IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

BVAGQ-AR for Fragmented Database Replication Management

A.Noraziah[1,2], Ainul Azila[3], Sharifah Hafizah Sy Ahmad Ubaidillah[1], Basem Alkazemi[4], Julius Beneoluchi Odili[5]

[1] Faculty of Computing, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia.
[2] Centre for Software Development & Integrated Computing, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia.
[3] Faculty of Computer and Mathematical Sciences, Universiti Teknologi Mara Kelantan, 18500 Machang, Kelantan, Malaysia.
[4] College of Computer and Information Systems, Umm Al-Qura University, Makkah, Saudi Arabia
[5] Departments of Mathematical Sciences, Anchor University Lagos, Lagos, Nigeria

Corresponding author: noraziah@ump.edu.my

ABSTRACT Large amounts of data have been produced at a rapid rate since the invention of computers. This condition is the key motivation for up-to-date and forthcoming research frontiers. Replication is one of the mechanisms for managing data, since it improves data accessibility and reliability in the distributed database environment. In recent years, the amount of various data grows rapidly with widely available low-cost technology. Although we have been packed with data, we still have lacked of knowledge. Nevertheless, if the impractical data is used in database replication, this will cause waste of data storage and the time taken for a replication process will be delayed. This paper proposes Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR) algorithm in order to handle fragmented database synchronous replication. BVAGQ-AR algorithm is capable for partitioning the database into disjoint fragments. Fragmentation in distributed database is very useful in terms of usage, reliability and efficiency. Managing fragmented database replication becomes a concern for the administrator because the distributed database is disseminated into split replica partitions. The result from the experiment shows that handling fragmented database synchronous replication through proposed BVAGQ-AR algorithm able to preserve data consistency in distributed environment.

INDEX TERMS Replication, algorithm, fragmentation, data mining, computational intelligence, distributed databases, data grid

I. INTRODUCTION

Large amounts of data have been produced at a rapid rate since the invention of computers. This condition is the key motivation for up-to-date and forthcoming research frontiers. Nowadays, huge numbers of data are generated around the world distributed across data grid. One of the biggest problems that data grids users have to overcome today is to improve the management of data. Providing reliable services along with high data availability and the performance are the important requirements that need to be essentially met. The concept of replication is used to ensure these requirements. The main idea of replication is to manage large volumes of data in a distributed manner, speeds up data access, reduces access latency and increases data availability [1, 2]. In addition, fragmentation replication is designed to enhance the data availability and the system performance of the distributed database for data management [3].

Distributed database replication is a very challenging platform especially when dealing with a huge data. However, in recent years, with widely available, low-cost technology, the amount of various data grows rapidly. The problem is although we are packed with data, but we still lacked of knowledge. Nevertheless, if the impractical data is used in database replication, this will cause waste of data storage and the time taken for a replication process will be delayed. In Distributed Indexing Dispatched Alignment (DIDA), when there are too many requests and/or huge targets, the arrangement process becomes computationally challenging [4]. However, this research not focusing on query updates processing. The BSCA strategies [5] applied association rules in its replication strategies. Association Rules is used to find the correlations between the data. This method will improve the average response time for the transactions. However, data replication will only be done during the collecting components process. Hence, this method does not apply synchronous replication method. In

Prefetching-Based Replication Algorithm (PRA), when a local site obtains a file request but the file is not stored locally, it will search other site to transfer the required file replica through the Replica Directory Server [6]. The local site will select some adjacent files to start the replication process. However, the sequence databases need some storage space. This is because as the time goes on, the size of the databases will become larger. Hierarchical Replication Scheme (HRS) consists of a root database server and one or more database servers organized into a hierarchy topology [7]. Once the changes have been made, all the data will be replicated into the entire replicas. In order to maintain consistency among the updates by clients, all blocks are propagated and locked during the transaction process. This means only one client can modify the data at a time. Branch Replication Scheme (BRS) is composed of a different set of sub-replicas organized using a hierarchical topology [7]. In order to maintain consistency among the updates by clients, a mechanism is proposed. Clients only can modify the data located in the terminal replica, or referred as the leaf nodes of the replication tree. A problem may occur in BRS when a client tries to write in a sub-replica which is not terminal, because that sub-replica has been split into other sub replica. For replication techniques namely Read-One-Write-All (ROWA), they copy all data to all sites which means all servers will have the same data [8, 9]. Data reliability and availability is confirmed but the issues are the data redundancy will be high, it will waste the storage space and the processing time for a transaction also will be high because it has to commit the transaction at all servers.

Although data availability is better because data are stored at more than one site, most of existing replication strategies neglects the correlation between the data files in a Distributed Database Systems (DDS). The information about the data correlation can be dig out from past data using techniques from data mining field. Data mining technique is a part of data clustering method [10]. It is a powerful tool for assisting the extraction of meaningful data from large data sets [11, 12, 14, 15]. The objective for mining grid data is analyzing grid systems with data mining techniques in order to find new meaningful knowledge. The information later can be used to improve grid systems in numerous fields. However, only a small number of works have applied data mining techniques to discover file correlations in data grids [13]. Therefore, the study on this basis is initiated.

In our previous work, the Binary Vote Assignment on Grid (BVAG) has been proposed in order to increase write query availability with low communication cost through the small replication quorum [21]. However, the paper not considering the data fragmentation design, which is more suitable for distributed database environment. Thus, this paper proposes Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR) algorithm in order to handle fragmented database synchronous replication. BVAGQ-AR algorithm is capable for partitioning the database into disjoint fragments.

This paper is organized as the following. The nature of data mining in grid is explained in Section 2. Section 3 presents the BVAGQ-AR technique for data management. Section 4 elaborate experimental results in distributed environment. Finally, Section 5 and 6 discuss and conclude our research finding from this article.

.

II. DATA MINING IN GRID

One of the data mining techniques is called Association rules. The rules are created by analyzing data for frequent if/then patterns and using the criteria support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database. In addition, Association rules are also able to discover a set of items that appear frequently together in a transaction by using Apriori algorithm. This data set is called a frequent item set.

The basic concepts of data mining association rules are called support and confidence. These concepts showed the practicality and certainty in data discovery rules.

Rule 1: $A \Rightarrow B$ set up in transaction D, it has support s, where P is percent of $A \cup B$ in transaction D, it is the $P\ (A \cup B)$ where A and B are item sets which $A \neq B$. So support is defined as:

$$support\ (A \Rightarrow B) = P\ (A \cup B) \quad (1)$$

Each discovery mode should be denoted by a certainty measure of its efficiency or reliability, so rule 2 is:

Rule 2: $A \Rightarrow B$ has confidence c, it is percent both A and B in transaction D. It is conditional probability $P\ (A \mid B)$, so the certainty measure confidence is defined as:

$$confidence\ (A \Rightarrow B) = P\ (A \mid B) \quad (2)$$

If rule 1 and rule 2 meet the specified minimum support and confidence, that the rules for strong association rules.

Rule 3: it is strong association rule, if $support \geq \min support$ and $confidence \geq \min confidence$. The min support is minimum support, and min confidence is minimum confidence.

An algorithm namely Apriori is proposed for mining frequent item sets for Boolean association rules [16]. The name of the algorithm is established on the fact that the algorithm uses prior knowledge of frequent item set properties, which will be explained later. Apriori is an iterative method known as a level-wise search, where $k - item$ sets are used to explore $(k + 1) - item$ sets.

First, the set of frequent 1-itemsets is discovered by scanning the database to determine the count for each item, and assembling those items that satisfy the minimum support. The resulting set is represented as L_1. After that, L_1 is used to

identify the set of frequent 2-itemsets, L_2, which later is used to identify L_3, and so on, until no more frequent k-item sets can be discovered. The process of discovering each of the L_k involves one full scan of the database.

An important property called the Apriori property is used to reduce the search space in order to improve the efficiency of the level-wise generation of frequent item sets,

Apriori property: *All nonempty subsets of a frequent item set must also be frequent.*

The Apriori property is based on the following observation. By definition, if an item set, I does not satisfy the minimum support threshold, *min sup*, then I is not frequent, that is, $P(1) < \min sup$. If an item A is added to the item set I, then the resulting item set (i.e., $I \cup A$) cannot occur more frequently than I. Therefore, $I \cup A$ is not frequent either, that is, $P(I \cup A) < min\ sup$.

III. BVAGQ-AR TECHNIQUE

The main idea of replication is to create multiple copies of the same data or replicas in several storage resources. However, while focusing in replication, there are some methods that neglect the correlation among different data files. Actually, in many applications, data files may be correlated in terms of accesses and have to be considered together in order to reduce the access cost [17]. Indeed, the analysis of data usage in several real data grids such as Dzero [18] and Coadd [19] revealed the existence of strong correlations between files, i.e., jobs tend to request a set of correlated files. This paper proposes Binary Vote Assignment on Grid Quorum with Association Rule (BVAGQ-AR) technique. In BVAGQ-AR, all sites are logically organized in the form of a two-dimensional grid structure. For example, if BVAGQ-AR consists of twenty-five sites, it will be logically organized in the form of 5 x 5 grid. There are four phases involves in BAVGQ-AR framework, which are:

1. Data mining – Apriori algorithm from Association Rules
2. Database fragmentation
3. Database allocation
4. Database replication

Figure 1 shows the BVAGQ-AR framework.

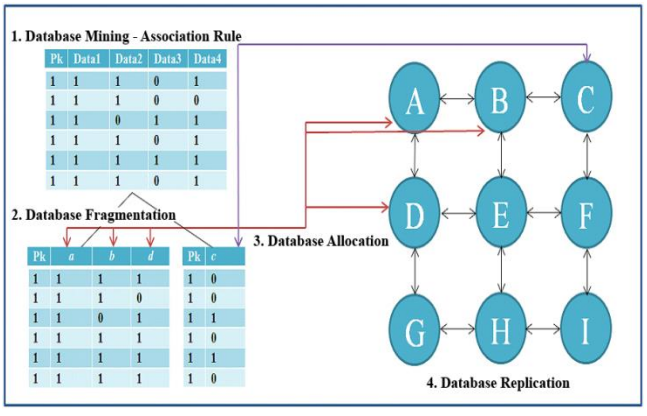

FIGURE 1. BVAGQ-AR framework

1. Data mining – Apriori algorithm from Association Rules

Data mining technique that has been deployed in this experiment called association rules. It is used to discover the correlation between data. Apriori algorithm is an algorithm for frequent item set mining and association rules learning over transactional databases. Learning association rules basically means finding the items that are appeared together more frequently than the others.

2. Database fragmentation

This method also has been proposed to make sure data replication can be effectively done while minimize storage. In general, applications work with some relations rather than entire relations. Therefore, for data distribution, it is better to work with subsets of relation as the unit of distribution. Thus, not all data will be replicated to all sites. The data is fragmented based on data mining analysis results.

3. Database allocation

All sites are logically organized in the form of two-dimensional grid structure. For example, if BVAGQ-AR consists of twenty-five sites, it will logically organize in the form of 3 × 3 grids. Each site has database relation files. The databases that are produced after database fragmentation process are allocated at their assigned sites.

4. Database replication

After database allocation process, each site has a database relation file. A site is either operational or failed and the state (operational or failed) of each site is statistically independent to the others. A copy at a site is available when the site is operational; otherwise it is unavailable [20,21].

A. BVAGQ-AR ALGORITHM DEFINITION

In this section, BVAGQ-AR is proposed by considering the distributed database fragmentation. The following notations are defined:

i. S is a relation in database.
ii. S' is relation after mining
iii. s is the instance in S or S'
iv. J_1 is the frequent item sets
v. J_2 is not the frequent item sets
vi. $S(B)^1$ is the four sites in the corners
vii. $S(B)^2$ is the other sites on the boundaries
viii. $S(B)^3$ is the middle sites
ix. V is a transaction.
x. T is a tuple in J_1.
xi. x is an instant in T which will be modified by element of V.
xii. y is an instant in T which will not be modified by element of V.
xiii. S_1 is a vertical fragmented relation with instant x derived from J_1.

xiv. S_2 is a vertical fragmented relation without instant x derived from J_1.
xv. Pk is a primary key.
xvi. Pk, x is a primary key with data x.
xvii. Pk, y is a primary with data y, where $y \neq x$
xviii. $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$ are a horizontal fragmentation relation derived from J_1.
xix. η and ψ are groups for the transaction V.
xx. $\lambda = \eta \; or \; \psi$ where it represents different transaction V (before and until get quorum).
xxi. V_η is a set of transactions that comes before V_ψ, while V_ψ is a set of transactions that comes after V_η.
xxii. D is a union of all data objects managed by all transactions V of BVAGQ-AR.
xxiii. Target set $= \{1,0\}$ is a result of transaction V.
xxiv. BVAGQ-AR transaction element V_λ is an element either in different set of transactions V_η or V_ψ.
xxv. wV_λ is write counter for the transaction.
xxvi. $\hat{V}_{\lambda_x}$ is a transaction that is transformed from V_{λ_x}
xxvii. V_{μ_x} represents the transaction feedback from a neighbour site. V_{μ_x} exists if either V_{λ_x} or $\hat{V}_{\lambda_x}$ exists.
xxviii. Successful transaction at primary site $V_{\lambda_x} = 0$ where $V_{\lambda_x} \epsilon D$ (i.e., the transaction locked an instant x at primary). Meanwhile, successful transaction at neighbour site $V(\mu_x) = 0$, where $\mu_x \epsilon D$ (i.e., the transaction locked a data x at neighbour).
xxix. $\lceil \frac{n}{2} \rceil$ is the greatest integer function (i.e., $n = 9, \lceil \frac{9}{2} \rceil = 5$.

This model starts with inserting database S. Then, S is mined into S'. From S', the data is fragmented into J_1 and J_2. If J_1 is less than or equivalent to three, then the data will be allocated at $S(B)^1$ because it has three replication servers. If the J_1 is equivalent to four, the data will be allocated at $S(B)^2$ because it has four replication servers. If J_1 is more than or equivalent to five, then the data will be allocated at $S(B)^3$ because it has five replication servers. After all data are replicated to their specific servers, the replication process can be executed.

The primary replica for a particular instant x is a replica that accepts the client's request. In BVAGQ-AR model, each replica of $S(B)$ can be a primary or a neighbour replica at the same time. Any replica $i \epsilon S(B)$ can be chosen as the primary replica, while other replicas $j \epsilon S(B)$ where $i \neq j$ are neighbours. When a transaction V_η request an instant x from any replica of $S(B)$, that replica will be the primary, while others will be the neighbour replica for processing V_η. At the same time, if other sets of transactions invoke to update x after V_η, these set of transactions are called V_ψ. When V_ψ obtain lock from instant x from any site of $S(B)$, which is a different site of the primary replica for processing V_η, that site becomes the primary processing for V_ψ. Simultaneously, the primary processing for V_ψ also functions as neighbour replica for processing V_η and vice versa. Other sites of $S(B)$ that is neither primary replica for processing V_η nor primary replicas for processing V_ψ will function as neighbour replicas for processing V_{λ_x}, where $\lambda = \eta, \psi$.

$S(B)$ is the set of replicas with replicated copies are stored corresponding to the assignment B for particular instant x,

$$S(B_x) = \begin{Bmatrix} m(i,j), m(i-1,j), m(i,j-1), \\ m(i,j+1), m(i+1,j) \end{Bmatrix}.$$

Two sets of transactions, V_η request instant x from $m(i,j)$ replica, while V_ψ request instant x from $m(i-1,j)$ respectively. The $m(i,j)$ replica functions as the primary replica for processing V_η, where $m(i-1,j), m(i,j-1), m(i,j+1), m(i+1,j)$ are neighbour replicas for processing $V_{\gamma_x} \epsilon V_\eta$. Simultaneously, $m(i-1,j)$ replica functions as the primary replica for processing V_ψ, while $m(i,j-1), m(i,j+1), m(i+1,j)$ and $m(i,j)$ are neighbour replicas for processing $V_{\gamma_x} \epsilon V_\psi$. Both $m(i,j)$ and $m(i-1,j)$ replicas execute two different processing task concurrently. The $m(i,j)$ replica is the primary replica processing V_η and neighbour replica processing for V_ψ, whereas the $m(i-1,j)$ replica is the primary replica for processing V_ψ and neighbour replica for processing V_ψ. BVAGQ-AR model considers different sets of transactions V_η and V_ψ. V_η is a set of transactions that comes before V_ψ, while V_ψ is a set of transactions that comes after V_η. The effect of BVAGQ-AR transaction is defined as the processing of one instance of the transaction.

One site has a preliminary database, S, which will be converted into binary format. Each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise.

For example, a database with binary variable is shown in Table I. W and Z represent the items in the database and n is the total number of transactions.

Support, s, is the fraction of transactions that contain both W and Z where

$$s = \sigma(a,b,c,d)/n = 7/20 = 0.35 \; @ \; 35\% \qquad (3)$$

Confidence, c, measures how often items in Z appear in transactions that contain W.

$$c = \sigma(a,b,c,d)/\sigma(a,b) = 7/10 = 0.7 \; @ \; 70\% \qquad (4)$$

TABLE I
DATABASE WITH BINARY VARIABLE

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*I*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*
1	1	1	1	0	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	1	1	0
1	1	1	1	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	1
1	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	0	1	0	0	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0
1	0	0	0	1	0	1	0	1	0	1	1	1	1	0	1	0	0	0	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1
1	1	0	1	1	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1

For simplicity, data from row 1 to 5 and column 1 to 6 in Table 2 is used for this example case. Figure 2 shows an illustration of the frequent item set generation in the Apriori algorithm for the transactions. It is assumed that the support threshold is 60%, which is equivalent to a minimum support count equal to three because in this example, the items have to appear more than half of the transactions to be taken as a frequent item sets. In large databases, if the threshold is 40% or below, all the data most likely will appear together.

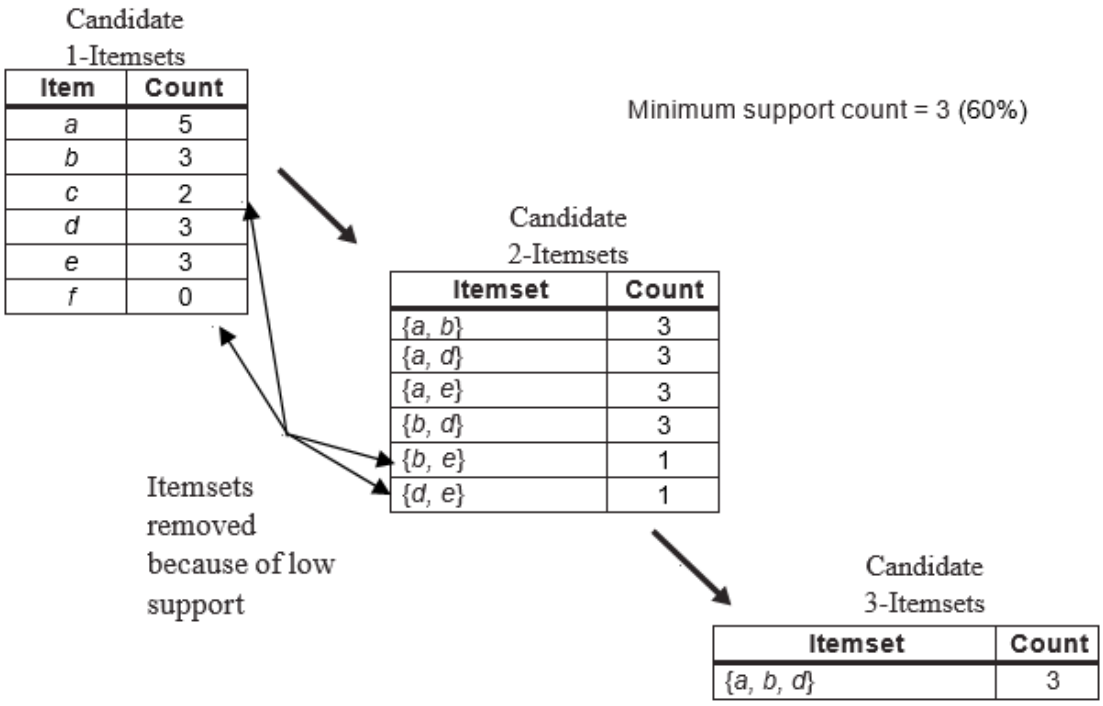

FIGURE 2. **Generating frequent item sets using the Apriori algorithm**

Initially, every item is considered as a candidate 1-itemset. After counting their supports, the candidate item sets $\{c\}$ and $\{f\}$ are discarded because they appear in fewer than three transactions. In the next iteration, candidate 2-itemsets are generated using only the frequent 1-itemsets because the Apriori algorithm ensures that all supersets of the infrequent 1-itemsets must be infrequent. Because there are only four frequent 1-itemsets, the number of candidate 2-itemsets generated by the algorithm is $(24) = 6$.

Two of these six candidates, $\{b, e\}$ and $\{d, e\}$, are subsequently found to be infrequent after computing their support values. The remaining four candidates are frequent, and thus will be used to generate candidate 3-itemsets. Without support-based pruning, there are $(36) = 20$ candidate 3-itemsets that can be formed using the six items given in this example. With the Apriori algorithm, only candidate 3-itemsets whose subsets are frequent will be kept. The only candidate that has this property is $\{a, b, d\}$.

The relation that is resulted from identifying the frequent item sets, S' will be fragmented into relation with frequent item sets, J_1 and relation without frequent item sets, J_2 using vertical fragmentation. When S' is fragmented, it is divided into a number of fragments $S'_1, S'_2, \ldots . S'_n$.

$$S' = S'_1 \cup S'_2 \cup \ldots . \cup S'_n \quad (5)$$

The fragmentation should be done in such a way that relation S can be reconstructed from the fragments:

$$\boldsymbol{S' = S'_1 \bowtie S'_2 \bowtie \cdots . \bowtie S'_n} \quad (6)$$

It is necessary to include the primary key or some candidate key attribute in every vertical fragment so that the full relation can be reconstructed from the fragments. After fragmentation, J_1 is allocated at its replica sites, $S(B)^1, S(B)^2$ *or* $S(B)^3$.

Each site now has a primary data file which is either operational or failed, and the state (operational or failed) of each site is statistically independent to the others. When a site is operational, the copy at the site is available; otherwise it is unavailable.

Recall the Binary Vote Assignment on Grid (BVAG) technique [13]. However, BVAG only covers the voting and a part of the replication process.

Definition 1: A site X is a neighbour to site Y, if X is logically located adjacent to Y.

A data will replicate to the neighboring sites from its primary site. The number of data replication, d, can be calculated using Property 1, as described below.

Property 1: The number of data replication from each site, $d \leq 5$.

Proof: Let n be a set of all sites that are logically organized in a two-dimensional grid structure form. Then

n sites are labelled $m(i,j), 1 \leq i \leq \sqrt{n}, 1 \leq j \leq \sqrt{n}$. Two way links will connect sites $m(i,j)$ with its four neighbours, sites $m(i \pm 1, j)$ and $m\,(i, j \pm 1)$, as long as there are sites in the grid. Note that, four sites on the corners of the grid have only two adjacent sites, and other sites on the boundaries have only three neighbours. Thus, the number of neighbours of each site is less than or equal to 4. Since the data will be replicated to neighbours, then the possible number of data replication from each site, d, is:

$$d \leq the\ number\ of\ neighbours + a\ data\ from\ itself \leq 4 + 1 = 5$$

IV. EXPERIMENTAL RESULTS

In this section, the experiments for managing transaction and replication are described. To demonstrate BVAGQ-AR transaction, 9 servers that logically organized in 3 × 3 are considered based on BVAGQ-AR two-dimensional logical design. 9 servers have been used because the number of replicated data, d, can be 3, 4 or 5. Hence, 9 servers are chosen in order to get maximum replicated data, $d = 5$ in the experiment. The 5 replication servers have been deployed as in Figure 3. Each server or node is connected to one another through a fast Ethernet switch hub. Theoretically, each of the neighbour replication servers and the primary replication server should be connected each other logically as shown in Figure 2. Each server has been assigned with vote 0 or 1. Vote 0 means the server is free locked and able to proceed with a new transaction. In contrast, vote 1 means the server is busy which means it is already locked. Hence, new transaction cannot be initiated on that server.

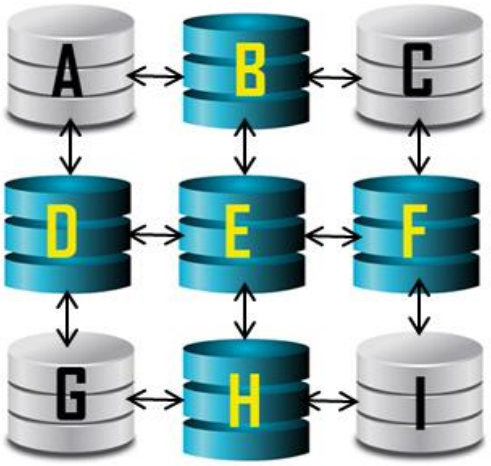

FIGURE 3. Five replication servers connected to each other

The Binary Vote Grid Coordination depicted in Table 2. Replica B with IP 172.21.202.163, replica D with IP 172.21.202.162, replica E with IP 172.21.202.169, replica F with IP 172.21.202.168 and replica H with IP 172.21.202.2167 locate instant e.

TABLE II
BVAGQ-AR GRID COORDINATION

Primary	Neighbours			
B: 172.21.202.163	D: 172.21.202.162	E: 172.21.202.169	F: 172.21.202.168	H: 172.21.202.167
D: 172.21.202.162	E: 172.21.202.169	F: 172.21.202.168	H: 172.21.202.167	B: 172.21.202.163
E: 172.21.202.169	F: 172.21.202.168	H: 172.21.202.167	B: 172.21.202.163	D: 172.21.202.162
F: 172.21.202.168	H: 172.21.202.167	B: 172.21.202.163	D: 172.21.202.162	E: 172.21.202.169
H: 172.21.202.167	B: 172.21.202.163	D: 172.21.202.162	E: 172.21.202.169	F: 172.21.202.168

In this experiment, a transaction, V_η requests to update instant *e* at site E. The aim of this experiment is to record the job execution time for the replication process. The result for this experiment is presented in Table 4.

From the result from Table 4, at time equivalent to 1 $(t1)$, instant e at all servers are unlocked. At $(t2)$, the transaction begins. At $(t3)$, there is a transaction, V_{η_e} request to update instant e at server E. The transaction initiates lock. Hence, write counter for server E now is equal to 1. At $(t4)$, V_{η_e} propagate lock at its neighbour replica B at server B, V_{η_e}lock (e) from E. Thus at $(t6)$, the transaction achieved in getting locked from the B then write quorum is equal to 2. Next, V_{η_e} propagates lock at server D at $(t7)$ and at $(t8)$, V_{η_e}lock (e) from E. Thus at $(t9)$, the transaction achieved in getting locked from the D then write quorum is equivalent to 3. After that, V_{η_e} propagate lock at server F at $(t10$)and at $(t11)$, V_{η_e}lock (e) from F. Thus, at $(t12)$, the transaction achieved in getting locked from the F then write quorum is equivalent to 4. Then, V_{η_e}propagate lock at server H at $(t13)$ and at $(t14)$, V_{η_e}lock $((e))$ from H. Thus at $(t15)$, the transaction achieved in getting locked from the H then write quorum is equal to 5. At $(t16)$, V_{η_e}obtain all quorums and then instant *e* is updated at $(t17.)$ At $(t18)$, the relation *S* is fragmented into S_1 and S_2 using vertical fragmentation. At $(t19)$, the relation S_1 is fragmented again using horizontal fragmentation into $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$. Finally, at $(t20)$, $\hat{V}_{\lambda_e} \in V_\eta$ is commit and at $(t21)$, instant e at all replica servers will unlock and ready for the next transaction to take place.

TABLE III
EXPERIMENTAL RESULT FOR ONE TRANSACTION AT ONE SITE

REPLICA/ TIME TAKEN (ms)	E	B	D	F	H
t1	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)
t2	begin_transaction	begin_transaction	begin_transaction	begin_transaction	begin_transaction
t3	V_{η_e} write lock(e), counter_w(e)=1				
t4	V_{η_e} propagate lock: B				
t5		V_{η_e} lock(e) from E			
t6	V_{η_e} get lock: B, counter_w(e)=2				
t7	V_{η_e} propagate lock: D				
t8			V_{η_e} lock(e) from E		
t9	V_{η_e} get lock: D, counter_w(e)=3				
t10	V_{η_e} propagate lock: F				
t11				V_{η_e} lock(e) from E	
t12	V_{η_e} get lock: F, counter_w(e)=4				
t13	V_{η_e} propagate lock: H				
t14					V_{η_e} lock(e) from E
t15	V_{η_e} get lock: H, counter_w(e)=5				
t16	V_{η_e} obtain quorum				
t17	V_{η_e} update e				
t18	S is fragmented into S_1 and S_2				
t19	S_1 is fragmented into $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$				
t20	commit $\widehat{V}_{\lambda_e} \in V_\eta$	commit $\widehat{V}_{\lambda_e} \in V_\eta$	commit $\widehat{V}_{\lambda_e} \in V_\eta$	commit $\widehat{V}_{\lambda_e} \in V_\eta$	commit $\widehat{V}_{\lambda_e} \in V_\eta$
t21	unlock(e)	unlock(e)	unlock(e)	unlock(e)	unlock(e)

V. DISCUSSION

The proposed BVAGQ-AR has been compared with other replication techniques in terms of the total job execution time for a transaction. In this section, the total job execution time to update data between five existing techniques namely Dynamic Replication based on the Correlation of the File Strategy in Multi-Tier Data Grid Algorithm (BSCA) [5], A Prefetching-Based Replication Algorithm (PRA) [6], Hierarchical Replication Scheme (HRS) [7], Branch Replication Scheme (BRS) [7] and Read-One-Write-All (ROWA) [8, 9] have been compared with the proposed technique.

A. VALIDITY THREATS

Several validity threats can be associated with these experimental studies. Few threats have been identifies and their effects on the results are elaborated.

First, the benchmark choice represents an essential threat. The experimental benchmarks from other studies in literature have been adopted. However, we cannot guarantee these benchmarks represent the actual software and hardware configurations in real world. Nevertheless, the benchmarks are derived from configurations of different software programs.

Second, a comparison with other techniques is another threat. Other replication techniques with data mining such as BSCA and PRA are tested using simulation tools. This research focus on testing the replication technique in real time DDS because simulation cannot capture the problems that arise in real time environment. Nevertheless, the comparison is valid because all the techniques that we compared we have tested them using the same software and hardware in real time environment.

B. REPLICATION JOB EXECUTION TIME COMPARISON

Two series of experiments has been done in order to compare the job execution time for each technique. The first experiment is executed using the minimum number of replication servers of each replication technique. Table 4 shows the time comparison for the first experiment.

TABLE IV
COMPARISON OF JOB EXECUTION TIME FOR THE MINIMUM NUMBER OF REPLICATION SERVERS

Replication Techniques	Min. number of servers	Initiate Lock (ms)	Propagate Lock (ms)	Obtain Majority Quorum (ms)	Database Fragmentation & Commit (ms)	Total time taken:
BSCA	3	4.044	48.481	3.864	39.743	88.404
PRA	3	4.136	46.998	3.882	41.695	96.711
ROWA	9	4.275	144.522	8.187	105.259	262.243
HRS	9	3.956	147.227	7.870	98.875	257.928
BRS	8	4.523	64.268	8.112	60.254	137.157
BVAGQ-AR	3	3.905	16.369	3.890	42.384	66.548

Table 4 shows the execution time comparison between BSCA, PRA, ROWA, HRS, BRS and BVAGQ-AR in their minimum replication servers. From the Table 4, it is proved that BVAGQ-AR requires the lowest time to complete a transaction. It took only 66.548 milliseconds to complete a transaction. The second lowest execution time is BCSA with 88.404 milliseconds followed by PRA with total time taken is 96.711 milliseconds. PRA takes longer time due to user prefetching data from other servers. Next is BRS which takes 137.157 milliseconds to complete the replication process. ROWA and HRS takes the longest execution times which are more than 250 milliseconds. As it shown in the Table 4, there are big differences of total job execution time between BSCA and PRA with ROWA, BRS and HRS. This is because the data in ROWA, BRS and HRS is not mined since the original techniques do not consider the data correlation.

For the second experiment, it is executed using the maximum number of replication servers for each method. Table 5 shows the time comparison for the second experiment.

TABLE V
COMPARISON OF JOB EXECUTION TIME FOR THE MAXIMUM NUMBER OF REPLICATION SERVERS

Replication Techniques	Min. number of servers	Initiate Lock (ms)	Propagate Lock (ms)	Obtain Majority Quorum (ms)	Database Fragmentation & Commit (ms)	Total time taken:
BSCA	9	4.097	75.272	8.433	105.172	192.974
PRA	9	3.974	81.250	9.214	97.170	191.608
ROWA	9	4.275	147.498	8.002	107.912	267.687
HRS	9	4.152	146.136	9.107	107.536	266.931
BRS	9	4.480	64.864	8.835	93.993	172.172
BVAGQ-AR	5	4.280	23.808	3.950	51.830	83.868

Table 5 shows the execution time comparison between BSCA, PRA, ROWA, HRS, BRS and BVAGQ-AR for maximum replication servers. From the Table 6, again, it is proved that BVAGQ-AR requires the lowest time to complete a transaction as the maximum replication servers in this technique is only five. It took only 83.868 milliseconds for BVAGQ-AR to complete a transaction. The second lowest execution time is PRA with 191.608 milliseconds. This is followed by BSCA with total time taken is 192.974 milliseconds. Next is BRS which took 185.172 milliseconds to complete the replication process. ROWA and HRS took the longest execution times which are more than 250 milliseconds. Compare to other methods, BRS need less time to do a transaction because the data in this technique are fragmented and allocated at several different sites while other methods replicate all data to all sites.

TABLE VI
CBVAGQ - AR IMPROVEMENT IN TERMS OF JOB EXECUTION TIME (%)

REPLICA SERVERS	BSCA	PRA	ROWA	HRS	BRS
Minimum	31.19	24.72	74.62	74.20	51.48
Maximum	56.54	56.23	68.67	68.58	51.23

From Table 6, it is shown that, BVAGQ-AR has 31.19% improvement from BCSA when experiment is executed in minimum number of replication servers and 56.54% improvement in maximum number of replication servers. This is followed by PRA where BVAGQ-AR has 24.72% improvement from it in minimum number of replication servers and 56.23% improvement in maximum number of replication servers. The improvement in BSCA and PRA has a big different since in BVAGQ-AR, the minimum and maximum number of replication servers are 3 and 5 while in BSCA and PRA are 3 and 9. BVAGQ-AR had improved 74.62% from ROWA and 74.20% from HRS in minimum number of servers, 68.67% and 68.58% in maximum number of replication servers. There are not much different in the results since ROWA and HRS use 9 replication servers in both experiments. Last but not least is BRS, where BVAGQ-AR has 51.48% improvement from it in minimum number of replication servers and 51.23% improvement in maximum number of replication servers. The percentages are much higher in ROWA, HRS and BRS compare to BSCA, PRA and BVAGQ-AR because they do not take correlations between data into consideration. Hence, the processing times for these techniques are longer. In conclusion, BVAGQ-AR has the lowest job execution time to complete a transaction compare to BSCA, PRA, ROWA, HRS and BRS.

VI. CONCLUSION

In order to preserve data consistency and reliability of the systems, managing transactions is very important. BVAGQ-AR resolves this by setting the lock with small quorum size before update and commits transaction synchronously to the sites that has the same fragmented data. Since this technique using small size of quorum, less computational time is needed to send and receive messages from its neighbours' replicas. BVAGQ-AR only took only 66.548 milliseconds to complete a transaction while the second lowest execution time is BCSA with 88.404 milliseconds followed by PRA with total time taken is 96.711 milliseconds. PRA takes longer time due to user prefetching data from other servers. BRS takes 137.157 milliseconds to complete the replication process and ROWA and HRS takes the longest execution times which are more than 250 milliseconds. In addition, maintaining data consistency also easier compare to other techniques because it has low communication cost. This is because less computational time required for the locking of the small quorum size in synchronization process. From the experiment result, we can say that managing replication and transaction through proposed BVAGQ-AR able to preserve data consistency. It also increases the degrees of parallelism because by using fragmentation, replication and transaction can be divided into several subqueries that operate on the fragments. BVAGQ-AR can be improved in many different ways. As we know, server failure can happen anytime. Currently, BVAGQ-AR does not support handling fragmented database replication transaction management by considering failure cases. In future, BVAGQ-AR will take this challenge to handle fragmented database failure case and fault tolerance such as system crashes, statement failure, application software errors, network failure and media failure in real time distributed database system in real time environment.

ACKNOWLEDGMENT

The authors appreciate the Ministry of Higher Education Malaysia for additional supporting under Fundamental Research Grant Scheme, RDU190185 with Reference no: FRGS/1/2018/ICT03/UMP/02/3, and UMP Short Term Grant RDU1903122 and UMP PGRS RDU170329 for financing this research.

REFERENCES

[1] B. A. Milani and N. J. Navimipour, "A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions," *Journal of Network and Computer Applications*, vol. 64, pp. 229-238, 2016.

[2] J. Wang, H. Wu, R. Wang, "A new reliability model in replication-based big data storage systems,"

Journal of Parallel and Distributed Computing, vol. 108, pp. 14 – 27, 2017.

[3] S. A. U. Sharifah Hafizah, A. Noraziah, J. B, Odili, "Fragmentation Techniques for Ideal Performance in Distributed Database – A Survey," *International Journal of Software Engineering and Computer Systems*, vol. 6 (1), pp. 18 – 24, 2020.

[4] H. Mohamadi, P. V. Benjamin, A. Raymond, S. D. Jackman, J. Chu, C. P. Breshears, I. Birol, "DIDA: Distributed Indexing Dispatched Alignment," *PLOS ONE*, https://doi.org/10.1371/journal.pone.0126409 \, 2015.

[5] Z. Cui, D. Zuo, Z. Zhang, "Based on Support and Confidence Dynamic Replication Algorithm in Multi-Tier Data Grid," *International Journal of Computational Information Systems*, vol. 10, pp. 3909 – 3918, 2013.

[6] T. Tian, J. Luo, Z. Wu, A. Song, "A Pre-Fetching-Based Replication Algorithm in Data Grid," *Proceedings of the 3rd International Conference on Pervasive Computing and Applications.* vol. 1, pp. 526–531, 2008.

[7] J. M. Pérez, F. G. Carballeira, J. Carretero, A. Calderón, and J. Fernández, "Branch replication scheme: a new model for data replication in large scale data grids," *Future Generation Computer Systems*, vol. 26, pp. 12-20, 2010.

[8] S. Budiarto, N. M. Tsukamoto, "Data Management Issues in Mobile and Peer-to-Peer Environment*," Data and Knowledge Engineering*, vol. 41, 183-204, 2002.

[9] A. Noraziah, A. N. Abdalla, and M. S. Roslina, "Data Replication Using Read-One-Write-All Monitoring Synchronization Transaction Systems in Distributed Environment," *Journal of Computer Science*, vol. 6 (10), pp. 1033-1036, 2010.

[10] H.A. Abdulwahab, A. Noraziah, A. A. Alsewari, and S. Q. Salih, "An enhanced version of black hole algorithm via levy flight for optimization and data clustering problems," *IEEE Access*, vol. 7, pp.142085-142096, 2019.

[11] J. Han, M. Kamber, J. Pei, "Data Mining: Concepts and Techniques," *Morgan Kaufmann Publishers*, 2010.

[12] M. J. Zaki and W. Jr. Meira, "Data Mining and Analysis: Fundamental Concepts and Algorithms," *Cambridge University Press.*

[13] A. Sánchez, J. Montes, W. Dubitzky, J. J. Valdés, M. S. Pérez, P. D. Miguel, "Data Mining Meets Grid Computing: Time to Dance. In: Data Mining Techniques in Grid Computing Environments," *John Wiley & Sons*, pp. 1–16, 2008.

[14] T. Hamrouni, S. Slimani, F. B. Charrada, "A Critical Survey of Data Grid Replication Strategies Based on Data Mining Techniques," *Procedia Computer Science*, vol. 51, pp. 2779–2788, 2015a.

[15] T. Hamrouni, S. Slimani, F. B. Charrada, "A Survey of Dynamic Replication and Replica Selection Strategies Based on Data Mining Techniques in Data Grids," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 140–158, 2015b.

[16] R. Agrawal, J. C. Shafer, "Parallel mining of association rules," *IEEE Tran. Knowledge and Data Engineering*, vol. 8 (6), pp. 962 -969, 1996.

[17] M. Tu, *A data management framework for secure and dependable data grid*. University of Texas at Dallas, 2006.

[18] S. Doraimani, "Filecules: A new granularity for resource management in grids," 2007.

[19] S. Y. Ko, R. Morales, I. Gupta, "New Worker-Centric Scheduling Strategies for Data-Intensive Grid Applications," *Proceedings of the International Conference on Middleware*, pp. 121–142, 2007.

[20] A. Noraziah, A., C. F. Ainul Azila, M. S. Roslina, M. Z. Noriyani, and A. H. Beg, "Lowest Data Replication Storage of Binary Vote Assignment Data Grid," *Proceedings of International Conference on Networked Digital Technologies*, pp. 466 – 473, 2010.

[21] M. M. Deris, D. J. Evans, M. Y. Saman, A. Noraziah, "Binary Vote Assignment on Grid for Efficient Access of Replicated Data," *International Journal of Computer Mathematics*, vol. 80 (12), pp. 1489 – 1498, 2003.