV (2010) Tree-Based Consistency Approach for Cloud Databases. In: IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), Indianapolis
16. Klems M, Bermbach D, Weinert R (2012) A Runtime Quality Measurement Framework for Cloud Database Service Systems. In: Proceedings of the 2012 Eighth International Conference on the Quality of Information and Communications Technology (QUATIC '12). IEEE Computer Society, Washington, DC. pp 38–46
17. Karthick M, Karthikeyan S, Pravin MC (2014) Glob J Res Eng (G) Ind Eng 14(2):1–3
18. Karthick M, Karthikeyan S, Pravin MC (2014) A Model for Managing and Controlling the Inventory of Stores Items based on ABC Analysis. Glob J Res Eng (G) Ind Eng 14(2):1–3
19. Roy DM, Kemp C, Mansinghka V, Tenenbaum JB (2007) Learning annotated hierarchies from relational data. Adv Neural Inf Process Syst 19:1185–1192
20. Blundell C, Heller KA (2010) Bayesian rose trees. In: Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS). Department of Statistics, University of Oxford, London. pp 1601–1609
21. Rimal BP, Jukan A, Katsaros D, Goeleven Y (2011) Architectural Requirements For Cloud Computing Systems: An Enterprise Cloud Approach. J Grid Comput 9(1):3–26
22. Joarder MK, Manzur M, Rajkumar B (2014) Workload-Aware Incremental Repartitioning of Shared-Nothing Distributed Databases for Scalable Cloud Applications. In: IEEE/ACM 7th International Conference on Utility and Cloud Computing, Amsterdam. pp 213–222
23. Rimma NA, Nicolas B (2011) Automated partitioning design in parallel database systems. In: The ACM SIGMOD International Conference on Management of Data, Athens. pp 1137–1148
24. Rimal BP, Jukan A, Katsaros D, Goeleven Y (2011) Architectural requirements for cloud computing systems: An enterprise cloud approach. J Grid Comput 9(1):3–26
25. Ciciani B, Didona D, Di Sanzo P, Palmieri R, Peluso S, Quaglia F, Romano P (2012) Automated Workload Characterization in Cloud-based Transactional Data Grids. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW). IEEE Computer Society, Washington, DC. pp 1525–1533
26. Curino C, Jones EPC, Madden S, Balakrishnan H (2011) Workload-aware database monitoring and consolidation. In: SIGMOD, Conference. ACM SIGMOD, Athens. pp 313–324
27. Kraska T, Hentschel M, Alonso G, Kossmann D (2011) Consistency Rationing in the Cloud: Pay only when it matters. Systems Group, Department of Computer Science, ETH Zurich. In: Proceedings of the VLDB Endowment, vol 2, issue 1. PVLDB, Seattle. pp 253–264
28. Louis-Rodríguez MJ, Navarro J, Arrieta-Salinas I, Azqueta-Alzúaz A, Sancho-Asensio A, Armendáriz-Iñigo JE (2013) Workload management for dynamic partitioning schemes in replicated databases. In: The 3rd International Conference on Cloud Computing and Services Science (CLOSER). CLOSER, Aachen

Publisher's Note