Malaysian Journal of Computer Science, Vol. 17 No. 2, December 2004, pp. 11-20

NEIGHBOR REPLICA DISTRIBUTION TECHNIQUE FOR CLUSTER SERVER SYSTEMS

Rabiei Mamat, Mustafa Mat Deris and Mashita Jalil
University College of Science and Technology Malaysia
Faculty of Science and Technology
21030 MengabangTelipot, Kuala Terengganu, Malaysia
email: rab@kustem.edu.my

ABSTRACT

Data replication increases the reliability and availability of a web server cluster. However, different data replication techniques have different reliability levels. In this paper, Neighbor Replica Distribution Technique (NRDT) has been proposed to improve the reliability of a web cluster server. This technique provides high reliability by imposing a neighbor logical structure on data copies. Data from one server will be replicated to its neighboring servers and vice versa in the case of failure. The algorithm of NRDT data replica scheme based on the asynchronous replication results in a higher reliability. It shows that this technique provides a convenient approach to high reliability for web server cluster.

Keywords: Distributed database, Replicated data, Reliability, Storage capacity

1.0 INTRODUCTION

With the ever increasing applications in WWW such as e-learning and e-commerce, the need for a reliable of the non-stop cluster server (CS) is likely to increase [1]. Reliability refers to the probability that the system under consideration does not experience any failures in a given time interval. Therefore, a reliable CS system is one that can continue to process user requests even when the underlying system is unreliable [2]. The system is unreliable due to unavailability of data or hardware failure. Thus, providing reliable and efficient services are the primary goals in designing a CS. It is an important mechanism because it enables organisations to provide users with access to current data when they need it. In order to provide reliable services, a CS needs to maintain the data on some replicas [3]. Therefore, data replication plays an important role in the CS environment to become a highly reliable system.

Several techniques have been proposed in managing replicated data. However, different techniques have different reliability levels of managing replicated data. They can be broadly classified into two categories. The first is called synchronous replication while the second approach is referred to as asynchronous replication. For the case of synchronous replication, one of the simplest techniques for managing this is called read-one write-all (ROWA). In this technique, read operations on an object are allowed to read any copy, and write operations are required to write all copies of the data object [5]. This protocol provides read operations with high degree of availability at low cost but severely restricts the availability of write operations since they cannot be executed at the failure of any copy. It results in the imbalance of data availability and communication cost of read and write operations, where read operations have high availability and low communication cost whereas write operations have low availability with higher communication cost. Another set of protocols called voting protocols [4, 1, 6, 7] became popular because they are flexible and are easily implemented. One of which is called weighted voting. The communication cost is dependent on the number of votes pre-selected for read and write operations. Because of the need to poll multiple copies before every read operation, the workload with high proportion of reads will not perform well [1]. However, the problems of voting techniques have been discussed widely in the [1]. Generally, one of the weaknesses of voting protocols is that writing an object is fairly expensive: A write quorum of copies must be larger than the majority of votes [8, 9, 10, 6, 11]. Several researchers have proposed imposing logical structure on the set of copies in the database, and using logical information to create intersecting quorums [12, 13, 8]. Nevertheless, synchronous replication has several drawbacks in practice [4]. The major argument is that the response time to execute the operations is high. This is due to the high time taken in order for all nodes that have the same copy to '*agree*' to execute an operation. Asynchronous replication provides what is called 'loose consistency' between data stores, where the consistency achieved between data will always be greater than zero [5]. Nevertheless, its response time is lower than the synchronous technique [4].

In this paper, without loss of generality, the terms server and node will be used interchangeably. If data is replicated to all nodes, the storage capacity becomes an issue, thus an optimum number of nodes to replicate the data is required with non-tolerated reliability of the CS system. Recently, Two-Replica Distribution Technique (TRDT) has been proposed by Shen et al [15, 16]. In this technique, there are N nodes in the CS and each node has an equal capacity of storage and all data have two replicas on different nodes and all nodes have two data replicas [15]. With N nodes, it divides to n set of nodes (N=2n) where each set consists of two nodes as illustrated in Fig. 1.

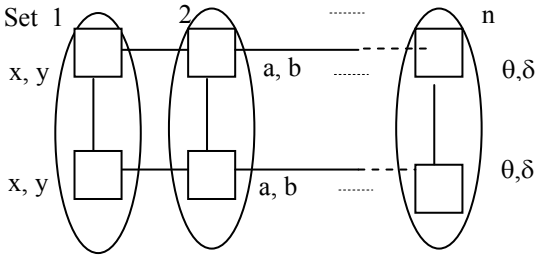

Fig. 1: Data replica distribution technique when N = 2n

The rectangle-shape shows the node, and each oval is a set that consists of two nodes. Data x and y have two replicas, that is in nodes from set 1, and data a and b also have two replicas which are located in nodes from set 2, and so on. If N = 2n+1, then all nodes will be divided into n-1 sets with two nodes and one set with three nodes. In such a case, each node of n-1 sets has two replicas whereas nodes of another set have three replicas. The shortcoming of this technique is that the system should have replica-availability of more than 99% in order to achieve high reliability. Also, if one set is unavailable, the operations cannot be constructed.

In this paper, we describe a technique called Neighbor-Replica Distribution Technique (NRDT) to improve the reliability of the CS. We only concentrate on the implementation of replication based on a lower response time, which is the asynchronous replication. We consider the update propagation is based on the *immediate-immediate* (for propagation and update) strategy as proposed in [4]. Each write operation performed by a transaction, T, is *immediate*ly propagated to neighboring nodes, without waiting for the commitment of the original update transaction T. At a neighbor node, an update transaction is started as soon as the first write operation is received from the primary node, and this is term as *immediate* updating. Thus, this scheme relaxes the mutual consistency property of two-phase commit (2PC).

The rest of the paper is as follows: In Section 2, the system model is discussed. In Section 3, the neighbor-replica distribution technique and algorithm is proposed. The update propagation strategy is presented in Section 4. Finally, in Section 5, the performance of the proposed technique is analysed and compared with other technique.

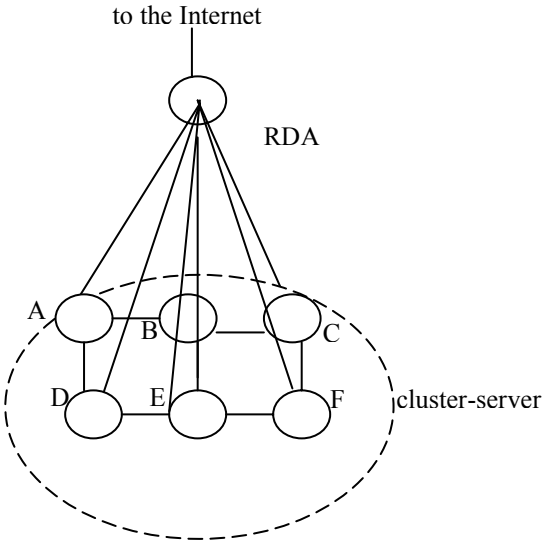

Fig. 2: A cluster with 6 servers

2.0 SYSTEM MODEL

The architecture of a CS refers to a set of server machines that are housed together in a single location. The servers are interconnected through a high-speed network. Each cluster server (hereafter called node) may be a workstation, a PC or a symmetric multiprocessor. Our design of the CS is that a client on the Internet will notice only one IP address coming from the cluster, not those of the individual servers in the cluster. It consists of a Request Distributor Agent (RDA) and a group of servers. The RDA holds the IP address of the cluster. Those servers are assigned with IP addresses. The servers are logically connected to each other in the form of a grid structure, each of which is connected together with the RDA. For example, Fig. 2 shows the cluster-server system with 6 servers. The RDA forwards legitimate Internet requests to the appropriate servers in the clusters, and returns replies from the servers back to the clients. Throughout this paper, we will use the term cluster server to refer to a non-RDA node in the cluster.

Assume that the master data *file a* will be located on server A, *file b* on server B, and so on. Since each node has the master data file, then the RDA will forward request to one of the primary servers depending on the request requirement. For example, if a client request is related to *file b* then the RDA will forward that request to server B. Each node, whether a primary or neighbor, supports service types and two components as shown in Fig. 3. The first component is the Replication module, which itself consists of three components: Log Monitor, Propagator and Receiver. The second component is the network interface, which is used to propagate and receive messages on the network. Detail of the discussion on the functionality of the Log Monitor, Propagator and Receiver can be found in [4].

One advantage a CS has over a single server is its high security. If a single server is used, it is reachable from the Internet and therefore vulnerable to vicious attacks [2]. On the other hand, as stated in Section 2.1, only the RDA has an IP address that is visible to the Internet, and all other nodes in the cluster bear only the private IP address. Therefore, all nodes are not reachable from the outside world. A firewall may be installed on the RDA node to protect the cluster. To attack one of the nodes, one has to first land on the RDA and launch an attack from there. If no user accounts are enabled and non-essential TCP/IP services are disabled on the RDA, leaving only the request distribution available, then breaking the RDA itself will be difficult. A Network Address Translation is used on the RDA to translate the destination IP address (based on request needs) of incoming packets to an internal private IP address, and that of the outgoing packets to the IP address on the Internet where the requests came from.

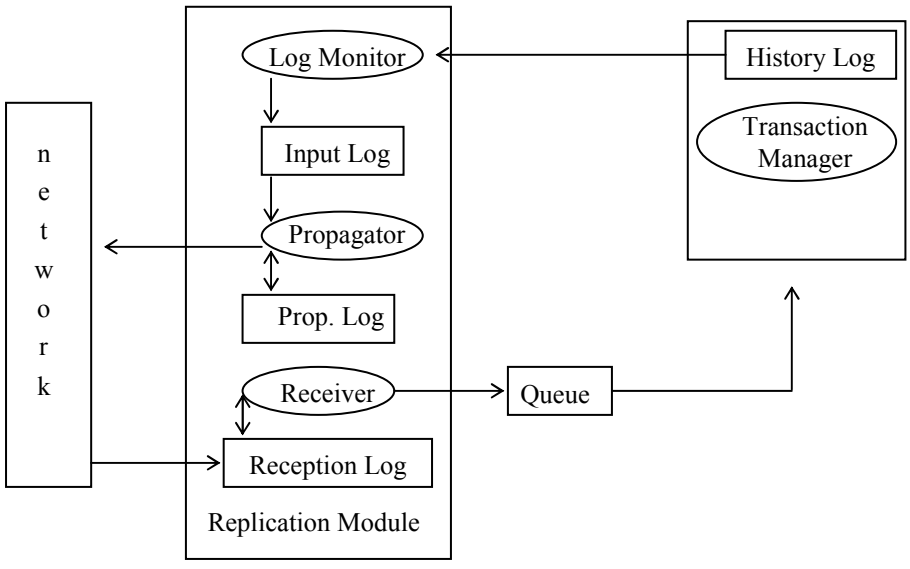

Fig. 3: Architecture of a node: square shapes represent data repositories and oval shapes represent system components

3.0 NEIGHBOR-REPLICA DISTRIBUTION TECHNIQUE (NRDT)

We assume that all nodes are logically organised in the form of two-dimensional n x n grid structure [12]. If there are nodes in the CS where $N = n^2$, then it will logically organise in the form of *n x n* grid.

For example, if a CS consists of sixteen nodes, it will logically be organise in the form of 4 x 4 grids as shown in Fig. 4. Each node has a master data file. In the remainder of this paper, we assume that replica copies are data files. Through this NRDT technique, updates on a primary copy are first committed at the primary node. Then each neighbor copy is updated asynchronously, in a separate transaction.

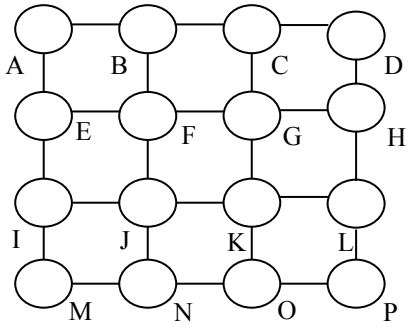

Fig. 4: A server cluster with 16 nodes

***Definition 3.1*:** A node X is a neighbor to node Y, if X is logically-located adjacent to Y.

A data will replicate to the neighboring nodes from its primary node. The number of data replication, m, can be calculated using Property 3.1, as described below.

Property 3.1: The number of data replication from each node, $m \leq 5$.

Proof: Let $N = n^2$ be a set of all nodes that are logically organised in a two-dimensional grid structure form. Then all nodes are labeled n(i,j), $1 \leq i \leq n$, $1 \leq j \leq n$. Two way links will connect nodes n(i,j) with its four neighbors, nodes n(i±1,j) and n(i,j±1), as long as there are nodes in the grid. (Note that, four nodes on the corners of the grid have only two adjacent nodes, and other nodes on the boundaries have only three neighbors). Thus, the number of neighbors of each node is less than or equal to 4. Since the data will be replicated to the neighbors, then the number of data replication from each node, m, is:

$m \leq$ the number of neighbors + a data from node itself
$= 4+1 = 5$.

For example, from Fig. 4, data from node A will replicate to node B and E, which are its neighbors. Node F has four neighbors, which are nodes B, E, G, and J. As such, node F has five replicas.

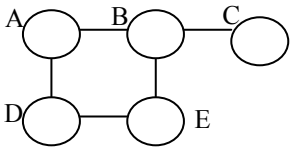

Fig. 5: A cluster- server with 5 nodes

For the case of $N \neq n^2$, at most there is only one node that has one neighbor. For example, if N = 5, it can be organised as shown in Fig. 5. It can be seen that only node C has one neighbor, that is node B, and other nodes has more than one neighbors. CS can still execute transactions successfully in the case of failure from any nodes, provided that at least one neighbor is still available. For example, from Fig. 4, if nodes A and B fail/unavailable and node E that is the neighbor of node A is available, and node C that is also the neighbor of node B is also available, then transactions that request for data in node A or node B can still be executed successfully by accessing required data from node E or node C respectively. This is due to the fact that, the master replicated data file from node A is available at node E, and the master replicated data file from node B is available at node C and node F.

4.0 UPDATE PROPAGATION STRATEGY

4.1 Immediate-Immediate Propagation

We consider the update propagation is based on the *immediate-immediate* strategy (for propagation and updating) as proposed in [4]. Each write operation performed by a transaction, T, is *immediate*ly propagated to neighboring nodes, without having to wait for the commitment of the original update transaction T. At a neighbor node, an update transaction is started as soon as the first write operation is received from the primary node, and terms as *immediate* updating. An update transaction T, is composed by the serial sequence of write operations. We assume that once transactions are submitted for execution to a local transaction manager at each node, all conflicts are handled by the local concurrency control protocol. With this *immediate-immediate* strategy, each time an operation is read, it is subsequently submitted to the local transaction manager. Here, the effect of the serial execution order of the update transactions performed at the primary is preserved (see [4] for more details). When an abort operation for a transaction T is read, the local transaction manager will abort the transaction.

4.2 Dealing With Node Failure

We assume that network omissions are bounded and taken into account by a multicast protocol. The recovery protocols are based on those used for transaction recovery proposed in [4].

4.2.1 Initialisation

To start update propagation towards a neighbor node, a primary node must first initiate a connection. When it is completed, the primary node can close the connection. The result of a connection or disconnection request is done using the connection table, which is described below. Connection and disconnection are requested using the following functions:

Connect (*primary_id, neighbor_id, replica_id*): *primary_id* requests a connection to *neighbor_id* in order to start update propagation on *replica_id*. A corresponding log record is generated and written in the *primary_id* Local History Log with the following information:

< "connect", primary_id, neighbor_id, replica_id>.

Disconnect (*primary_id, neighbor_id, replica_id*): *primary_id* requests a disconnection to *neighbor_id* in order to stop an update propagation on *replica_id*. A corresponding log record is generated and written in the *primary_id* Local History Log with the following information:

< "disconnect", primary_id, neighbor_id, replica_id>.

Each node *i* (primary or neighbor node) keeps control of its connection using a connection table. Each entry of this table corresponds to an established connection. The main attributes of this table are *node_id, replica_id and status*. *Node_id* identifies a node that is connected to node *i* and *replica_id* identifies the local replica copy involved in a specific connection. The *status* attribute indicates the current status of the connection whether a connection is *active* or *inactive*.

4.2.2 Recovery

We now present how primary and neighbor nodes recover after failure.

Primary failure

When a primary fails, all the connected neighbor nodes detect the failure through Receiver, which periodically checks for primary node availability using the network interface. As soon as a primary failure is detected, the neighbor *Receiver* writes a fail record of the form:

< "fail",primary_id>

in the correct pending queue. This record informs the local transaction manager of the failure of *primary_id*. When it recovers, it re-establishes its connection by propagating a Reconnect message towards each neighbor node it was connected to. When the neighbor node receives a Reconnect message, it stores its contents in the correct queue q. Thus, the reconnect record indicates the *primary_id* recovery. The receiver activities for *primary_id*.

Neighbor Failure

When a neighbor fails, all the connected primary nodes must stop sending messages to it. Failure detection is done by each primary Propagator using the network interface. It is based on periodically checking the neighbor as to whether it is up or not. A neighbor recovery is performed as follows: First, after recovery, the Receiver writes in each pending queue a *Reconnect* record of the form:

< *"reconnect", primary_id, neighbor_id, replica_id, message_id*>.

Message_id indicates the last message and is used to restart the master propagation and the neighbor updating activities. In the second step, the Receiver re-establishes its pending connections. It does so by searching in the Reception Log for all messages received from *master_id* but not yet processed, and stores each message in the correct queue, q.

5.0 PERFORMANCE ANALYSIS

In this section, we will analyse and compare the performance on the reliability of the two-replica [15] and our neighbor-replica distribution technique. Performance comparison between these two techniques will also be discussed. For simplicity, we will analyse the reliability for the case of N = 4, 5 and 6 only. Suppose that p is the probability that a node is *alive/available* in CS.

5.1 Two-Replica Distribution Technique (TRDT)

Suppose that all data have two replicas on different nodes and all nodes have two data replicas on CS for N≥2. All nodes can be divided into n different set with two nodes in each set. In such a case, the distribution reaches the highest reliability, R_N, to provide all data in a consistent state, thus

$$R_N = R_2^n = [1-(1-p)^2]^n .$$

For the case of N = 2n+1, then all nodes will be divided into (n-1) sets with two nodes and one set with three nodes. In such case, each node of (n-1) sets has two replicas and each node of another set has three replicas. Then, the highest reliability, R_N, given in [3] is;

$$R_N = R_3 R_2^{n-1} = p^2(3-2p)[1-(1-p)^2]^n$$

For example, if N=5, then

$$R_5 = p^2(3-2p)[1-(1-p)^2]^2 = p^3(6-7p+2p^2)$$

5.2 Neighbor-Replica Distribution Technique (NRDT)

In this technique, all data have some replicas on different nodes and all nodes have some data replicas under CS.

Definition 5.2.1: Let N be a number of nodes in the system. Each node has a master file. A set of nodes when the system cannot operate is called a fail set. A group of fail sets is called fail group, FG. Thus, from Fig. 5, when N=5;

$FG_{N=5} = \{\{a,b,d\},\{a,b,c,e\},\{b,c\},\{a,d,e\},\{b,d,e\}\}$

Definition 5.2.2: The degree of failure nodes, *deg*, is the number of nodes unavailable in the system.

Definition 5.2.3: Let $S_i \in FG_{N=5}$ for i = 1,2,…,5, and $R \in \varphi_{Si}^{deg}$, where φ_{Si}^{deg} is a fail group with degree, *deg*, such that $S_i \subseteq R$.

For example, from Fig. 5, let $S_1 = \{a,b,d\} \in FG_{N=5}$, and deg = 4, then

$$\varphi_{S_i}^{4} = \{\{a,b,c,d\}, \{a,b,d,e\}\}$$

The probability of φ_{Si}^{deg} is given by, ϕ_{Si}^{deg} such that

$$\phi_{Si}^{\deg} = \begin{pmatrix} 5-|Si| \\ \deg-|Si| \end{pmatrix} (1-p)^{\deg} p^{5-\deg}.$$

As in Fig. 4, with S_1 = {a,b,d} and *deg* = 4, then

$$\phi_{\{a,b,d\}}^{4} = \begin{pmatrix} 5-3 \\ 4-3 \end{pmatrix} (1-p)^{4} p^{5-4} = 2P(1\text{-}P)^4$$

Definition 5.2.4: Let $\varphi_{Si}^{\deg}$ and $\varphi_{Sj}^{\deg}$, i≠j, be the two fail groups with the common degree, *deg*. We define that the probability of $\varphi_{Si}^{\deg} \cap \varphi_{Sj}^{\deg} = \phi_{Si \cap Sj}^{\deg}$.

Thus,

$$\varphi_{S1}^{\deg} \cap \varphi_{S2}^{\deg} \cdots \varphi_{SN}^{\deg} = \phi_{Si \cap Sj \cap \ldots \cap SN}^{\deg}.$$

The reliability, R_N = {probability that the system with N nodes, does not experience any failures in a given time interval}

= 1- {probability that the system does experience any failures in the given time interval}

$$= 1 - \sum_{\deg=1} \cup \phi_{Si}^{\deg} \tag{1}$$

where,

$$\cup \phi_{Si}^{\deg} = \sum_{i} \phi_{Si}^{\deg} - \sum_{i<j} \phi_{Si \cap Sj}^{\deg} + \sum_{i<j<k} \phi_{Si \cap Sj \cap Sk}^{\deg} - \ldots + (-1)^{N+1} \phi_{S1 \cap S2 \cap \ldots \cap SN}^{\deg} \tag{2}$$

For the case of N=4, from equations 1 and 2, the reliability, R_4, can be represented as,

$$R_4 = 1 - 4p(1-p)^3 - (1-p)^4 = P^2(6 - 8P + 3P^2), \tag{3}$$

while for the case of N =5, the reliability, R_5, can be represented as,

$$R_5 = 1 - [(1\text{-}P)^2 P^3 + 6(1\text{-}P)^3 P^2 + 5(1\text{-}P)^4 P + (1\text{-}P)^5]$$

$$= P^2(4 - 3P - P^2 + P^3), \tag{4}$$

and for the case of N =6, the reliability, R_6, can be represented as,

$$R_6 = 1 - [4P^3(1\text{-}P)^3 + 12P^2(1\text{-}P)^4 + 6P(1\text{-}P)^5 - (1\text{-}P)^6]$$
$$= P(3P + 4P^2 - 15P^3 + 12P^4 - 3P^5) \tag{5}$$

5.3 The Correctness

Definition 5.3: The reliability, R_N, under CS of N nodes is in a closed form if $0 \leq R_N \leq 1$, for $0 \leq p \leq 1$.

Proposition 5.3*:* The reliability under NRDT with N nodes is in the closed form.

***Proof*:** Equation 3 is analogous to the Inclusion and Exclusion method [17], where $\sum_{deg=1} \cup \phi_{Si}^{deg} \leq 1$.

Thus, $0 \leq R_N \leq 1$ as $0 \leq p \leq 1$.

Table 1: The reliability of SC for TRDT and NRDT techniques for N=4, 5 and 6

Technique		Probability of a Node Available (*P*)								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TRDT	N=4	0.036	0.130	0.260	0.410	0.563	0.706	0.828	0.922	0.980
	N=5	0.005	0.037	0.110	0.225	0.375	0.544	0.713	0.860	0.962
	N=6	0.007	0.047	0.133	0.262	0.422	0.593	0.754	0.883	0.970
NRDT	N=4	0.052	0.181	0.348	0.525	0.688	0.821	0.916	0.973	0.996
	N=5	0.0369	0.135	0.273	0.433	0.594	0.740	0.859	0.942	0.987
	N=6	0.026	0.132	0.283	0.463	0.641	0.793	0.904	0.970	0.996

5.4 Performance Comparison

Table 1 and Fig. 6 show the reliability of SC for TRDT and NRDT techniques for the number of nodes, N=4, 5, and 6. It is shown that the reliability under NRDT technique is better than that the reliability under TRDT technique. It can be seen that NRDT technique needs only the probability of a node alive, $P = 0.7$, while TRDT needs $P > 0.8$ in order to maintain $R_N > 0.9$. For example, when the probability of a node being available is 70%, the reliability for TRDT is approximately 83% whereas the reliability for NRDT is approximately 92%. It is more than 10% better than that of TRDT when N =4. The reliability for TRDT is approximately 75% when N =6 whereas the reliability for NRDT is approximately 90%. This is more that 20% better than that of TRDT.

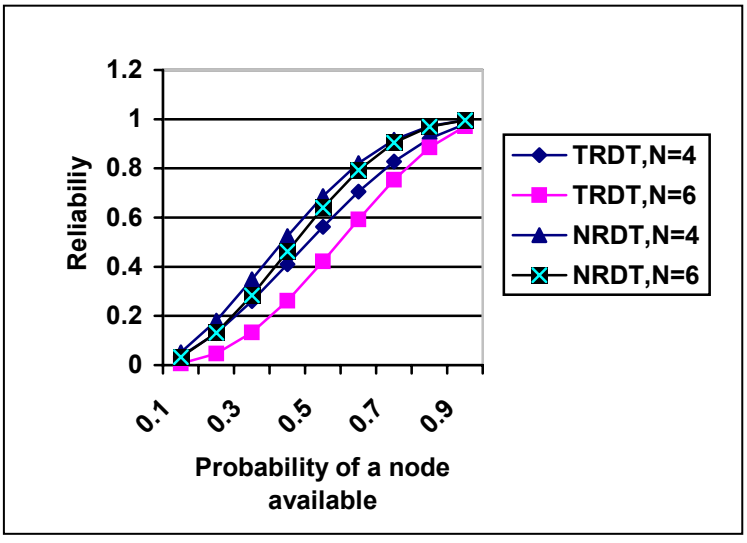

Fig. 6: Comparison of the reliability between TRDT and NRDT for N =4 and 6

6.0 CONCLUSION

In this paper, a new technique called Neighbor-Replica Distribution Technique (NRDT) with an *immediate-immediate* propagation strategy was proposed for improving the reliability of the cluster CS systems. The analysis of the NRDT technique was presented in terms of reliability and was compared to Shen's TRDT. It was shown that the NRDT technique provides a convenient approach to high reliability of data replication for CS systems. Thus, this technique represents an alternative design philosophy to the data replica distribution technique in the CS systems.

REFERENCES

[1] R. Short, R. Credle, and J. Pelles, *Clustering and High Availability Guide for IBM Netfinity and IBM PC Servers*. 1997.

[2] J. Liu, L. Xu, B. Gu, J. Zhang, "A Scalable, High Performance Internet Cluster Server". *IEEE 4th Int'l Conf. HPC-ASIA*, Beijing, 2000, pp. 941-944.

[3] M. Mat Deris, A. Mamat, P. C. Seng, H. Ibrahim, "Three Dimensional Grid Structure for Efficient Access of Replicated Data". *Int'l Journal of Interconnection Networks, World Scientific*, Vol. 2, No. 3, 2001, pp. 317-329.

[4] E. Pacitti, E. Simon, "Update Propagation Strategies to Improve Fresin Lazy Master Replicated Databases". *Journal VLDB*, Vol. 8, No. 4, 2000.

[5] M. Buretta, *Data Replication: Tools and Techniques for Managing Distributed Information*. John Wiley, New York, 1997.

[6] J. Tang and N. Natarajan, "Obtain Coteries That Optimizing The Availability of Replicated Databases". *IEEE Trans*. *Knowledge Data Eng.,* 5, 1993, 309-321.

[7] W. Zhou and A. Goscinski, "Managing Replicated Remote Procedure Call Transactions". *The Computer Journal*, Vol. 42, No. 7, 1999, pp. 592-608.

[8] M. Ahmad, M. H. Ammar, "Performance Characterization of Quorum Consensus Algorithm for Replicated Data". *IEEE Trans. Soft. Eng*, 15, 1989, 492-296.

[9] Y. Amir and A. Wool, "Optimal Availability Quorum Systems: Theory and Practice". *Information Process Lett*. 65, 1998, 223-228.

[10] H. K. Chang and S. M. Yuan, "Optimal Binary Vote Assignment for Replicated Data". *The Journal of Systems and Software, Elsevier*, Vol. 53, 2000, pp. 73-82.

[11] Z. Tong and R. Y. Kain, "Vote Assignment in Weighted Voting Mechanisms". *IEEE Trans. Comput*., Vol. 40, 1991, pp. 664-667.

[12] D Agrawal and A. El Abbadi, "Using Reconfiguration for Efficient Management of Replicated Data". *IEEE Trans. On Knowledge and Data Engineering*, Vol. 8, No. 5, 1996, pp. 786-801.

[13] A. A. Helal, A. A. Heddaya, and B. B. Bhargava, *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, Norwel, Massachusetts, 1996.

[14] A. Moissis, *Sybase Replication Server: A practical Architecture for Distributing and Sharinf Information*. Technical Document, Sybase Inc, 1996.

[15] H. H. Shen, S. M. Chen, M. M. Shen, and W. M. Zheng, "Research on Data Replica Distribution Technique for Server Cluster". *IEEE Proc. 4th Int'l. Conference on Peformance Computing*, Beijing, 2000, pp. 966-968.

[16] H. H. Shen, S. M. Chen, W. M. Zheng, and S. M. Shi, "A Communication Model for Data Availability on Server Clusters". *Proc. Int'l. Symposium on Distributed Computing and Application*, Wuhan, 2001, pp. 169-171.

[17] S. M. Ross, *Introduction to Probability Model*. 5th Ed. Academic Press, 1993.

BIOGRAPHY

Rabiei Mamat received the M.Sc. degree in Computer Science from University College of Science and Technology Malaysia (KUSTEM) in 2004. Currently he is a lecturer at the Department of Computer Science, University College of Science and Technology Malaysia (KUSTEM). His research interests include cluster server management, networking, and computer systems.

Mustafa Mat Deris received the M.Sc. degree in Computing from University of Bradford in 1989. He obtained his PhD in Computer Science from University Putra Malaysia, in 2001. Currently he is the head of the Department of Computer Science, University College of Science and Technology Malaysia (KUSTEM). He is a member of IEEE and also a technical committee member of IASTED on Databases. His research interests include distributed databases, cluster server management, computer system performance and data mining.

Mashita Jalil received M.Sc in Engineering Business Management from Warwick, 2000. Currently she is a lecturer at the Department of Computer Science, KUSTEM. Her research interests include Computer Systems and Knowledge Management.