2011 International Conference on Computer Communication and Management
Proc .of CSIT vol.5 (2011) © (2011) IACSIT Press, Singapore

A New Binary Vote Assignment on Grid Algorithm to Manage Replication in Distributed Database Environment

Ainul Azila Che Fauzi, A.Noraziah, Noriyani Mohd Zain and A.H.Beg

Faculty of Computer Systems & Software Engineering
University Malaysia Pahang
Pahang, Malaysia
e-mail: ainulazila@yahoo.com

Abstract. Data replication is one of the mechanisms in data grid architecture since it improves data access and reliability. Therefore, the storage, availability, and consistency are important issues to be addressed in order to allow distributed users efficiently and safely access data from many different sites. This paper presents a new algorithm namely Binary Vote Assignment on Grid techniques to manage transaction and replication for distributed database systems. The main objective of this technique is to preserve consistency in database environment. Result shows that managing replication and transaction through proposed algorithm able to prepare data consistency.

Keywords: Data replication, Replication algorithm, Synchronous, BVAG, data grid.

1. Introduction

Organizations need to provide current data to users who may be geographically remote and to handle a volume of requests of data distributed around multiple sites in distributed environment. Therefore, the storage, availability, and consistency are important issues to be addressed in order to allow distributed users efficiently and safely access data from many different sites [1]. One way to provide access to such data is through replication. Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system [2]. A distributed database is distributed into separate partitions/fragments. Each partition/fragment of a distributed database may be replicated (ie. redundant fail-overs, RAID like) [1]. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations. Replication provides user with fast, local access to shared data, and protects availability of applications because alternate data access options exist. Even if one site becomes unavailable, users can continue to query or even update the remaining locations. Synchronous replication can be categorized into several schemes, i.e., all data to all sites (full replication), all data to some sites and some data to all sites. Expensive synchronization mechanisms are needed in order to maintain the consistency and integrity of the replicated data in distributed environment. One of the simplest techniques is Read-One-Write-All (ROWA) technique [3]. This technique has been proposed for managing data in mobile and peer-to-peer (P2P) environment [4]. Voting techniques [5] became popular because they are flexible and are easily implemented. This technique has been applied to the front-end clusters for managing replicated data [6]. Tree quorum (TQ) [7] uses quorums that are obtained from a logical tree structure imposed on data copies. TQ has been proposed for persistent consistent distributed database commit in a dynamic a synchronous network, peer-to-peer network [8].

2. Related Work

+ Ainul Azila Che Fauzi. Tel.: +6095492121; fax: +6095492144
E-mail address: ainulazila@yahoo.com

2.1. Replication Concept

Replication is the process of sharing information to ensure consistency between redundant resources such as software or hardware components. This process helps to improve reliability, fault-tolerance, or accessibility of data [9, 12]. Data replication may occur if the same data is stored in multiple storage devices. Meanwhile, computation replication occurs when the same computing task is executed many times. A computational task is typically replicated in space, i.e. executed on separate devices, or it could be replicated in time, if it is executed repeatedly on a single device. Whether one replicates data or computation, the objective is to have some group of processes that handle incoming events. If we replicate data, these processes are passive and operate only to maintain the stored data, reply to read requests, and apply updates. When we replicate computation, the usual goal is to provide fault-tolerance. For example, a replicated service might be used to control a telephone switch, with the objective of ensuring that even if the primary controller fails, the backup can take over its functions [8].

There are two types of data replication namely, synchronous replication and asynchronous replication. Asynchronous replication usually transmitted inconsistently rather than a steady stream. Asynchronous replication also caused the receiver to have problems in receiving the data from the sender. Many vendors like Lotus Notes adopted asynchronous replication as a solution for managing replicated data because asynchronous replication works reasonably well for single object updates. However, asynchronous replication fails when involving multiple objects with single update.

Therefore, synchronous replication is the answer for constraints that the asynchronous brought. Synchronous replication will guarantee data consistency since it works based on quorum to execute the operations. Plus, synchronous replication provides a 'tight consistency' between data stores [12]. For any copy that has been updated, the updates are applied immediately to all the copies within the same transaction [12]. This will ensure that all the copies in any site are the same and consistent. A consistent copy in all sites give advantages to the organization as it provides an updated data that is accessible anytime at any place in the distributed systems environment. However, synchronous replication require vast amount of storage capacity as multiple copies of replicated data stored in many sites and expensive synchronization mechanism are needed to maintain the consistency of data when changes are made. As a result, a proper strategy is needed to manage the replicated data in distributed systems environment [11].

3. BVAG Algorithm

3.1. Binary Vote Assignment on Grid Model

Binary Vote Assignment Grid (BVAG) technique will be used to approach the research. In BVAG, all sites are logically organized in the form of two-dimensional grid structure. For example, if a BVAG consists of twenty-five sites, it will logically organize in the form of 5 x 5 grids as shown in Figure 1. Each site has a premier data file. In the remainder of this paper, they assume that replica copies are data files. A site is either operational or failed and the state (operational or failed) of each site is statistically independent to the others. When a site is operational, the copy at the site is available; otherwise it is unavailable [10].

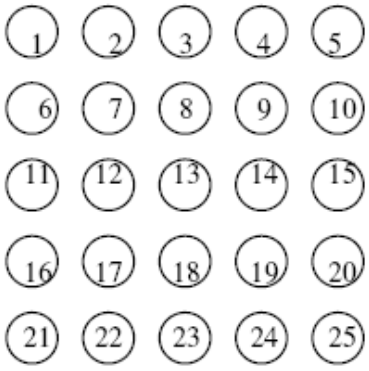

Fig. 1: Sites logically organized in two-dimensional grid structure

For this research, 9 sites will be used which will be logically organize in the form of 3 x 3 grids as shown in Figure 2.

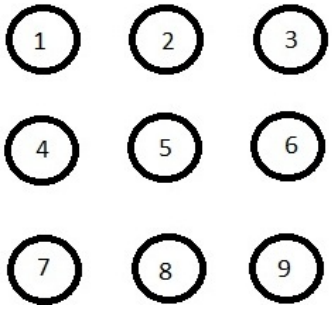

Fig. 2**:** BVAG sites.

A data will replicate to the neighbouring sites from its primary site. Four sites on the corners of the grid have only two adjacent sites, and other sites on the boundaries have only three neighbours. Thus, the number of neighbours of each sites is less than or equal to 4. Refer to Figure 3, data from site 1 will replicate to site 2 and 3 which are its neighbours. Site 5 has four neighbours, which are sites 2, 4, 6 and 8. So, site 5 has five replicas. Meanwhile, site 6 replicates to site 3, 5 and 9.

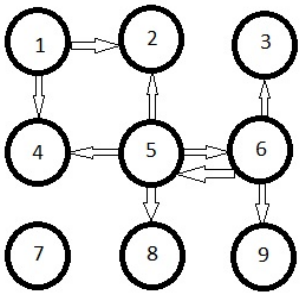

Fig. 3: All replicas updated.

3.2. Binary Vote Assignment on Grid Algorithm

The following notations are defined:

a) V is a transaction.
b) S is relation in database.
c) S_i is vertical fragmented relation derived from S, where i = 1,2,...,n.
d) P_k is a primary key
e) x is an instant in T which will be modified by element of V.
f) T is a tuple in fragmented S.
g) ${}^{Si}_{Pkxx}$ is a horizontal fragmentation relation derived from S_i.
h) P_i is an attribute in S where i = 1,2,...,n.
i) $M_{i,j}$ is an instant in relation S where i and j = 1,2,...,n.
j) i represent a row in S.
k) j represent a column in S.
l) η and ψ are groups for the transaction V.
m) $\gamma = \alpha$ or β where it represents different group for the transaction V (before and until get quorum).
n) V_η is a set of transactions that comes before V_ψ, while V_ψ is a set of transactions that comes after V_η.
o) D is the union of all data objects managed by all transactions V of BVAG.
p) Target set = {-1, 0, 1} is the result of transaction V; where -1 represents unknown status, 0 represents no failure and 1 represents accessing failure.
q) BVAG transaction elements $V_\eta = \{V_{\eta x,qr} | r=1,2,...,k\}$ where $V_{\eta x,qr}$ is a queued element of V_η transaction.
r) BVAG transaction elements $V_\psi = \{V_{\psi x,qr} | r=1,2,...,k\}$ where $V_{\psi x,qr}$ is a queued element of V_ψ transaction.
s) BVAG transaction elements $V_\lambda = \{ V_{\lambda x,qr} | r=1,2,...,k\}$ where $V_{\lambda x,qr}$ is a queued element either in different set of transactions V_η or V_ψ.
t) $\hat{V}_{\lambda x,q1}$ is a transaction that is transformed from $V_{\lambda x,qr}$.
u) $V_{\mu x,q_1}$ represents the transaction feedback from a neighbour site. $V_{\mu x,q_1}$ exists if either $V_{\lambda x,qr}$ or $\hat{V}_{\lambda x,q1}$

Algorithm

Start
V_η and V_ψ initiates lock
    If set $V\lambda_{x,}q_1 = 0$
        Then abort $V\lambda_{x,}q_{r+1}$, ..., $V\lambda_{x,}q_k$
        Execute $V\lambda_{x,}q_1$, $\lambda = {}_{\eta,\psi}$
    End if
V_η and V_ψ propagate lock.
If $V\lambda_{x,}q_1$, $\lambda = {}_\eta$ or ${}_\psi$ get quorum
    If (write counter of w $V\lambda_{x,}q_1 \square V_\eta$ is equal n/2)
        Then w $V\lambda_{x,}q_1 \square V_\eta$ get quorum.
        $V\lambda_{x,}q_1$, $\lambda = {}_\eta$ transform to $\hat{V}\lambda_{x,}q_1$
        $\hat{V}\lambda_{x,}q_1$, $\lambda = {}_\eta$ release lock. $\forall\, V\lambda_{x,}q_1$, $\lambda = {}_\psi$
        $\hat{V}\lambda_{x,}q_1$, $\hat{V}\lambda_{x,}q_1$, $\lambda = {}_\eta$
    Else if (write counter of w $V\lambda_{x,}q_1 \square V_\psi$ is equal n/2)
        Then w $V\lambda_{x,}q_1 \square V_\psi$ get quorum.
        $V\lambda_{x,}q_1$, $\lambda = {}_\psi$ transform to $\hat{V}\lambda_{x,}q_1$
        $\hat{V}\lambda_{x,}q_1$, $\lambda = {}_\psi$ release lock. $\forall\, V\lambda_{x,}q_1$, $\lambda = {}_\eta$
        $\hat{V}\lambda_{x,}q_1$, $\hat{V}\lambda_{x,}q_1$, $\lambda = {}_\psi$

    End if
End if
S is fragmented into S_i
S_i is fragmented into ${}^{Si}{}_{Pkxx.}$
${}^{Si}{}_{Pkxx}$ divided into V_i
$\hat{V}\lambda_{x,}q_1$ update x.
Commit $\hat{V}\lambda_{x,}q_1$
    End.

4. Result and Discussion

To make it clearer on how we manage the transaction using BVAG, here we present the experiment result. If two sets of transactions, and initiates to update database *a* at replica 1 and 2, first it needs to request to update database *a* from primary replica 1 and 2. Then, neighbor binary voting assignment is initiated. All transactions in both nodes will try to get lock. The first transaction that initiated will get the lock and other transactions will be aborted. So now Replica 1 and 2 have a transaction waiting but transactions cannot read or update database *a* at same time.

Primary nodes 1 and 2 propagate lock to its neighbor replicas based on Picture 4. Primary replica for $V_{\eta x,q1}$ propagates lock to its neighbor replicas 2 and 4. Primary replica for $V_{\psi x,q1}$ propagates lock to its neighbor replicas 1 and 4. The first transaction get majority quorum will be transform to $\lambda_{x,}q_{1.}$

Table1: Experiment Result

Replica/ Time	1	2	4
t1	unlock (*a*)	unlock (*a*)	unlock (*a*)
t2	Begin transaction	Begin transaction	
t3	write lock (*a*), counter_w(*a*)=1	write lock (*a*), counter_w(*a*)=1	
t4	wait	wait	
t5	propagate lock: 2	propagate lock: 4	
t6	propagate lock: 2	get lock: 2, counter_w(*a*)=2 obtain quorum, release lock: 1	
t7	abort $V_{\eta a,q1,}$ lock (*a*) from 2		
t8		$V_{\psi a,q1}$ fragmented into S_2	
t9		S_2 is fragmented into ${}^{S2}{}_{Pkxx}$	
t10		${}^{S2}{}_{Pkxx}$ divided into T_2 where $T_1 = P_1$ = Primary key $T_2 = P_6$ = instant *a*	
t11		update *a*	
t12	Commit $\lambda_{a,}q_1$	Commit $\lambda_{a,}q_1$	Commit $\lambda_{a,}q_1$
t13	unlock (*a*)	unlock (*a*)	unlock (*a*)

5. Conclusion

In order to preserve data consistency and reliability of the systems, managing transactions is very important. This paper presents a new model to manage transactions called Binary Vote Assignment on Grid. We have implemented BVAG in a database environment using synchronous replication. Time-efficiency of transaction processing is calculated by how fast the time for commit or abort of the transaction has been reached. The faster the decision is reached, the higher the time efficiency will be [13]. Timeliness in synchronization has become show stopper to maximize the usage of system but at the same time contribute to the consistent and reliable computing. BVAG resolve this challenge by alleviates lock with small quorum size before capturing update and commit transaction synchronously to the database that require the same update data. The algorithm shows that it guarantees the consistency since the transaction execution is equivalent to one-copy-serializability.

6. Acknowledgement

Appreciation conveyed to Ministry of Higher Education Malaysia for project financing under Fundamental Research Grant Scheme, RDU 100109. Last appreciation will be to researchers in this research group.

7. References

[1] Mustafa Mat Deris A1, Jemal H. Abawajy A2, David Taniar A3, Ali Mamat , "Managing data using neighbour replication on a triangular-grid structure" , International Journal of High Performance Computing and Networking, Volume 6, Number 1, pp. 56 – 65, 2009.

[2] M. Mat Deris, D. J. Evans, M. Y. Saman, A. Noraziah, "Binary Vote Assignment on Grid For Efficient Access of Replicated Data", Int'l Journal of Computer Mathematics, Taylor and Francis, vol.80, no.12, pp. 1489 – 1498, 2003.

[3] Thomas Connolly & Carolyn Begg, Database Systems: A Practical Approach to Design, Implementation and Management 5th Edition, Addison Wesley, 2010. ISBN-13: 9780321601100

[4] Budiarto, S. Noshio, M. Tsukamoto, "Data Management Issues in Mobile and Peer-to-Peer Environment", Data and Knowledge Engineering, Elsevier, 41, pp.183-204, 2002.

[5] D. K. Gifford, "Weighted Voting for Replicated Data", Proceeding 7th Symposium on Operating System Principles, pp.150-162, Dec 1979.

[6] I.R. Chen, D.C. Wang and C.P. Chu, "Analyzing User-Perceived Dependability & Performance Characteristics of Voting Algorithms for Managing Replicated Data", Distributed and Parallel Databases, Kluwer Academic Publisher, pp. 199-219, 2003.

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

[8] B. Awerbuch and C. Tutu, "Maintaining Database Consistency in Peer-to-Peer Networks", Technical Report CNDS-2002-2, 2002.

[9] Noraziah A, Mat Deris M. Ahmed NA, Norhayati R, Saman MY, Zeyed MA (2007) Preserving Data Consistency through Neighbour Replication on Grid Daemon, American Journal of Applied Science, 4(10):748-755.

[10] F. Garcia-Carballeira, J. Carretero, A. Calderon, J.D. Garcia, L.M. Sanchez, A global and parallel file systems for grids, Future Generation Computer Systems 23 (1), 2007.

[11] Mat Deris M, Evans DJ, Saman MY, Noraziah A (2003) Binary Vote Assignment on Grid For Efficient Access of Replicated Data, International Journal of Computer Mathematics, Taylor and Francis, Vol.80, No.12, pp. 1489 – 1498.

[12] ATLAS at the University of Chicago (2010) http://www.sciencedirect.com/science?_ob=RedirectURL&_method=externObjLink&_locator=url&_cdi=5638&_plusSign=%2B&_targetURL=http%253A%252F%252Fhep.uchicago.edu%252Fatlas%252F.

[13] H. Yang, Z. Wang, Q. Deng, "Scheduling optimization in coupling independent services as a Grid transaction", Journal of Parallel and Distributed Computing (accepted on Jan 2008).