ENHANCING BVAG DATA REPLICATION TRANSACTIONS WITH HIGH-PRIORITY-NEIGHBOUR FAULT TOLERANCE APPROACH

Sharifah Hafizah Sy Ahmad Ubaidillah[1], A. Noraziah[1*], Basem Alkazemi[2], Ahmad Shukri Mohd Noor[3], and Noriyani Mohd Zin[4]

[1]Faculty of Computing, University Malaysia Pahang Al-Sultan Abdullah,
Pahang, Malaysia

[2]Department of Software Engineering, College of Computing, Umm Al-Qura University,
Makkah 21955, Saudi Arabia

[3]Faculty of Computer Science and Mathematics, University Malaysia Terengganu,
Terengganu, Malaysia

[4]Mahabbah Empire Group Sdn Bhd., PT 4939, Tingkat 1, Bandar Baru Taman Kota Harmoni,
Jalan Cempaka Merah, 17500 Tanah Merah Kelantan, Malaysia

Emails: noraziah@umpsa.edu.my[*]

ABSTRACT

In distributed systems with failure interruption, the performance of database replication transactions might become very critical. Any distributed system that enforces data replication can be impacted by this problem. The fault tolerance approach is crucial to ensure the data replication transactions are always effective and dependable despite failures. The key advantage of fault tolerance is its capacity to complete the transaction notwithstanding a failure and restore system availability. This paper proposes a fault tolerance approach namely Binary-Vote-Assignment-Grid with High-Priority-Neighbour (BVAGHPN). It improves the efficiency of the data replication transaction in term of total execution time. This approach combines BVAG data replication transaction manager with the HPN to manage the transaction in the event of disasters. Instead of waiting for the problem to be fixed in the event of disaster, BVAGHPN halts the transaction on a failure replica, remove the failing replica from the alive quorum, and proceed the transaction with other replicas based on its own rating. BVAGHPN improves the outcomes of BVAG and BVAGCR in terms of the total execution time for two cases, PR failure and NR failure. For PR failure, BVAGHPN exceeds BVAG with 69.02% and BVAGCR (54.67%), respectively. Meanwhile, for NR failure, BVAGHPN improves BVAG with 76.88% and BVAGCR (71.97%).

Keywords: Data Replication; Distributed System; Fault Tolerance; Binary Vote Assignment on Grid; Computational Intelligence.

1.0 Introduction

Data replication support the Distributed Database System (DDS) since it delivers extremely dependable and effective access to critical data, and deployed to boost the distributed performance [1, 2, 3]. Furthermore, data replication is an effective technique that raise the efficiency of a computational intelligence system. Continuously,

moving data from one database to another is known as data replication, and it can improve availability, speed up response times, and enable efficient system local access [4, 5]. Data replication techniques have their own limitations when it comes to handling failure scenarios, despite being helpful for managing any distributed database system. A single failure could cause a serious disruption that includes lost data, downtime, and financial damages [6].

Controlling data replication is crucial to the system's ability to function in the event of a failure. One of the core issues with data replication is the need for replicas to ensure consistency in the case of updates, independent of any errors in execution [5]. To tackle these issues, the fault tolerance is the only viable solution. Fault tolerance is an important concept in distributed computing as it allows transactions to continue even when an error occurs. Its most crucial feature is the ability to carry on with the transaction even if one or more of its components fail [7, 8]. The fault tolerance in the distributed database systems is a proactive approach that maintains availability and dependability by keeping related transactions together. Resolving problems and, if possible, recognizing them are aided by effective fault tolerance mechanisms [9].

Researcher [10] integrates Binary Vote Assignment on Grid (BVAG) for data replication with Checkpoint and Rollback-Recovery (CR) fault tolerance approach. The purpose of CR is to create a checkpoint file that contains all of the relevant transactional data. The checkpoint file is then obtained each time failure recovery occurs, and the transaction is then carried out using the most recent checkpoint that was stored previously before failure occurred. The presented Binary Vote Assignment on Grid with Checkpoint Rollback Recovery (BVAGCR) eliminates the requirement for a transaction to start over from scratch, which is how it differs from the standard BVAG without fault tolerance approach. The research found that BVAGCR outperforms BVAG in the areas of execution time, U phase completion time, and time taken to restart and complete transactions after failure recovery.

The total time needed for BVAG and BVAGCR to finish a transaction, however, depends on how long it takes to bounce back from each particular setback. Put another way, it takes longer to complete a transaction the longer it takes to resolve the issue. These limitations could affect how well a database replication transaction performs. Consequently, High Priority Neighbour (HPN), a different fault tolerance approach has been combined with BVAG in this research. The Binary Vote Assignment on Grid with High Priority Neighbour Transaction Manager (BVAGHPNTM) acts as an eliminator for the failed copy without requiring the failure to wait for recovery.

The structure of the paper is described as followed. A thorough explanation of BVAG and follows in Section 2. The process involved in HPN Binary Vote Assignment on Grid with High Priority Neighbour (BVAGHPN) technique that is employed with BVAGHPNTM, has been covered in Section 3: Methodology. The results of the study are displayed along with an analysis that goes with them in the next Section 4. The rationale of the proposed data replication model is concluded in the last chapter, along with recommendations for additional exploration.

1.2 Literature Review

1.2.1 Binary Vote Assignment on Grid (BVAG)

Binary Vote Assignment on Grid (BVAG) duplicates data from the Primary Replica (PR) to its Neighbors' Replica (NR) sites that are situated near the PR [11]. Full replication requires a large quantity of data transfer and storage capacity [12]. Meanwhile, BVAG reduces the actual time of the replication transaction for DDS by replicating specific data at predefined sites. In addition, BVAG focuses on increasing write query durability via replication, whilst the query extension strategy entails supplementing the basic query with extra terms that are relevant to user needs [13]. By maximizing write availability while incurring minimum communication costs due to the use of a restricted number of quorum sizes, BVAG is charting a new course in replication. Replication and the processes of transaction are linked [14].

Each site in BVAG is logically organized into a grid form with two dimensions. Figure 1 illustrates how BVAG will spontaneously arrange into a $r \times r$ grid, where r represents the number of sites at the boundary. If C is logically adjacent to B and F, therefore physically, site C is adjacent to sites B and F, as illustrated by Figure 1. Consequently, the site in the center of the grid configuration has four neighbors, but the others alongside the borders only have three [14]. This is because four of the sites on the grid's corners have only two neighbors each.

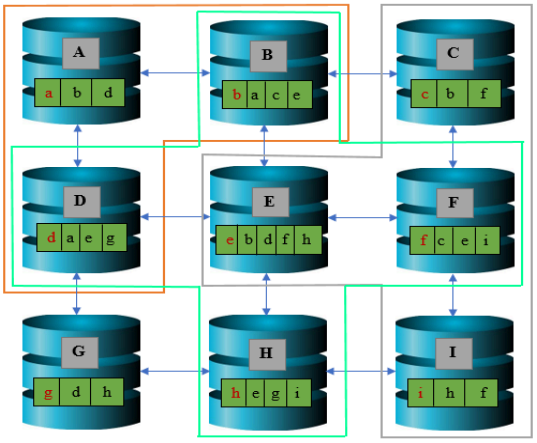


Fig. 1 Binary Vote Assignment on Grid (BVAG)

Every site has unique primary data. Copies of the Primary Replica (PR) data will be forwarded to the Neighbour Replica (NR) sites [11]. In general, the PR site of a data and its surrounding ones are rated as zero (0) or one (1). A neighbour BVAG, B *is* $B(z) \epsilon$ $\{0,1\}, 1 \leq z \leq s$ in which $B(z)$ denotes the vote specified to site z. A replica allocation happens whenever a vote is assigned to one site and an identical one is assigned to another. If data is duplicated to neighbours, the largest amount of replicas from each of the sites, determined by the value of s, should be as follows:

$$s \leq \text{quorum}, Q \text{ (the numbers of neighbours + primary sites)}$$

To give an example, neighbouring sites B and D receive copies of the primary data "e", from site A. The replicated data "e" will be distributed to the four sites B, D, F, and H that are close to site E. As a result, site E now has five clones. In the meanwhile, the primary data "h" from site H, is copied at sites E, G, and I. The number of quorums employed is determined by the total amount of replica and primary data, s that can vary between three, four, or five [11, 14].

Each transaction during the data replication process is manages by Binary Vote Assignment on Grid Transaction Manager (BVAGTM). In accordance with their state, the PR and NR sites receive different votes. Table 1 shows the two different kinds of votes used in the present research. If the value is zero, the site is free. In the meantime, one (1) indicated the site as busy. The technical conditions at each site differ from one another [11, 14].

Table 1 Type of Votes

Vote	Description
0	Free
1	Busy

BVAGTM consists of eight major phases: Initiate Lock (IL), Propagate Lock (PL), Obtain Quorum (OQ), Check Quorum (CQ), Update (U), Commit Replication (CR), Unlock (UL), and Release Lock (RL) [11,14]. The IL phase

entails securing the PR site while it is in the free (0) state. If the PR site is busy (1), it will be released (RL). Following the lockdown of the PR site, the status of each NR site is tracked during the PL phase. If any NR sites are accessible, they have all been locked. If not, the NR sites will be declared accessible (RL). The achieved quorum is then acceptable to proceed with the transaction, in accordance to the OQ phase.

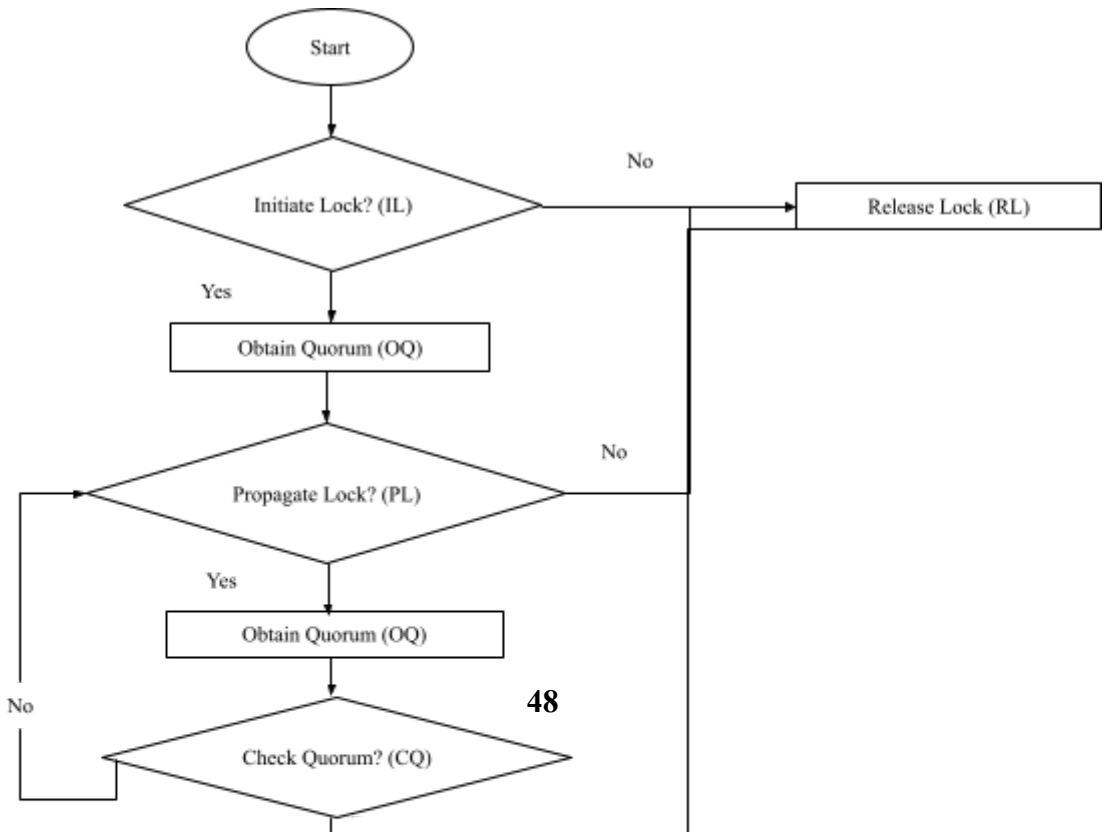


Fig. 2 Binary Vote Assignment on Grid Transaction Manager (BVAGTM)

1.2.2 High Priority Neighbour (HPN)

The High Priority Neighbour (HPN) fault tolerance strategy's main purpose is to determine which neighbour site will be the highest priority neighbour if PR site fails [15]. As shown in Table 2, the priority neighbour for each PR site is established via Primary-Neighbour Grid Coordination. NP represents the NR site with P priority.

The most crucial number is N1, followed by N2, N3, and N4. Each NR site is close to a PR location. When a PR site fails, the transaction's status is determined by the HPN. The primary concept behind the HPN fault tolerance technique is that the NR sites that are active have the highest priority ranking based on Primary-Neighbour Grid Coordination [15]. If any of the NR sites fails, the sites will be removed from the transaction. Because the failure site has been removed from the transaction, the transaction will be restarted without needing to wait for it to recover.

Table 2 Primary-Neighbours Grid Coordination

PR	NR			
	$NR1$	$NR2$	$NR3$	$NR4$
$m(x, y)$	$m(x-1, y)$	$m(x, y-1)$	$m(x, y)$	$m(x+1, y)$
$m(x-1, y)$	$m(x, y-1)$	$m(x, y+1)$	$m(x+1, y)$	$m(x, y)$
$m(x, y-1)$	$m(x, y+1)$	$m(x+1, y)$	$m(x, y)$	$m(x-1, y)$
$m(x, y+1)$	$m(x+1, y)$	$m(x, y)$	$m(x-1, y)$	$m(x, y-1)$

$m(x+1,y)$ $m(x,y)$ $m(x-1,y)$ $m(x,y-1)$ $m(x,y+1)$

1.3 Methodology

This section discusses the methodology for the suggested technique, BVAGHPN. The algorithm of the BVAGHPN technique was illustrated in Fig. 3 and Fig. 4 for a single data replication transaction using Binary Vote Assignment on Grid with High Priority Neighbour Transaction Manager (BVAGHPNTM). In BVAGHPN, the process of resuming the transaction with PR or NR failure is difference.

If PR failed, the PR is eliminated and replaced with a potential NR which is obtained based on p-priority rule. The transaction is then continued with new PR and the remaining NRs. Meanwhile, if NR is having a failure, the failed NR is eliminated and the transaction is continued with the original PR and the remaining NRs. The transaction done using BVAGHPNTM can be continued after the elimination of the failed site without having to wait for the failure to be resolved.

Table 3 shows the Primary-Neighbour Grid Coordination used in BVAGHPNTM. This rule is applied to determine which NR will be the highest Priority Neighbour (N) when PR failed to execute the current transaction. In this experiment, Site E which has the most neighbour replicas is used as the PR. As can be seen in the table below, for Site E, the highest priority neighbour (N_1) is Site B, followed by Site D (N_2), Site F (N_3) and lastly, Site H (N_4).

Table 3 Primary-Neighbours Grid Coordination in BVAGHPNTM

Primary Replica Site	Neighbour Replica Sites			
	N_1	N_2	N_3	N_4
A	B	D		
B	A	C	E	
C	B	F		
D	A	E	G	
E	B	D	F	H
F	C	E	I	
G	D	H		
H	E	G	I	
I	F	H		

There are also eight phases involved in BVAGHPNTM which are Initiate Lock (IL), Propagate Lock (PL), Obtain Quorum (OQ), Check Quorum (QR), Update (U), Commit Replication (CR), Unlock (UL) and Release Lock (RL). The HPN process take places after the specific transaction is interrupted by any failure. First, the following notations are defined as:

a) T is transaction.
b) λ represents different group of transaction T (before and until get quorum).

c) T_{λ} is transaction of group λ.

d) X is the data to be update.

e) q_i is the queue for transaction T_{λ}. i = 1,2,3, … .

f) $T_{\lambda_{x_{q_i}}}$ is database replication transaction λ for data χ in queue *i.*

g) S is the status of the required site.

h) PR stands for Primary Replica site.

i) NR stands for Neighbour Replica site.

j) NR_{N_i} is the Priority Neighbour Replica of PR based on Primary-Neighbours Grid Coordination, i = 1,2,3,… .

k) S_{PR} is status for PR.

l) $S_{NR_{N_i}}$ is status for NR_{PN_i}.

m) $S_{PR_{\lambda_{x_{q_i}}}}$ is the status of S_{PR} which hold data *x* in queue *i* for transaction λ.

n) $S_{NR_{N_i\lambda_{x_{q_i}}}}$ is the status of $S_{NR_{PN_i}}$ which hold data *x* in queue *i* for transaction λ.

o) W is the quorum needed to continue the transaction of $T_{\lambda_{x_{q_i}}}$.

p) $S_{W_{\lambda_{x_{q_i}}}}$ is the status of quorum *W,* which needed in order to replicate data *x* in queue *i* for transaction λ .

q) D is database.

r) D_{PR} is the database for PR.

s) D_{NR} is the database for NR.

t) $D_{PR_{\lambda_{x_{q_i}}}}$ is the database of D_{PR} that consists of data χ in queue *i* for transaction λ.

u) $D_{NR_{N_i}}$ is the database of D_{NR} for the neighbour *i*.

v) $D_{NR_{N_i\lambda_{x_{q_i}}}}$ is the database of $D_{NR_{N_i}}$ that consists of data χ in queue *i* for transaction λ.

w) $D_{PR_{\lambda_{x_{q_i}}}}$ is the database of D_{PR} that consists of data χ in queue *i* for transaction λ.

x) $D_{NR_{N_i\lambda_{x_{q_i}}}}$ is the database of $D_{NR_{N_i}}$ that consists of data χ in queue *i* for transaction λ.

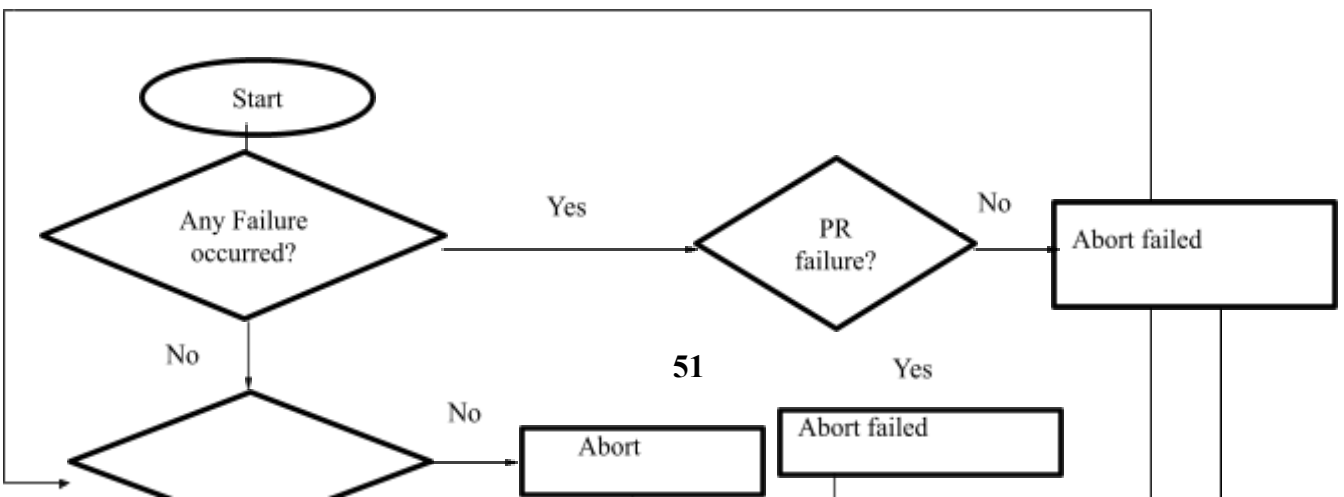


Fig. 3 Flowchart of BVAGHPNTM

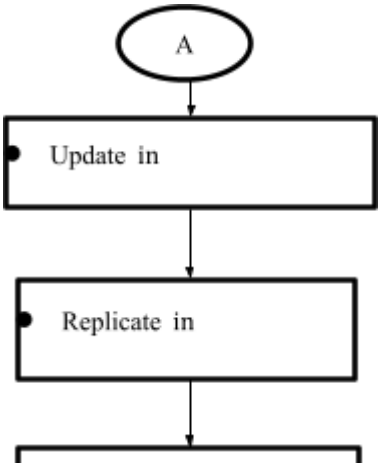


Fig. 4 Flowchart of BVAGHPNTM(continue)

Fig. 3 and Fig. 4 demonstrated the flowchart of BVAGHPNTM for a single data replication transaction with PR or NR failure. A new data replication transaction named as $T_{\lambda_{x_{q_i}}}$ which request to update data x is in first queue q_i in BVAGHPNTM.

Firstly, the transaction checked the status of the Primary Replica (PR) which hold the data x, $S_{PR_{\lambda_{x_{q_i}}}}$ whether it's status is free (0), busy (1) or having a failure (-1). This phase is called as Initiate Lock (IL). If the PR is free to be used in the transaction, the status will be set as 1 and the number of quorum, W is equal to 1. Else, the PR will be released because it is unavailable which is also known as Unlock (UL) phase.

Next, the transaction request to lock all the Neighbour Replicas (NRs), $S_{NR_{N_i}\lambda_{x_{q_i}}}$. This phase is called as Propagate Lock (PL). Each time the NR gets lock as 1, the number of quorum, W is increasing; the Obtain Quorum (OQ) phase. Once the number of W is equal to $S_{W_{\lambda_{x_{q_i}}}}$, the replication transaction can be proceed as in Check Quorum (CQ) phase. However, if the amount of W is not enough as per requested in the transaction, thus, all NRs are release from the transaction (RL). For BVAGHPN, each NR has its own N number which is set as NR_{N_i}.

Next, the Update (U) phase is executed. Data x is update at $D_{PR_{\lambda_{x_{q_i}}}}$ which is situated in the PR database. Afterward, the updated data x is replicate to all $D_{NR_{PN_i\lambda_{x_{q_i}}}}$. This process is called as CR phase. Lastly, the UL phase started. $S_{PR_{\lambda_{x_{q_i}}}}$ and all $S_{NR_{PN_i\lambda_{x_{q_i}}}}$ are unlock (status set to 0).

If any failure is detected, there is no need to wait for the recovery process of the failure site. The transaction was restarting again by firstly, removing the failed site. For *PR* failure case, the *PR* site is removed from current

transaction and the neighbour site named as NR_{N_1} was set as the new *PR*. Meanwhile, for any *NR* failure, the unsuccessful *NR* site is instead dropped from the transaction.

Next, the transaction was then restarted. The number of quorum *W* is reduced by one as one replica is removed and the new *W* the current transaction is set. All the remaining *NR* sites had their status changed to 1 (busy or in use). Later, the process of updating data *x* in the database $D_{PR_{\lambda_{x_{q_i}}}}$ located at $S_{PR_{\lambda_{x_{q_i}}}}$ was carried out.

Next, in *CR* phase, all of the $S_{NR_{N_i\lambda_{x_{q_i}}}}$ connected to the transaction were subjected to the replication process. As soon as the replication process was finished, all of the copies engaged in it were unlocked and their statuses were set to 0 and the transaction ($T_{\lambda_{x_{q_i}}}$) came to a close.

1.4 Result and Discussion

In this study, there are two type of failure expected to happen; PR failure and NR failure. It is assume that no case of all replica fail at a specific given point of time. Only one replica whether a PR or NR fail in a transaction. For BVAGCR and BVAGHPN, the replicas are being locked until the transaction is completed. Hence, there will be no intervention from other new invoking transactions during the failure recovery and restart process as in BVAG.

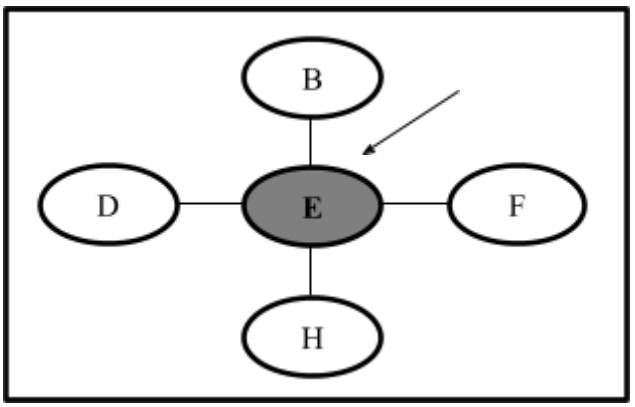


Fig. 5 Primary Replica (PR) E has a system failure

Fig. 5 illustrates the PR failure that is assumed to happen in Site E as the PR. For BVAGCR, the interrupted transaction $T_{\lambda_{x_{q_i}}}$ needs to wait for the failure to recover before it can resume functioning at the very last phase before it has been disturbed by the failure based on the information saved in CP file. Meanwhile, for BVAGHPN, the transaction can be resumed immediately by removing the PR and assign NR1 as the new PR which has been explained earlier in BVAGHPNTM. Then, the transaction in BVAGHPNTM can be restarted again.

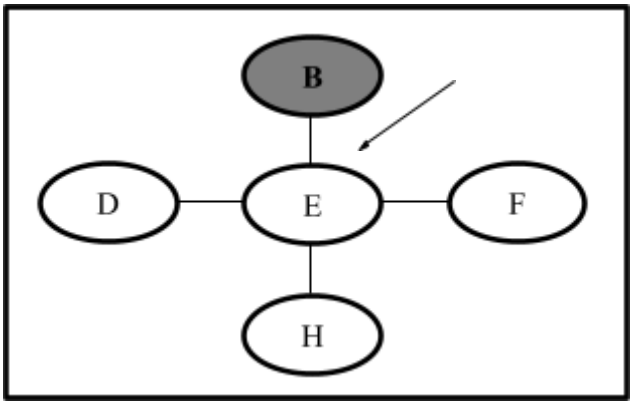


Fig. 6 Neighbour Replica (NR) B has a system failure

Fig.6 demonstrates the NR failure that is assumed to happen in Site B as the NR. For BVAGCR, same as in PR failure condition, the interrupted transaction $T_{\lambda_{x_{q_i}}}$ needs to wait for the failure to recover before it can resume functioning at the very last phase before it has been disturbed by the failure based on the information saved in CP file. In the meantime, for BVAGHPN, the transaction can be resumed immediately by removing the failed NR. Then, the transaction in BVAGHPNTM can be restarted again from the beginning.

For simplicity of all these failure cases, assume that only one transaction come to access a particular data file *x*. During the experimental session, the replica is assumed has been crashed in the Update (U) phase. MATLAB simulation was used to carry out the experiment. The benchmark model, BVAG and BVAGCR replication models applied the 10 second theory between failure and recovery. The transaction takes place after BVAG and BVAGCR have recovered from failures. For BVAGHPN, the transaction was immediately resumed following the removal of the unsuccessful site. The completion times for each phase of BVAGHPN was recorded before failure, during failure and after failure occurred for both cases.

1.4.1 Single Transaction with Primary Replica (PR) Failure

The experiment's transaction flow is displayed in Table 4. In this experiment, Site E is regarded in this transaction as the *PR* holding the primary data *e*. The neighbouring replicas that will receive a copy of the data from site E are sites B, D, F, and H.

Table 4 BVAGCRHPN Transaction's Flow of PR Failure

Replica	E	B	D	F	H
Time					
T_1	begin transaction $T_{\lambda_{e_{q1}}}$				
T_2	initiate lock: S_{E_e}				
T_3	obtain quorum and write lock $S_{E_e} = 1$				
T_4	set $S_{E_e} = PR$				
T_5		propagate lock: $S_{B_{e'}}$			
T_6		obtain quorum and write lock $S_{B_e} = 1$			
T_7		set $S_{B_e} = NR_1$			
T_7		check quorum: obtain majority quorum?			
T_8			propagate lock: $S_{D_{e'}}$		
T_{10}			obtain quorum and write lock $S_{D_e} = 1$		
T_{11}			set $S_{D_e} = NR_2$		
T_{12}			check quorum: obtain majority quorum?		
T_{13}				propagate lock: $S_{F_{e'}}$	
T_{14}				obtain quorum and write lock $S_F = 1$	
T_{15}				set $S_{F_e} = NR_3$	
T_{16}				check quorum: obtain majority quorum?	
T_{17}					propagate lock: $S_{H,}$
T_{18}					obtain quorum and write lock $S_H = 1$
T_{19}					set $S_H = NR_4$
T_{20}					check quorum: obtain majority quorum?
T_{21}	obtain majority quorum				
T_{22}	update (e) in D_{E_e}				
T_{23}	Failure Occur				
T_{24}	Remove PR				
T_{25}		Set $NR_{N_1} = PR$			
T_{26}		And Set new W begin transaction $T_{\lambda_{e_{q1}}}$			
T_{27}		initiate lock: S_{B_e}			
T_{28}		obtain quorum and write lock $S_{B_e} = 1$			

T_{29}	set $S_{B_e} = PR$			
T_{30}		propagate lock: $S_{D_{e'}}$		
T_{31}		obtain quorum and write lock $S_{D_e} = 1$		
T_{32}		set $S_{D_e} = NR_2$		
T_{33}		check quorum: obtain majority quorum?		
T_{34}			propagate lock: $S_{F_{e'}}$	
T_{35}			obtain quorum and write lock $S_F = 1$	
T_{36}			set $S_{F_e} = NR_3$	
T_{37}			check quorum: obtain majority quorum?	
T_{38}				propagate lock: $S_{H,}$
T_{39}				obtain quorum and write lock $S_H = 1$
T_{40}				set $S_H = NR_4$
T_{41}				check quorum: obtain majority quorum?
T_{42}	obtain majority quorum			
T_{43}	update (*e*) in D_{B_e}			
T_{44}		Commit Replication of data *(e)* in D_{D_e}	Commit Replication of data *(e)* in D_{F_e}	Commit Replication of data *(e)* in D_{H_e}
T_{45}	write lock $S_{B_e} = 0$	write lock $S_{D_e} = 0$	write lock $S_{F_e} = 0$	write lock $S_{H_e} = 0$
T_{46}	End transaction $T_{\lambda_{e_{q1}}}$			

According to Table 4, the transaction of $T_{\lambda_{e_{q1}}}$ started at time corresponding to 1 (T_1). It requested to update data *e* at siteS_E. The transaction initiates lock for site S_{E_e} at T_2. Hence, the status of S_{E_e} is equal to 1 for the time being and S_E was set as the *PR* in this experiment (T_4). There was now 1 in the quorum.

At T_5 to T_{20} , $T_{\lambda_{e_{q1}}}$ propagated lock for data *e* at its neighbours site which are site $S_{B_{e'}}$, $S_{D_{e'}}$, S_{F_e} and $S_{H,}$. The neighbours were set as NR_1, NR_2, NR_3 NR_4 each time the neighbor sites were locked, adding 1 to the quorum in the process.

If there was a sufficient quorum to continue the transaction (see T_{21}), the update procedure was carried out. All the sites (*PR* and *NR* sites) were, however, released from the transaction if there weren't enough quorums. As in this instance, there were enough quorums to move on with the transaction.

The process of updating the data in database D_{E_e} situated at S_{E_e} then starts at time T_{22}. However, an error was encountered when updating the precise data as in T_{23}. For this experiment, the *PR* was having the failure at T_{24} .

For BVAGHPN, there is no need to wait for the recovery process of the specific site which is having a failure. Instead, the failed PR site is removed from the transaction and been replaced with NR_{N_1}. the number of quorum W is reduced by one. Then, the transaction was restart again (T_{30}).

Later, the process of updating data *e* in the database D_{B_e} located at S_{B_e} was carried out (T_{43}). The *CR* phase had already begun, as seen in T_{44}. All of the *NRs* connected to the transaction were subjected to the replication process. As soon as the replication process was finished, as in T_{45}, all of the copies engaged in it were unlocked and their statuses were set to 0. At T_{46}, the transaction ($T_{\lambda_{e_{q1}}}$) came to its end.

In the case of PR failure (refer Table 5), BVAGHPN requires a total of 4.9124 seconds to complete a transaction. Before malfunctioning (highlighted in blue), the transaction went through four phases: IL, PL, CQ, and OQ. These phases took 0.2561 seconds to complete. The transaction was stopped during failure occur and entered the HPN phase (highlight in orange), which removed the failed PR site and replaced it with NR1. This stage was completed in 0.2506 seconds. The transaction was then reopened using the new PR site. After being restarted (highlight in green), the transaction was completed in 4.4057 seconds. The amount of time required to complete the crucial phase (U) is 3.8672 seconds.

Table 5 Execution time for BVAGHPN data replication transaction with failure at PR

CONDITIONS	PHASES	TIME(SECONDS)
BEFORE FAILURE OCCUR	INITIATE LOCK^{A}	0.2397
	PROPAGATE LOCK^{A}	0.0164
	CHECK QUORUM^{A}	0.0000
	UPDATE, FAILURE OCCUR	0.0000
DURING FAILURE / FAILURE RECOVERY	HPN and RESTART	0.2506
AFTER FAILURE RECOVERY	INITIATE LOCK^{B}	0.4820
	PROPAGATE LOCK^{B}	0.0083
	OBTAIN QUORUM^{B}	0.0000
	UPDATE^{B}	3.8672
	COMMIT REPLICATION^{B}	0.0331
	UNLOCK^{B}	0.0151
	TOTAL	4.9124

Table 6 shows the comparisons of time taken before failure occur, during failure, after failure recovery, during update phase and total execution time of a transaction done using BVAG, BVAGCR and BVAGHPN. Based on the table below, it can be seen that both hybrid data replication models (BVAGCR and BVAGHPN) outperformed standard BVAG (no fault tolerance approach) in terms of their overall execution time to complete a transaction. BVAGCR has outperformed standard BVAG by 30.58% while BVAGHPN has also improved the performance of standard BVAG by 69.02%. In the same time, BVAGHPN had also outperformed BVAGCR by 54.67% as a result of eliminating the failed replica.

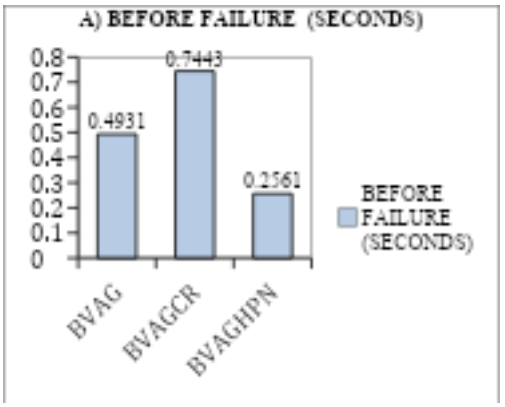
Table 6 Time comparison between standard BVAG, BVAGCR and BVAGHPN (PR Failure)

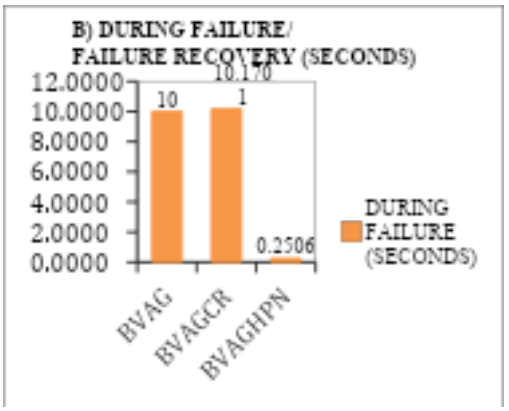
TIME (SECONDS)	BVAG	BVAGCR	BVAGHPN

BEFORE FAILURE OCCUR	0.4931	0.7443	0.2561
DURING FAILURE / FAILURE RECOVERY	10.0000	10.1701	0.2506
AFTER FAILURE RECOVERY	5.3643	0.0938	4.4057
UPDATE PHASE	4.7809	0.0516	3.8672
TOTAL TIME	15.8574	11.0082	4.9124
IMPROVEMENT FROM BVAG (%)		30.58%	69.02%
IMPROVEMENT FROM BVAGCR (%)			54.67%

More details explanation for each time phase was explained based on Fig. 7. Fig. 7 (A) makes it obvious that BVAGCR required the longest amount of time (0.7443s) before failure occurred. This is due to the fact that BVAGCR must save each and every piece of data in the Checkpoint file for backup and recovery. As a result, at this point in time, it took longer than other data replication models. Compared to standard BVAG and BVAGCR, BVAGHPN took the least amount of time (0.2561s) when a failure occurred in Fig. 7 (B).

As can be seen in Fig. 7 (C), BVAGCR took the shortest time (0.0938s) to complete the transaction after failure recovery. This is due to the existence of the Checkpoint file which is used to keep all the information happened before failure occurred. Thus, the transaction did not need to be restart from the beginning, but, it only need to retrieve the information saved previously and restart the transaction from the last saved phase. The Update phase (Fig. 7 (D)) happened after failure had been resolved. Thus, this means that the information saved in the Checkpoint file can be retrieved for BVAGCR. This is why BVAGCR took the shortest time (0.0516s) to complete the Update phase. For standard BVAG and BVAGHPN, the time taken after failure and in Update phase are longer than BVAGCR because there is no Checkpoint file has been implemented in both data replication models. Thus, both of them need to be restart from the beginning of the transaction.





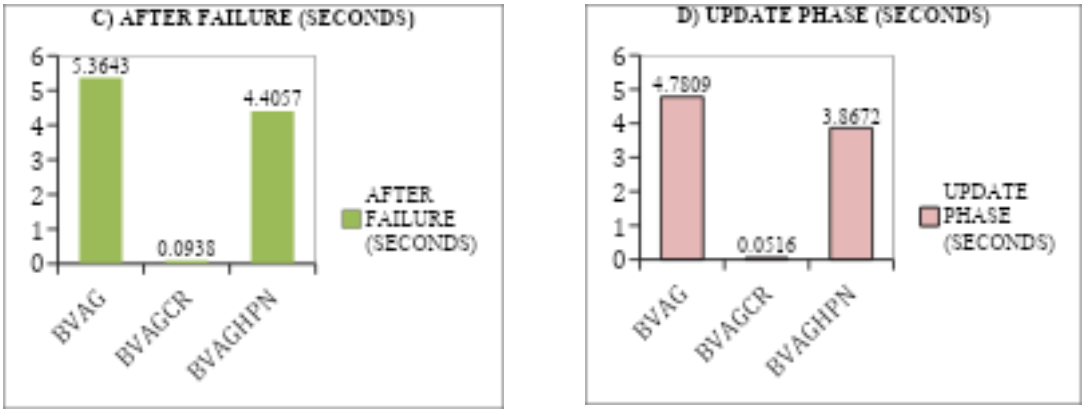


Fig. 7 Details on Time Comparison for Standard BVAG, BVACGR and BVAGHPN with PR failure

1.4.2 Single Transaction with Neighbour Replica (NR) Failure

Table 5 displays the experiment's transaction flow. In this experiment, Site E is regarded as the primary data holder in this transaction. Sites B, D, F, and H will receive a duplicate of the data from site E.

Table 5 BVAGCRHPN Transaction's Flow of NR Failure

Replica	E	B	D	F	H
Time					
T_1	begin transaction $T_{\lambda_{e_{e1}}}$				
T_2	initiate lock: S_{E_e}				
T_3	obtain quorum and write lock $S_{E_e} = 1$				
T_4	set $S_{E_e} = PR$				
T_5		propagate lock: $S_{B_{e'}}$			
T_6		obtain quorum and write lock $S_{B_e} = 1$			
T_7		set $S_{B_e} = NR_1$			
T_8		check quorum: obtain majority quorum?			
T_9			propagate lock: $S_{D_{e'}}$		
T_{10}			obtain quorum and write lock $S_{D_e} = 1$		
T_{11}			set $S_{D_e} = NR_2$		

T_{12}			check quorum: obtain majority quorum?		
T_{13}				propagate lock: $S_{F_{e'}}$	
T_{14}				obtain quorum and write lock $S_F = 1$	
T_{15}				set $S_{F_e} = NR_3$	
T_{16}				check quorum: obtain majority quorum?	
T_{17}					propagate lock: $S_{H,}$
T_{18}					obtain quorum and write lock $S_H = 1$
T_{19}					set $S_H = NR_4$
T_{20}					check quorum: obtain majority quorum?
T_{21}	obtain majority quorum				
T_{22}	update (e) in D_{E_e}				
T_{23}	Failure Occur				
T_{24}		Remove NR_1 and W--			
T_{25}	initiate lock: S_{E_e}				
T_{26}	obtain quorum and write lock $S_{E_e} = 1$				
T_{27}	set $S_{E_e} = PR$				
T_{28}			propagate lock: $S_{D_{e'}}$		
T_{29}			obtain quorum and write lock $S_{D_e} = 1$		
T_{30}			set $S_{D_e} = NR_2$		
T_{31}			check quorum: obtain majority quorum?		
T_{32}				propagate lock: $S_{F_{e'}}$	
T_{33}				obtain quorum and write lock $S_F = 1$	
T_{34}				set $S_{F_e} = NR_3$	
T_{35}				check quorum: obtain majority quorum?	
T_{36}					propagate lock: $S_{H,}$
T_{37}					obtain quorum and write lock $S_H = 1$
T_{38}					set $S_H = NR_4$
T_{39}					check quorum: obtain majority quorum?
T_{40}	obtain majority quorum				
T_{41}	update (e) in D_{E_e}				
T_{42}			Commit Replication of data *(e)* in D_{D_e}	Commit Replication of data *(e)* in D_{F_e}	Commit Replication of data *(e)* in D_{H_e}
T_{43}	write lock $S_{E_e} = 0$		write lock $S_{D_e} = 0$	write lock $S_{F_e} = 0$	write lock $S_{H_e} = 0$
T_{44}	End transaction $T_{\lambda_{e_{e1}}}$				

The transaction of $T_{\lambda_{e_{q1}}}$ began at the time corresponding to 1 (T_1), as shown in Table 5. It asked for the site S_E data to be updated (data *e*). Site S_{E_e} was locked by the transaction at time T_2. As a result, S_{E_e}'s status is currently set to 1, and S_E was chosen as the *PR* for this experiment (T_4). The quorum now consisted of one.

$T_{\lambda_{e_{q1}}}$ propagated a lock for data *e* at its neighboring sites, $S_{B_{e'}}$, $S_{D_{e'}}$, S_{F_e} and S_{H_e} from T_5 to T_{20}. Each time the neighbor sites were locked, the neighbours were set to NR_1, NR_2, NR_3, NR_4 raising the quorum by 1. The update operation can be done if there was a sufficient quorum to continue the transaction (see T_{21}). However, if there weren't enough quorums, the transaction was aborted for both *PR* and *NR* sites. There were sufficient quorums to proceed with the transaction in this case.

At time T_{22}, the procedure for updating the data in database D_{E_e} located at S_{E_e} begins. When updating the precise data, as in T_{23}, an error was made. The failure for this experiment happened at NR_1 ($S_{B_{e'}}$). There is no need for BVAGHPN to wait for the specific site's failure site to complete its recovery process. The unsuccessful site is instead dropped from the transaction (The current NR_1 was then removed (T_{24})). The number of quorum *W* is reduced by one as one replica is removed. The other three neighbours that still available to be utilized had been assigned to the transaction at that point.

The transaction was then restarted. The operation of updating the data *e* in the database D_{E_e} at S_{E_e} was then completed (T_{41}). As demonstrated in T_{42}, the *CR* phase had already started. The replication procedure was applied to every *NR* involved in the transaction. Like in T_{43}, as soon as the replication operation was complete, all of the copies involved were unlocked and had their statuses set to 0. The transaction ($T_{\lambda_{e_{q1}}}$) came to a close at T_{44}.

BVAGHPN takes 3.0806 seconds to finish a transaction in the event of a NR failure (see Table 6). Four steps of the transaction occurred before the issue existed (highlighted in blue): IL, PL, CQ, and OQ. It took 0.0345 seconds to finish these steps. When the failure occurred, the transaction was halted and moved into the HPN phase (highlighted in orange), which eliminated the failed NR site. The completion time for this step was 0.0043 seconds. After that, the transaction was resumed without the failed NR site. 3.0418 seconds were needed to finish the transaction once it was restarted (highlighted in green). The time needed to finish the critical stage (U) is 2.6926 seconds.

Table 6 Execution time for BVAGHPN data replication transaction with failure at one of NR

CONDITIONS	PHASES	TIME(SECONDS)
BEFORE FAILURE OCCUR	INITIATE LOCK[A]	0.0039
	PROPAGATE LOCK[A]	0.0306
	CHECK QUORUM[A]	0.0000
	UPDATE, FAILURE OCCUR	0.0000
DURING FAILURE / FAILURE RECOVERY	HPN and RESTART	0.0043
AFTER FAILURE RECOVERY	INITIATE LOCK[B]	0.0049
	PROPAGATE LOCK[B]	0.0199
	OBTAIN QUORUM[B]	0.0000
	UPDATE[B]	2.6926
	COMMIT REPLICATION[B]	0.2961

UNLOCK[B]	0.0283
TOTAL	3.0806

Table 7 shows the comparisons of time taken before failure occur, during failure, after failure recovery, during update phase and total execution time of a transaction done using BVAG, BVAGCR and BVAGHPN. Based on the table below, it can be seen that both hybrid data replication models (BVAGCR and BVAGHPN) outperformed standard BVAG (no fault tolerance approach) in terms of their overall execution time to complete a transaction. BVAGCR has outperformed standard BVAG by 17.51% while BVAGHPN has also improved the performance of standard BVAG by 76.88%. In the same time, BVAGHPN had also outperformed BVAGCR by 71.97% as a result of eliminating the failed replica.

Table 7 Time comparison between standard BVAG, BVAGCR and BVAGHPN (NR Failure)

TIME (SECONDS)	BVAG	BVAGCR	BVAGHPN
BEFORE FAILURE OCCUR	0.0079	0.0499	0.0345
DURING FAILURE / FAILURE RECOVERY	10.0000	10.4807	0.0043
AFTER FAILURE RECOVERY	3.3183	0.4616	3.0418
UPDATE PHASE	3.3100	0.3030	2.6926
TOTAL TIME	13.3262	10.9922	3.0806
IMPROVEMENT FROM BVAG (%)		17.51%	76.88%
IMPROVEMENT FROM BVAGCR (%)			71.97%

More details explanation for each time phase was explained based on Fig. 8. Fig. 8 (A) shows that standard BVAGCR required the longest amount of time (0.0499s) before failure occurred. This is due to the fact that BVAGCR must save each and every piece of data in the Checkpoint file for backup and recovery. As a result, at this point in time, it took longer than other data replication models. Compared to standard BVAG and BVAGCR, BVAGHPN took the least amount of time (0.0043s) when a failure occurred and during failure recovery in Fig. 8(B). This is because, rather than waiting for the failure to be resolved, BVAGHPN automatically removed the failed replica and restarts the transaction from the very beginning immediately.

As can be seen in Fig. 8(C), BVAGCR took the shortest time (0.4616s) to complete the transaction after failure recovery. This is due to the existence of the Checkpoint file which is used to keep all the information happened before failure occurred. Thus, the transaction did not need to be restart from the beginning, but, it only need to retrieve the information saved previously and restart the transaction from the last saved phase. The Update phase (Fig. 8(D)) happened after failure had been resolved. Thus, this means that the information saved in the Checkpoint file can be retrieved for BVAGCR. This is why BVAGCR took the shortest time (0.303s) to complete the Update phase. For standard BVAG and BVAGHPN, the time taken after failure and in Update phase are longer than BVAGCR because there is no Checkpoint file has been implemented in both data replication models. Thus, both of them need to be restart from the beginning of the transaction.

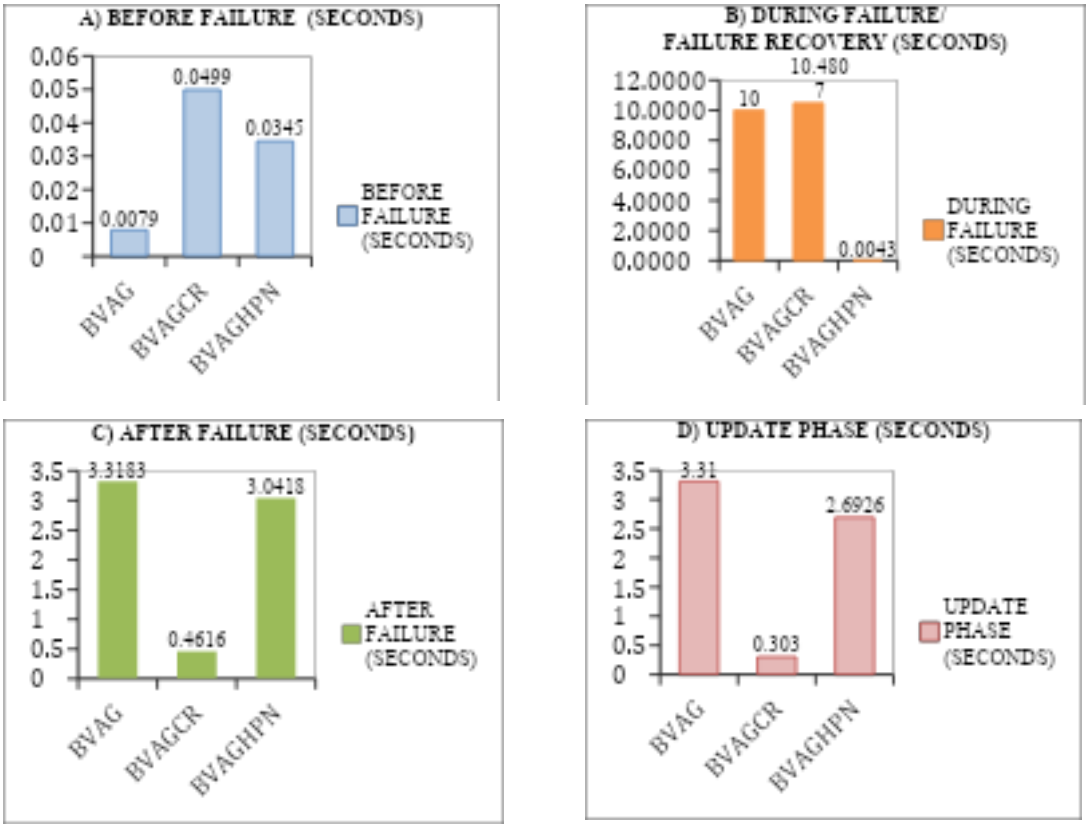


Fig. 8 Details on Time Comparison for Standard BVAG, BVACGR and BVAGHPN with NR failure

1.5 Conclusion

This study investigated the BVAGHPN technique, which is a combination of data replication technique with fault tolerance approach. According to the findings, the suggested BVAGHPN reduced the overall time required for a single database replication transaction when compared to the regular BVAG and BVAGCR. Due to HPN's elimination of the failed replica, the transaction can now be restarted right away without having to wait for the damaged replica to be fixed. BVAGHPN, however, requires more time during the after-failure phase and the update phase. This is because there is no backup data saved during the transaction before failure happened as in BVAGCR.

Therefore, it is recommended that the BVAG data replication technique be used with both HPN and CR fault tolerance approaches in future work. This is due to the possibility that by combining both fault tolerance approaches, the data replication model's performance may be enhanced and may even be superior to utilizing a single fault tolerance approach. Additionally, the suggested data replication model can be used in big data transactions to determine if it can help to increase the effectiveness of the big data management system.

Acknowledgments

This research is supported by Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), Distinguish Research Grant under RDU253002. Special thanks to Prof. Dr. Basem Alkazemi from Umm Al-Qura University for also funding this paper. The authors also acknowledge UMPSA for providing laboratory facilities, library services, and the UMP PGRS Grant under RDU170329.

REFERENCES

[1] N. Dogra and S. A. Singh, “A survey of dynamic replication strategies in distributed systems,” International Journal of Computer Applications., vol. 110, no. 11, Jan 2015.

[2] S. H. S. A. Ubaidillah, and A. Noraziah and Sahabudin, "A Survey on Potential Reactive Fault Tolerance Approach for Distributed Systems in Big Data" Proc. SPIE 12590, Third International Conference on Computer Vision and Information Technology (CVIT 2022), 1259009, 2023.

[3] Ahmad Shukri Mohd Noor, Auni Fauzi Che Fauzi, Ainul Azila Che Fauzi, Noraziah Ahmad, MohamadSyauqi Mohamad Arifin. "An automate failure recovery for synchronous distributed database system," Indonesian Journal of Electrical Engineering and Computer Science., vol. 28, no. 2, November 2022.

[4] Kumar, Nirmal, and Girish V. Mattur. "Data replication in a database environment." U.S. Patent Application No. 15/813,828.

[5] B. Kemme, “Data Replication,” In: Liu L., Özsu M. (eds) Encyclopedia of Database Systems. Springer, New York, NY, 2017.

[6] Sahran, Firdaus, and Nor Badrul Anuar. "Scalability Improvement of Active Probing for Failure Detection in Large-Scale Software Defined Networks using Community Detection Algorithms." Malaysian Journal of Computer Science, vol. 36, no. 4, pp. 404-425, Oct 2023.

[7] A. Sarı and E. Çağlar, “Performance Simulation of Gossip Relay Protocol in Multi-Hop Wireless Networks,” Owner: Girne American University Editor: Asst. Prof. Dr. İbrahim Erşan Advisory Board: Prof. Dr. Sadık Ülker Assoc. Prof. Dr. Zafer Ağdelen Cover Graphic Design: Asst. Prof. Dr. İbrahim Erşan., pp. 145, 2015.

[8] Ubaidillah, S. H. S. A., Ahmad, N., & Sahabudin, N. A., "A survey on potential reactive fault tolerance approach for distributed systems in big data," In Third International Conference on Computer Vision and Information Technology (CVIT 2022). vol. 12590, pp. 61-68, SPIE, February 2023.

[9] A. Sari and M. Akkaya, “Fault tolerance mechanisms in distributed systems,” International Journal of Communications, Network and System Sciences., vol. 8, no. 12, pp. 471, Dec 2015.

[10] Ubaidillah, Sharifah Hafizah Sy Ahmad, Basem Alkazemi, and A. Noraziah. "An Efficient Data Replication Technique with Fault Tolerance Approach using BVAG with Checkpoint and Rollback-Recovery", International Journal of Advanced Computer Science and Applications, vol. 12, no. 1, pp. 473-480, 2021.

[11] A. Noraziah, A. A. Fauzi, S. H. Ubaidillah, Z. Abdullah, R. M. Sidek, M. A. Fakhreldin, “Managing Database Replication Using Binary Vote Assignment on Grid Quorum with Association Rule,” Advanced Science Letters., vol. 24, no. 10, pp. 7834-7837, Oct 2018.

[12] J. P. Yang, “Elastic load balancing using self-adaptive replication management,” IEEE Access., vol. 22, no. 5, pp. 7495-7504, Nov 2016.

[13] Muhammad Ahsan Raza, M. Rahmah, A. Noraziah, Mahmood Ashraf, “Sensual Semantic Analysis for Effective Query Expansion “, International Journal of Advanced Computer Science and Applications, vol. 9, no. 12, pp. 55-60, 2018.

[14] A. A. Fauzi, A. Noraziah, T. Herawan, Z. Abdullah, R. Gupta, “Managing Fragmented Database Using BVAGQ-AR Replication Model,” Advanced Science Letters., vol. 23, no. 11, pp. 11088-11091, Nov 2017.

[15] N. Ahmad, “Managing Replication and Transactions using Neighbour Replication on Data Grid”, (Doctoral dissertation, UMT).