Proceedings of the 5th WSEAS Int. Conf. on DISTANCE LEARNING AND WEB ENGINEERING, Corfu, Greece, August 23-25, 2005 (pp205-211)

Data Replication Model For Remote Procedure Call Transactions

MUSTAFA MAT DERIS, ALI MAMAT*, MISWAN SURIP, SAZALI KHALID

Department of Information Systems,
Faculty of Information Technology and Multimedia
College University Technology Tun Hussein Onn,
86400 Parit Raja, Batu Pahat, Johor
MALAYSIA
*Faculty of Computer Science and Information Technology
University Putra Malaysia,
Serdang, 43000, Selangor
MALAYSIA

Abstract

Remote Procedure Call (RPC) is the most popular model to facilitate the building of distributed programs. However, it only provides a restricted availability of update operations and does not support fault-tolerant. The combination of RPC and data replication techniques has been developed to support fault-tolerant from Remote Procedure Call (RPC). However, different data replication techniques provide a different availability and fault-tolerant. This paper proposes a data replication technique called Three Dimensional Grid Structure (TDGS) to form a high availability and fault-tolerant of RPC in distributed computing environment. Fault-tolerant programs developed in this environment are able to tolerate failures such as server failure, site failure or even network partitioning. In order to obtain the proposed programs, firstly we generalize the Three Dimensional Grid Structure protocol to provide high availability of read and write operations. Secondly, the replica management to support the development of reliable services in a replicated-server environment is established.

Keywords : *Distributed Database, Data Availability, Replication, Data Consistency, Remote Procedure Call, Transaction Management.*

1. Introduction

Many commercial database systems such as *Oracle 8* and *IBM DB2 Propagator* provide the required support for data distribution and inter-database communication [1]. However, one of the major issues in data distribution is replicated data management. Typical replicated data management parameters are data availability and communication costs: the higher the data availability with lower communication costs the better the system is. In other words, replication is a useful technique to provide high availability, fault tolerance, and enhanced performance for distributed database systems [2,3,4,5] where an object will be accessed (i.e., read and written) from multiple locations such as from a local area network environment or geographically distributed over a possible large region, a country, or even the whole world. For example, student's results at a college will be read and updated by lecturers of various departments. Financial instruments' prices will be read and updated from all over the world [6].

Transaction Management is a well established concept in database system research [7]. A transaction can be defined as a sequence of operations over an object system, and all operations must be performed in such a way that either all of them execute or none of them do [8,9]. Transactions are used to provide reliable computing systems and a mechanism that simplifies the understanding and reasoning about programs.

A Remote Procedure Call (RPC) model is the most popular model used in today's distributed software development and has become a *de facto* standard for distributed computing [9]. However, it only provides a restricted availability of update operations and does not support fault-tolerant. Since fault tolerance is not provided in the RPC level, system services and user applications have to employ their own mechanisms in dealing with reliability and availability of the system. This limitation has resulted in a number of problems such as (1) adding another dimension of difficulties in software development; and (2) repeated development of fault-tolerant mechanisms in every services and applications. For example, the directory service of the distributed computing environment uses a primary copy and a number of read-only copies to provide a distributed and replicated repository for information on various resources of a distributed system. This mechanism has the inconsistency and reconfiguration problems in the case of Failures.

An outstanding issue in supporting for fault-tolerant from the RPC has been done with a combination of remote procedure, transaction management, and replication proposed in [7]. However, the replication protocol being used is analogous to Read-One Write All (ROWA) protocol [6,8]. If one site is not accessible, the processing of an object is noted in the *partial commit* state, and resolved it after some time delay. This will increase the response time (one of the major performance parameter [8]), and therefore decreases the performance of the system. For the case of availability, RPC provides restricted availability of update operations since they cannot be executed (normal state i.e., commit or abort) at the failure of any copy.

In this paper, firstly, a new protocol called Three Dimensional Grid Structure (TDGS) protocol is proposed in order to provide high data availability of read and write operations. Secondly, replica management is presented. This paper will discuss only cases where the number of replicas , $n \geq 8$. The proposed model is called Three Dimensional Grid Structure Remote Procedure Call (TDGS-RPC) model.

The remainder of the paper is organized as follows. Section 2 presents the model and the generalized TDGS. Section 3 describes the replica management. Section 4 concludes the paper.

2 System Model

A distributed system with replicated servers consists of many sites interconnected by a communication network. A site may become inaccessible due to site or partitioning failure. No assumptions are made regarding the speed or reliability of the network. We assume that sites are fail stop and communication links may fail to deliver messages. Combinations of such failures may lead to partitioning failures [6] where sites in a partition may communicate with each other, but no communication can occur between sites in different partitions.

A distributed database consists of a set of objects stored at different sites in a computer network. Users interact with the database by invoking transactions, which are partially ordered sequences of atomic read and write operations. The execution of a transaction must appear atomic: a transaction either commits or aborts [3,5].

In a replicated database, copies of an object may be stored at several sites in the network. Multiple copies of an object must appear as a single logical object to the transactions. This is termed as one-copy equivalence and is enforced by the replica control protocol. The correctness criteria for replicated database is one-copy serializability [8], which ensures both one-copy equivalence and the serializable execution of transactions. In order to ensure one-copy serializability, a replicated object may be read by reading a quorum of copies, and it may be written by writing a quorum of copies. The quorum for an operation is defined as a set of copies whose number is sufficient to execute that operation. Since each site stores a copy of a data object, thus a quorum of copies is equivalent to the number of copies in the quorum. In what follows, the terms a number of copies and a number of sites will be used interchangeably. The selection of a quorum is restricted by the quorum intersection property to ensure one-copy equivalence: For any two operations o[x] and o'[x] on an object x, where at least one of them is a write, the quorum must have a non-empty intersection.

Briefly, a site S initiates a TDGS transaction to update its object. For all accessible objects, a TDGS transaction attempts to access a TDGS quorum. If a TDGS transaction gets a TDGS write quorum without non-empty intersection, it is accepted for execution and completion, otherwise it is rejected. We do not need to worry about the read quorum if two transactions attempt to read a common object, because read operations do not change the values of the object. Since read and write quorums must intersect and any two TDGS quorums must also intersect, then all transaction executions are one-copy serializable.

2.1 The Generalized TDGS Protocol

The TDGS protocol has been proposed in [9] to define quorums for both read and write operations in the system. With this protocol, it provides high availability with low communication cost when compared with other protocols, such as Tree quorum and Grid structure protocols. However, this protocol only discussed with 24 copies of an object to define quorums for both read and write operations in the system. In this paper we generalized the TDGS protocol by considering N copies of an object in the system. With this protocol, copies are logically organized into a box-shape structure with four planes. Each copy is located based on the coordinate (x,y,z) belongs to them.

Fig. 1, shows a box-shape structure that consists of four planes (α_1,α_2,α_3, and $\alpha 4$) with the small circle representing a copy at location $C_{0,0,0}$, $C_{0,0,1}$, …, $C_{l\text{-}1,l\text{-}1,l\text{-}1}$. If the numbers of copies in each plane are equal, then the box-shape structure is in perfect square.

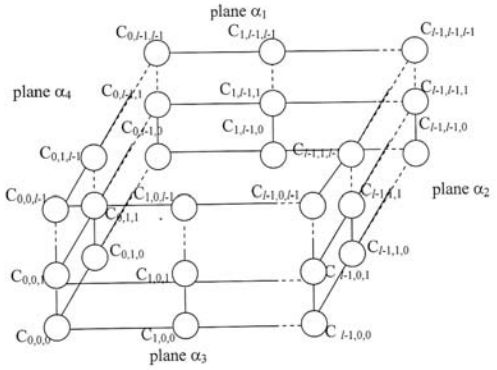

Fig. 1: A TDGS organization with n copies of a data object

Definition 2.1.1. A pair of copies that can be constructed from a hypotenuse edge in a box-shape structure (organization) is called hypotenuse copies.

For a TDGS quorum, read operations on an object are executed by acquiring a read quorum that consists of any hypotenuse copies. In Fig.1, copies $\{C_{0,0,0},C_{l-1,l-1,l-1}\}$, $\{C_{0,0,l-1},C_{l-1,l-1,0}\}$, $\{C_{0,l-1,l-1},C_{l-1,0,0}\}$, or $\{C_{l-1,0,l-1},C_{0,l-1,0}\}$ are hypotenuse copies from which is sufficient to execute a read operation. Since each pair of them is hypotenuse copies, it is clear that, read operation can be executed if one of them is accessible, thus increasing the fault-tolerance of this protocol.

Write operations on the other hand, are executed by acquiring a write quorum from any plane that consists of hypotenuse copies, and all copies which are vertices. For example, if the hypotenuse copies, say $\{C_{0,0,0},C_{l-1,l-1,l-1}\}$ are required to execute a read operation, then copies $\{C_{0,0,0},C_{l-1,l-1,l-1},C_{l-1,l-1,0},C_{0,l-1,l-1},C_{0,l-1,0}\}$ are sufficient to execute a write operation, since one possible set of copies of vertices that correspond to $\{C_{0,0,0},C_{l-1,l-1,l-1}\}$ is $\{C_{l-1,l-1,l-1},C_{l-1,l-1,0},C_{0,l-1,l-1},C_{0,l-1,0}\}$. Other possible write quorums are $\{C_{0,0,0},C_{l-1,l-1,l-1},C_{l-1,l-1,0},C_{l-1,0,l-1},C_{l-1,0,0}\}$,$\{C_{l-1,l-1,l-1},C_{0,0,0},C_{0,0,l-1},C_{l-1,0,l-1},C_{l-1,0,0}\}$,$\{C_{l-1,l-1,l-1},C_{0,0,0}, C_{0,0,l-1},C_{0,l-1,l-1},C_{0,l-1,0}\}$, etc. It can be easily shown that a write quorum intersect with both read and write quorums in this protocol (Section 6.1.).

In addition, TDGS allows us to construct a write quorum even though three out of four planes are unavailable as long as the hypotenuse copies are accessible. In other words, this protocol tolerates the Failure of more than three quarter of the copies in the TDGS protocol. Consider the case when only one plane which consists of four copies of vertices and hypotenuse copies are available, e.g., the set $\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1},C_{l-1,0,l-1}, C_{l-1,0,0}\}$ is available as shown in Fig.1. A TDGS transaction can be executed successfully by accessing those copies in a TDGS quorum. Hence, accessing those available copies forms the write quorum. Read operations, on the other hand, need to access the available hypotenuse copies. Thus the proposed protocol enhances the fault-tolerance in write operations compared to the grid configuration protocol.

Therefore, this protocol ensures that read operations have a significantly lower cost, i.e., two copies, and have a high degree of availability, since they are not vulnerable to the Failure of more than three quarter of the copies. Write operations, on the other hand, are more available than the grid configuration protocol since only five copies are needed to execute write operations.

2.2 Performance Analysis and Comparison

In this section, we present the communication cost and the availability of ROWA and TDGS protocols for update operations. The communication cost of an operation is directly proportional to the size of the quorum required to execute the operation. Therefore, we represent the communication cost in terms of the quorum size. $C_{X,Y}$ denotes the communication cost with X protocol. In estimating the availability of operations, all replicas are assumed to have the same availability *p*, and A_X will represent the availability of update operation with X protocol.

2.2.1 ROWA

An update operation needs to access n replicas if there are n replicas in the system. Thus, the communication cost of an update operation, $C_{ROWA} = n$ … (1)

This protocol requires an update to all replicas. Thus, the availability for an update operation A_{ROWA} is:

$$A_{ROWA} = \sum_{j=n}^{n} \binom{n}{j} p^{j} (1-p)^{n-j} = p^{n} \quad \cdots (2)$$

2.2.2 TDGS Protocol

The cost of a write operation , $C_{TDGS,W}$, can be represented as:

hypotenuse copies +
(all copies of vertices in a plane - hypotenuse copy in the same plane).
= 2 + (4-1) = 5.

For example, if hypotenuse copies is $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$, then all copies of vertices in plane α_1 that correspond to $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$ is $\{ C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$. Therefore,

$$C_{TDGS} = |\{C_{0,0,0}, C_{l-1,l-1,l-1}\}|+|\{ C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}|-|\{C_{l-1,l-1,l-1}\}| = 2+(4-1) = 5. \quad \ldots (3)$$

On the contrary, a write quorum can be constructed as follows: Let $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4\}$ be a set of planes in the TDGS protocol as shown in Fig.1. Let $i=\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$, be the hypotenuse copies, then write availability that consists of hypotenuse copies i, W_i, can be represented as:

Probability{ $C_{0,0,0}$ is available}* [φ available]
+ Probability{ $C_{l-1,l-1,l-1}$ is available}* [ϕ available]
- Probability{ $C_{l-1,l-1,l-1}$ and $C_{0,0,0}$ are available}*[(φ and ϕ) are available] … (4)

where, $\varphi = \Omega(\alpha_1) + \Omega(\alpha_2) - \Omega(\alpha_1 \cap \alpha_2)$,
$\phi = \Omega(\alpha_3) + \Omega(\alpha_4) - \Omega(\alpha_3 \cap \alpha_4)$,
and $\Omega(\alpha_i)$ = Probability of plane α_i available.

Without loss of generality, we assume that a non-vertex copy, say $C_{1,l-1,l-1} \in \alpha_1$ is a primary copy. The probability of α_1 available, $\Omega(\alpha_1)$, can be represented as:

Probability{ all copies of vertices from α_1 and primary copy are available} + Probability{ (all copies of vertices and primary copy + 1 copy) from α_1 are available} + …+ Probability { all copies from α_1 are available}

$$= p^{5} \sum_{j=0}^{m-5} \binom{m-5}{j} p^{j} (1-p)^{m-5-j} = p^{5}$$

… (5)

However, the probability of α_i , i =2,3,4 available, $\Omega(\alpha_i)$, can be represented as:
Probability{ all copies of vertices from α_i are available}
+ Probability{ (all copies of vertices i + 1 replica) from α_i are available} + …+ Probability { all copies from α_i are available}

$$= p^{4} \sum_{j=0}^{m-4} \binom{m-4}{j} p^{j} (1-p)^{m-4-j} = p^{4}$$

… (6)

where m is a number of copies in each plane. Thus $\Omega(\alpha_1) = p^5$, and $\Omega(\alpha_i) = p^4$, for i=2,3,4.

Since, the probability {V is available } = probability {C is available} = *p*, then , equation (2.4), W_i is:

$$= p\,\varphi + p\,\phi - p^{2}(\varphi * \phi) \quad \ldots (7)$$

Similarly, with the write availability that consists of other hypotenuse copies, thus $W_i = W_j = W_k = W_l$, where $i \in \mathbf{R}$. To compute the write availability, $A_{TDGS,W}$, let $W_i = \beta$, $i \in \mathbf{R}$,then $A_{TDGS,W}$ is:

$$= \binom{4}{1} \text{x } W_i - \binom{4}{2} \text{x } (W_i \cap W_j) + \binom{4}{3} x (W_i \cap W_j \cap W_k)$$

$$- W_i \cap W_j \cap W_k \cap W_l, \quad \text{for } i,j,k,l \in \mathbf{R}$$

$$= 1-(1-\beta)^4 \qquad \ldots (8)$$

2.2.3 Comparison of Costs and Availabilities

We assume that the distributed system has n copies. From equations (1) and (3), it is apparent that TDGS has the lowest cost for write operation in spite of having a bigger number of copies when compared with ROWA. It can be seen that, TDGS needs only 5 copies (6 copies if the primary copy is not an hypotenuse copy) in all instances, while the communication costs for ROWA needs n copies. Thus, the TDGS has a better performance in terms of communication costs when compared to ROWA.

The write availabilities of TDGS and ROWA protocols are compared in Fig.2. We assume that all copies have the same availability, and N = {10 ,16}.

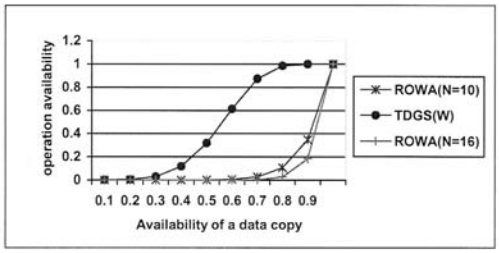

Fig.2. Comparison of the write availability between TDGS and ROWA for N=10 and 16.

Fig. 2. shows the TDGS protocol has better performance of availabilities when compared to the ROWA protocol. For example, when an individual copy has availability of 70%, the write availability in the TDGS is more than 87%, whereas the write availability in the ROWA is approximately 3%. Moreover, write availability in the ROWA decreases as N increases. For example, when an individual copy has availability 90%, write availability is approximately 35% for N=10 whereas write availability is approximately 19% for N = 16. While the write availability of TDGS is the same due to the same number of write quorum required for all instances.

3. Replica Management

We define the *primary replica* for a data object *d* as a replica that is the nearest site in performing the three dimensional grid structure remote procedure call (TDGS-RPC). The TDGS-RPC and transaction processing models are analogous with the models proposed in [7]. However, coordinating algorithm of the primary replica will be changed by adopting the TDGS protocol for read or update operations in the system.

3.1 The Coordinating Algorithm For The Primary Replica

When a primary replica receives an TDGS-RPC from a transaction manager, it uses the coordinating algorithm to maintain the consistency of all replicas in terms of TDGS-RPC. This section describes the coordinating algorithm.

In the coordinating algorithm, the primary replica uses the 2PC protocol to ensure the replication consistency. In the first phase, the primary replica asks the TDGS quorum whether it can be formed or not. If the TDGS quorum can be formed (replicas under TDGS quorum return a successful for such execution), the primary replica returns a successful (SUC) to the transaction manager. If the transaction manager requests a commit, then in the second phase the primary replica asks all replicas to commit the TDGS-RPC execution. If TDGS quorum cannot be formed, then the primary replica returns a fail (FL) to the transaction manager

and asks all replicas to abort the operation in the second phase. If operation returns a partial commit (PC), the primary replica returns a PC to the transaction manager and other non-primary replicas may return a PC. The primary replica also records the number of replicas that return a PC. This number will be used in conflict resolution during the recovery process.

4. Conclusion

A system for building reliable computing over TDGS-RPC system has been described in this paper. The system combines the replication technique and embeds it into the TDGS-RPC system. It describes the models for TDGS, replica management, and the algorithms for coordinating the primary replica. It shows that the system supports the development of reliable services in the TDGS-RPC level. Fault-tolerant programs developed in our environment are able to tolerate failures such as server failure, a site failure or even a network partitioning. Furthermore, the availability of the TDGS-RPC outperformed the availability with RPC given in [7]. Thus, this system represents an alternative design philosophy to the remote procedure call model.

References

[1] M.T. Ozsu and P. Valduriez,"*Principles of Distributed Database Systems*", 2nd Ed., Prentice Hall, 1999.

[2] B. Bhargava,"Concurrency Control in Database Systems," *IEEE Trans. Knowledge and Data Engineering*,vol 11,no. 1, pp.3-16,1999.

[3] S.K. Madria, M.Mohania, S.S. Bhowmick, B. Bhargava,"Mobile Data and Transaction management", *Journal of Information Sciences, Elsevier*, Vol. 141, pp. 279-309, 2002.

[4] D Agrawal and A.El Abbadi, "The generalized Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data,"*ACM Trans. Database Systems*,vol.17,no. 4, pp. 689-717, 1992.

[5] S.Jajodia and D. Mutchles,"Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database,"*ACM Trans. Database Systems*,vol 15,no. 2, pp. 230-280, 1990.

[6] S.Y. Cheung, M.H. Ammar, and M. Ahmad,"The Grid Protocol: A High Performance Schema for Maintaining Replicated Data," *IEEE Trans. Knowledge and Data Engineering*,vol. 4, no. 6,pp. 582-592, 1992.

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

[8] P.A. Bernstein and N.Goodman, "An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases,"*ACM Trans. Database Systems*,vol 9,no. 4, pp.596-615,1994.

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