Handling Fragmented Database Replication through Binary Vote Assignment Grid Quorum

Ainul Azila Che Fauzi, A. Noraziah, Noriyani Mohd Zain, A.H. Beg, Nawsher Khan and Elrasheed Ismail Sultan
Faculty of Computer Systems and Software Engineering, University Malaysia Pahang, 26300, Kuantan, Pahang, Malaysia

Abstract: Problem statement: Organizations critically needed to supply recent data to users who may be geographically remote, while at the same time handle a volume request of distributed data around multiple sites. 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.

Approach: Data replication is a way to deal with this problem since it provides user with fast, local access to shared data and protects availability of applications because alternate data access options exist. Handling fragmented database replication becomes challenging issue to administrator since the distributed database was scattered into split replica partitions or fragments.

Results: This study presented a new mechanism on how to handle the fragmented database replication through the Binary Vote Assignment on Grid Quorum (BVAGQ). We address how to build reliable system by using the proposed BVAGQ for distributed database fragmentation.

Conclusion: The result shows that managing fragmented database replication and transaction through proposed BVAGQ is able to preserve the data consistency.

Key words: Data replication, data grid, Hierarchical Replication Scheme (HRS), Distributed Database Systems (DDS), Binary Vote Assignment on Grid Quorum (BVAGQ), vertical fragmentation, horizontal fragmentation

INTRODUCTION

Nowadays, organizations critically need to supply recent data to users who may be geographically remote and to handle a volume of requests of data distributed around multiple sites. One way to provide access to such data is through replication. It is broadly installed in disaster tolerance systems to replicate data from the primary system to the remote backup system dynamically and online (Ren et al., 2003). Replication provides user with fast, local access to shared data and protects availability of applications because alternate data access options exist (Dastgheib, 2010). Distributed database replication involves the process of copying and maintaining database objects in multiple databases that make up a Distributed Database Systems (DDS) (Ren et al., 2003). Handling fragmented database replication becomes challenging issue to administrator since the distributed database is scattered into split replica partitions or fragments. Each partition or fragment of a distributed database may be replicated into several different sites in distributed environment. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations. Fragmentation in distributed database is very useful in terms of usage, efficiency, parallelism and also for security. This strategy will partition the database into disjoint fragments. If data items are located at the site where they used most frequently, locality of reference is high. In fragmentations, similarly, reliability and availability are low Distributed Database, 2011. But by combining fragmentation with replication, performance should be good Distributed Database, 2011. Even if one site becomes unavailable, users can continue to query or even update the remaining fragments.

Data replication can be divided into three categories of fragmented replication scheme which are all-data-to-all-sites, some-data-to-all-sites and some-data-to-some-sites. The examples of all-data-to-all-sites protocols are Read-One-Write-All (ROWA) (Ahmad et al., 2010a; Deris et al., 2009) and Hierarchical Replication Scheme (HRS) (Perez et al., 2010). ROWA has been proposed preserving replicated data file in network environment (Ahmed et al., 2010a;
2010b). Meanwhile, replication in HRS starts when a transaction initiates at site 1. All the data will be replicated into other sites. All sites will have all the same data. For some-data-to-all-sites category, The Majority Quorum protocol and Weighted Voting protocol employ voting to decide the quorums techniques (Choi and Youn, 2010). A tree structure has been assigned to the set of replicas in this technique. The replicas are positioned only in the leaves, whereas the non-leaf nodes of the tree are regarded as “logical replicas”, which in a way summarize the state of their descendants (Storm and Theel, 2009). Besides Voting Protocol, Tree Quorum (TQ) (Choi and Youn, 2010) can also be categorized in some-data-to-all-sites. These replication protocols make use of a logical tree structure. The cost and availability vary according to the failure condition, whereas they are constant for other replication protocols (11). One more protocol in this category is Branch replication scheme (Perez et al., 2010). Its goals are to increase the scalability, performance and fault tolerance. Replicas are created as close as possible to the clients that request the data files. Using this technique, the growing of the replica tree is driven by client needs. Binary Vote Assignment on Data Grid (BVADG) (Ahmad et al., 2010b) is one of the protocols in some-data-to-some-sites protocol. A data will replicate to the neighboring 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 neighbors. Thus, the number of neighbors of each sites is less than or equal to four.

**Research background:** Data replication: 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 (Gudiu et al., 2010; Connolly and Begg, 1998). 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 (Storm and Theel, 2009).

**Distributed database fragmentation:** Fragmentation in distributed database is very useful in terms of usage because usually, applications study with only some of relations rather than entire of it (Connolly and Begg, 1998). In data distribution, it is better to study with subsets of relations as the unit of distribution. The other benefit from fragmentation is the efficiency. Data is stored close to where it is most frequently used and for data that is not needed, it is not stored. By using fragmentation, a transaction can be divided into several subqueries that operate on fragments. So, it will increase the degrees of parallelism. Besides, it also good for security as data not required for local applications is not stored. So, it will not available to unauthorized users. There are two main types of fragmentation which are horizontal and vertical. Horizontal fragments are subsets of tuples, whereas vertical fragments are subsets of attributes. Figure 1a and b show the horizontal and vertical fragmentations.

**Horizontal fragmentation:** Horizontal fragmentation groups together the tuples in a relation that are used by the important transactions (Atlas at the University of Chicago, 2011). A horizontal fragment is produced by specifying a predicate that performs a restriction on the tuples in the relation. It is defined using the Selection operation of the relational algebra. Given a relation R, a horizontal fragment is defined as:

$$\text{op}(R)$$

where, p is a predicate based on one or more attributes of the relation.

**Vertical fragmentation:** Vertical fragmentation groups together the attributes in a relation that are used jointly by the important transactions (Atlas at the University of Chicago, 2011). A vertical fragment is defined using the Projection operation of the relational algebra. Given a relation R, a vertical fragmentation is defined as:

$$\Pi a_1,\ldots, a_n(R)$$

where, a1,…,an are attributes of the relation R.

![Fig. 1a and b: Horizontal and vertical fragmentations](image)
MATERIALS AND METHODS

Binary Vote Assignment Grid Quorum (BVAGQ) technique will be used to approach the research. In BVAGQ, all sites are logically organized in form of two-dimensional grid structure. Each site has a premier data file. A site is either operational or failed and the state (operational or failed) of each site is statistically independent to the others. A data will replicate to the neighboring sites from its primary site. Consider a case of 9 sites logically organized in 3x3 two-dimensional grid structures. Four sites on the corners of the grid have only two adjacent sites and other sites on the boundaries have only three neighbors. Thus, the number of neighbors of each sites is less than or equal to 4. In Fig. 2, data from site 1 will replicate to site 2 and 4 which are its neighbors. Site 5 has four neighbors, which are sites 2, 4, 6 and 8. So, site 5 has five replicas. Meanwhile, site 6 replicates to site 3, 5 and 9.

Definition:

- V is a transaction
- S is relation in database
- S_i is vertical fragmented relation derived from S, where i = 1, 2, ..., n
- PK is a primary key
- x is an instant in T which will be modified by element of V
- T is a tuple in fragmented S
- S_PK개 is a horizontal fragmentation relation derived from S
- P_i is an attribute in S where i = 1, 2, ..., n
- M_ij is an instant in relation S where i and j = 1, 2, ..., n
- i represent a row in S
- j represent a column in S
- η and ψ are groups for the transaction V
- γ=a or b where it represents different group for the transaction V (before and until get quorum)
- 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
- D is the union of all data objects managed by all transactions V of BVAG
- 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
- BVAG transaction elements V_η = {V_ηx,qr | r=1,2,..,k} where V_ηx,qr is a queued element of V_η transaction
- BVAG transaction elements V_ψ = {V_ψx,qr | r=1,2,..,k} where V_ψx,qr is a queued element of V_ψ transaction

RESULTS

To make it clearer on how we manage the transaction using BVAGQ, here we present the example case. Each node is connected to one another through an Ethernet switch hub. A cluster with 3 replication servers connected to each as shown in Fig. 3.

Using BVAG rules, each primary replica will copy database x to its neighbor replicas. Client can access database x at any server that has its replica. We assume that primary database a located in Server 1, primary database b will be at Server 2 and so on. Based on BVAGQ model, a for V_λx,q1 will be any instant a, b, c, d, e, f, g, h and i.

DISCUSSION

For the first experiment, consider V_λx,q1,λ=η request to update data a at server 1. The first request
that get lock which is \( V_{\lambda, q_1} \) will proceed with the transaction and \( V_{\lambda, q_1} \), \( qr+1, \ldots, V_{\lambda, q_k} \) aborted as shown in Table 1. \( V_{\lambda, q_1} \) is the write counter for \( V_{\lambda, q_1} \) that increases when it gets a lock. Next, the \( V_{\lambda, q_1} \) fragmented into \( S_1 \) and \( S_2 \) is fragmented into \( S^2_{PKXX} \). Based on the primary key of the fragmented tuple; instant a will be updated. After finish update, the transaction will commit.

For second transaction, if two sets of transactions, \( V_{\lambda, q_1} = \eta \psi \) and \( V_{\lambda, q_1} = \psi \) initiates to update database a at replica 1, transaction \( V_{\lambda, q_1} = \eta \psi \) will abort. Transaction \( V_{\lambda, q_1} = \psi \) is aborted because we already fix the system will choose the first transaction that make request based on timestamps. After identify which transaction will be executed, we will fragmented the database using horizontal and vertical fragmentation to get the instant that we want to update. From Table 2, we can see that \( V_{\lambda, q_1} \) precede the transaction execution. \( V_{\lambda, q_1} \) fragmented into \( S_1 \) and again \( S_2 \) is fragmented into \( S^2_{PKXX} \). Instant a will be update. After that, all replica will commit and unlock.

### Table 1: Experiment result

<table>
<thead>
<tr>
<th>Time</th>
<th>Replica 1</th>
<th>Replica 2</th>
<th>Replica 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
</tr>
<tr>
<td>0.2</td>
<td>Begin transaction</td>
<td>Begin transaction</td>
<td>Begin transaction</td>
</tr>
<tr>
<td>0.3</td>
<td>( V_{\eta, q_1} ) get lock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>( V_{\eta, q_1} ), ( V_{\eta, q_2} ) aborted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>( V_{\eta, q_1} ) fragmented into ( S_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>( S_1 ) is fragmented into ( S^1_{PKXX} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>( S^1_{PKXX} ), divided into ( T_1 + \psi )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>( T_1 = P_1 = \text{Primary key} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>( \lambda \psi ) commits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
</tr>
</tbody>
</table>

### Table 2: Experiment result

<table>
<thead>
<tr>
<th>Time</th>
<th>Replica 1</th>
<th>Replica 2</th>
<th>Replica 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
</tr>
<tr>
<td>0.2</td>
<td>Begin transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>write lock (a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>wait</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>( V_{\eta, q_1} ), ( V_{\eta, q_2} ) aborted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>( V_{\eta, q_1} ), ( V_{\eta, q_2} ) aborted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>( V_{\eta, q_1} ) fragmented into ( S_2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>( S_2 ) is fragmented into ( S^2_{PKXX} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>( S^2_{PKXX} ), divided into ( T_2 + \psi )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>( T_2 = P_2 = \text{Primary key} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.11</td>
<td>( \lambda \psi ) commits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.12</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
<td>unlock (a)</td>
</tr>
</tbody>
</table>

### CONCLUSION

Handling fragmented database replication is very important in order to preserve the data availability, consistency and reliability of the systems. Therefore, a new Binary Vote Assignment on Grid Quorum technique has been proposed to maintain and manage the fragmented database replication. From the experiment result, it shows that the system preserves the data consistency through the synchronization approach for all replicated sites. Furthermore, it guarantees the consistency since the transaction execution is obeyed the one-copy-serializability.

### ACKNOWLEDGEMENT

Appreciation conveyed to Ministry of Higher Education Malaysia for supporting this project under Fundamental Research Grant Scheme, RDU100109.

### REFERENCES


