A Synchronous/Asynchronous Multi-Master Replication Method

Amer Chahine and Ramzi A. Haraty
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: {amer.chahine@lau.edu.lb, rharaty@lau.edu.lb}

Abstract

In this paper, we present synchronous/asynchronous multi-master replication architecture and a replica control algorithm, which guarantee and provide a flexible and transparent solution. This architecture holds the advantages of previously applied methods and techniques. System performance is increased by lowering the load or usage of network resources. High availability is guaranteed by keeping synchronized nodes online or active at all time. High reliability is maintained by ensuring that the available nodes are synchronized and up-to-date. Finally, flexibility is provided in handling new nodes without the need of taking the full system offline.

Keywords: Asynchronous architecture, database replication, and multi-master approach.

1. Introduction

Information availability becomes critical nowadays mostly due to the data-intensive applications, which populate the Internet. These throughput requirements, drive research towards distributed and replicated architectures, providing solutions that enable high availability through network shared contents. Such systems require efficient storage capability as well as easy retrieval and outstanding processing capabilities. Combining networked storage and processing entities, enables data distribution and/or replication, reduces wasted storage capacity and backup inconveniences as well as increases data availability.

Computer systems are considered to be more distributed, widespread and complex; thus, they are becoming more fragile. In terms of consequences, failures are becoming more serious and frequent. Fault tolerance and security issues are specialized topics in computer systems. Lately, societies are becoming more dependent on computers. Computer systems became an essential need in people daily life. Therefore, lives are affected by bugs, crashes and other serious problems [10].

Many work and research has been done on security and fault tolerance, even software sellers are considering stability and security as an official priority [6][7][9]. Different techniques and approaches have been proposed promising stable, secure, and fault-tolerant systems. Approaches that offer fault-tolerance include specialized computer languages, software engineering, and mainly replication [1][2][3]. In this work we concentrate on replication.

While replication is considered an optimal solution for fault tolerance, it is not easy to implement a consistent replicated system. A minimal error in a critical system that manages an industrial unit results with crucial consequences. For example, if a server crashes, a slightly wrong account in a banking system is unacceptable. To enforce strict consistency and stability in case of a system crashes, replication is considered a complicated problem.

Customer information, inventories, payrolls are other important data, which belong to a banking database application are stored and saved in a secured databases that represent the key element of the IT infrastructure of different companies. Therefore, it is a good idea to replicate such databases in order to increase fault tolerance. However, database replication is considered to be complex and complicated when strict consistency is enforced [4].

In this paper we propose combined replication architecture and a replica control algorithm, which guarantees and provides a flexible and transparent solution. This architecture holds the advantages of previously applied methods and techniques such as increasing the performance of the system, maintaining high reliability, and finally being able to handle additional replicas by maintaining the availability status of the whole system. Section 2 overviews multi-master replication. Section 3 presents the proposed architecture. Section 4 presents the experimental study, and section 5 concludes the work.

2. Multi-Master Replication

Multi-master replication technique is one of the latest techniques in the field of database replication. The environment of multi-master replication is as follows: each site is considered to be a primary; none of the available sites is considered secondary. All the sites contain the same and complete amount of data and all databases are fully synchronized at all times. Writing to

site and deleting from site is done to all sites at the same time. Simultaneously, when a modification is done, it is propagated to all other available sites in the system, whereas reading is done from the first available site. This keeps the replica in a full synchronization state; however, some problems are introduced which will be discussed shortly in the coming sections.

Multi-master replication technique enforces data items to be always synchronized. When any update or delete transaction is executed in one database, this transaction will be immediately propagated in the same way to all other replicas in the system.

The multi-master replication architecture consists of a group of databases forming a replication group. One of these databases is considered as the master definition site; where as all the other databases are defined as master sites [8]. The job of the master definition is site is to invoke most of the replication administration commands.

In spite of being a new technique that is developed and proposed to overcome the weaknesses of traditional replication methods, multi-master replication technique faces some challenges. Asynchronous multi-master replication has been around longer than synchronous multi-master replication [8]. In this section we will point out the advantages that the asynchronous multi-master replication has over the synchronous multi-master replication.

Asynchronous replication provides higher performance and less network communication than the synchronous replication. Moreover, asynchronous replication provides higher efficiency as it stores a bulk of transactions in a buffer and then it propagates them all as one big transaction instead of propagating each transaction individually. This approach is highly desirable when replicating remote nodes which are distributed in different regions.

The synchronous replication approach results in a high overhead as it demands each transaction to propagate individually to other replicas; hence, establishing a separate connection. Fewer connections are established while propagating transactions in the asynchronous multi-master replication; propagation is done to a set of transactions instead of individual ones.

Furthermore, asynchronous multi-master replication offers higher availability. When any site becomes offline in the asynchronous multi-master replication environment, all other replicas which are available will receive the updates and will be synchronized. Reading and writing to replicas will still be possible. Those crashed nodes which did not receive the transactions will be informed about the updates as soon as they become online and transactions will be propagated to them from the deferred transaction queue.

In synchronous multi-master replication environment, if there exist a single crashed site, which has been disconnected due to a certain failure, propagation of transaction will not take place. Synchronous multi-master replication provides lower availability because in order for propagation and synchronization to take place, it forces all the nodes in the replication group to be available. This method follows the all-or-none approach. A single disconnected site will prevent propagation of transaction to occur.

3. Proposed Architecture

The general architecture of the proposed replication combines the advantages of both the synchronous and the asynchronous multi-master methods. The synchronous approach provides reliability and availability and the asynchronous approach provides flexibility and performance.

The architecture is made up of two layers. The upper layer behaves asynchronously while dealing with clusters, whereas the lower layer behaves synchronously while dealing with the inner clustered nodes. Therefore, our proposed architecture behaves in a synchronous/asynchronous multi-master replication approach. Fore more information, the reader is referred to [5].

The top layer contains a group of clusters; every cluster consists of some nodes. The nodes are grouped or added to each cluster according to a certain criterion set by the system, such as distributing the nodes equally into clusters or distributing them according to the nearest site or region. For example, when adding new node to the system it may be inserted into the corresponding cluster with the least number of nodes in order to maintain equal number of nodes in each cluster. Say there are four clusters and a total of 20 nodes, so each cluster would contain five nodes. This criterion will maintain the system performance by dealing with one cluster of nodes rather than dealing with the entire nodes that might be distributed to one cluster, and will ensure availability by bringing some clusters offline sometimes and maintaining other clusters online at all times.

Following the asynchronous multi-master approach, each client request is delivered to the coordinator of the first layer following the same way as it is in [11]. A measurement for all clusters is calculated to find the best cluster with the lightest transaction load to send the transactions to. Asynchronously, transactions propagate to the coordinator of the chosen cluster. Then synchronous

multi-master approach will take place in propagating every transaction to each node within the cluster immediately. After updating the corresponding cluster, these transactions will be propagated again from the coordinator/refresher of the top layer to the second layer coordinator of the next best cluster following the same procedure. Eventually, all the nodes in each cluster will receive the transactions.

When any cluster gets disconnected from the system due to a certain system error or malfunction, transactions which are aware of the failed cluster, are saved in the logs of the main coordinator and then they will be forwarded to the best available cluster in the system. However, if one node fails in a certain cluster, it will be taken off totally from that cluster. Any incoming transaction to this cluster will be saved in the cluster's coordinator logs and will be propagated directly to all the active nodes within this cluster. When there is a recovered node or in case of adding additional node to a cluster, this node will synchronize with all the available nodes to become totally synchronized.

To resynchronize old existing nodes coming from failure or disconnection, the system puts the node back in the same cluster that it was under, and then it disconnects this cluster from the whole system in order for the internal nodes to be updated and synchronized internally as the case in synchronous replication method. In this case only one cluster is brought offline and the other clusters are kept online to maintain availability. The same procedure takes place when a newly available node is to be added to the replication system.

The advantage of this technique in handling failed nodes is to preserve the system performance. This technique removes the failed node in order to maintain system availability also.

3.1 Algorithms

In this section we present two algorithms. The first one corresponds to the architecture of the upper layer, mainly for the coordinator/refresher of this layer. The second one corresponds to the architecture or behavior of the lower layer coordinator.

Algorithm for Coordinator/Refresher

This algorithm waits for an incoming request transaction. Calculation of all active clusters is done for choosing the best cluster with lightest weight in order to forward transactions to. If the cluster contains no failures then transactions are propagated to it directly, else they will be forwarded back to the coordinator and to the cluster with the next lightest weight.

```
Do while system is up
   Wait for new transactions then
   If new transactions then
      Calculate cluster weights
   Find winner (least weight)
   If there exists any failure in a cluster then
      Forward transaction to refresher and next winner
   Else
      Forward to winner
   End If
   End If
End While
```

Algorithm for the Coordinator within the Cluster

In this algorithm we first check if new transactions arrived in order to propagate them directly within the cluster. Then the algorithm checks if a failure exists in a node so as to remove it totally. Finally, it checks if an old node is back online so as to resynchronize it by bringing the cluster offline and requesting the transactions saved in the local refresher to be propagated again, and then to ensure that the cluster is already synchronized it request any new transactions available in the main coordinator/refresher.

```
Do while the cluster is up
   If new transactions then
      Propagate transactions synchronously
   Else if failure exists in a node then
      Remove node totally
   Else if node is back online then
      Bring current cluster offline
      Resynchronize this node by requesting transactions
      from refresher
      Resynchronize the cluster by requesting
      transactions from coordinator/refresher
      Bring cluster online
   End If
End While
```

4. Experimental Study

In this section, we present an experimental study about how these transactions are going to propagate following the synchronous, asynchronous and finally the synchronous/asynchronous multi-master replication methods. We will focus on the time needed for full synchronization to take place, the number of connections established during synchronization process, how message exchange takes place, and finally the total amount of the replication system usage resources.

Assume the following three transactions tA, tB, and tC to be propagated to the following six replicas R1, R2, R3, R4, R5, and R6 in the same replication group. We also assume that each transaction needs certain period of time

to be successfully propagated to each replica taking into consideration that each replica is always available and having stable network resources and characteristics. Finally, we assume 2, 1, and 3 to be the corresponding values for the time needed for tA, tB, and tC respectively to propagate to each replica.

Following the synchronous approach:

For synchronous replication, the system first checks whether all replicas are online in order for the transactions to be propagated. Then, each transaction *t* is propagated separately to each replica establishing T*R connections for x_t period of time; where T is the total number of transactions needed to be propagated, R is the number of all replicas in the replication group, and x is the period needed for each transaction to be fully propagated on each replica. Referring to our example, there are 3 transactions and 6 replicas, establishing a total of 18 connections.

The total period of time needed for all connections is calculated by $\Sigma^{T}_{t=1} x_t$. Therefore, applying the formula we get a total period of: 2 + 1 + 3= 6.

Finally, as it is depicted in figure 1, the system will be always busy establishing full connection with each replica until every transaction is fully propagated.

Following the asynchronous approach:

First, the system checks for the best replica among all the available ones. Then, all transactions are propagated as one big transaction asynchronously to each replica establishing R connections for x_t period of time (starting with the *best node*). Therefore, as there are 6 replicas in our example, the total number of connections to be established is 6.

The total period of time needed for all connections is calculated by $R*\Sigma^{T}_{t=1} x_t$. Therefore, applying the formula we get a total period of: 6 *(2 +1 +3) = 36.

Finally, as it is depicted in figure 2, the system will be dealing with one replica at a certain period of time; thus, the system will consume fewer resources as few connections are established at a certain period of time through out the process of data synchronization.

Following the synchronous/asynchronous multi-master approach:

For this approach, we assume that the replicas are clustered equally into two clusters C1 and C2. First, the system checks for the best cluster among all available clusters in the replication system. Then, all transactions are propagated asynchronously to each cluster establishing C connections. Each transaction t is then propagated separately to each cluster establishing T*CR connections; where T is the total number of transactions needed to be propagated and CR is the number of clustered replicas in a certain cluster. Therefore, the total number of connections established during this replication method is T*CR+C. Referring to our example, there are 3 transactions and 6 replicas which are clustered into two separate clusters, establishing a total of 3*6+2=20 connections. The total period of time needed for all connections is calculated by

$$\Sigma^{C}_{t=1} \Sigma^{T}_{t=1} x_t.$$

Therefore, in our example, a total of: (2 + 1 + 3) + (2 + 1 + 3) = 12 units of time is required by the system to establish a full synchronization. However, applying the formula for C=1, i.e. for only one cluster (the *best cluster*), will produce a synchronized set of replicas. In other words, only a period of 6 is needed to produce a fully synchronized set or replicas under the same cluster.

Finally, as it is depicted in figure 3, the system will be dealing with only one cluster of replicas at a certain period of time; thus, the system will consume less resources as there is a partial number of connections established at a time to synchronize one cluster at a certain period of time (Total Amount of Resources/Number of Clusters).

Synchronous and Asynchronous vs. Synchronous/Asynchronous

Although the synchronous/asynchronous multi-master establishes more connections than the synchronous replication method and although the total period of time for the all connections required to be established in this approach is more than that established in the synchronous approach, the synchronous/asynchronous multi-master replication has fewer message exchanges and consumes less network and system resources than the synchronous method. The synchronous replication locks all the available replicas in the system while synchronization, where as in the synchronous/asynchronous replication few replicas are being handled at a time, providing a faster way of replicating data. Unlike the synchronous/asynchronous multi-master approach, the synchronous replication method establishes full connection to all replicas at every time a propagation of an update is needed to be done.

As for the asynchronous approach, although it establishes fewer connections than the synchronous/asynchronous multi-master approach, and despite its low usage to system resources, the total period of time needed for all the established connections during the process of system replication is much more than the period of time needed for connections in the proposed

synchronous/asynchronous multi-master architecture. Therefore, unlike the synchronous/asynchronous multi-master replication approach, the asynchronous replication method is not a good option when time sensitive data that needs to be updated more often is being replicated.

Table 1 presents a comparison between the synchronous, asynchronous and synchronous/asynchronous multi-master replication methods.

5. Conclusion and Further Work

Data replication is used to improve data availability and query load balancing and thus performance. In this work, we have proposed synchronous/asynchronous cluster architecture and a replica control algorithm, which guarantees and provides a flexible and transparent solution that enables users' requests to be managed asynchronously in order to avoid any type of system bottleneck.

We presented the design and architecture of a new replication technique. The multi-master refresher algorithm prevents conflicts, by exploiting the cluster's high speed network; thus, providing strong consistency, without the constraints of eager replication. Despite that this architecture proposes synchronous approach in layer level two, all its disadvantages are handled by the asynchronous approach that is proposed and followed in the top layer of the architecture. The need for synchronous approach, which guarantees reliability, comes from the fact that there exist some time sensitive data items (i.e. stock quotes) which would be updated more frequently than less time sensitive records (i.e., salary).

In conclusion, the proposed architecture holds the advantages of previously applied methods and techniques such as increasing the performance of the system by lowering the load or usage of network resources, ensuring high availability by keeping synchronized nodes online or active at all time, maintaining high reliability by ensuring that the available nodes are synchronized and up-to-date, and finally being able to handle additional nodes.

Future work includes proposing optimization techniques which would provide the refresher procedure a better and enhanced method to manage transactions on the running queue. Further work also includes defining the best criteria for distributing nodes into clusters so as to maintain load balancing and thus performance.

References

[1] Agrawal D., Alonso G., El Abbadi A., and Stanoi I. "Exploiting Atomic Broadcast in Replicated Databases." In Proceedings of European Conference on Parallel Processing (Euro-Par), pages 496–503, Passau, Germany. 1997.

[2] Anderson T. A., Breitbart Y., Korth H. F., and Wool A. "Replication, Consistency, and Practicality: Are these Mutually Exclusive?" In Proceedings of the ACM SIGMOD International Conference on the Management of Data, pages 484–495, Seattle, Washington. 1998.

[3] Bernstein P., Brodie M., Ceri S., DeWitt D., Franklin M., Garcia-Molina H., J. Gray, J. Held, J. Hellerstein, H. V Jagadish, M. Lesk,D. Maier, J. Naughton, H. Pirahesh, M. Stonebraker, and J. Ullman. "The Asilomar Report on Database Research." Technical Report MSRTR-98-57, Microsoft Research, One Microsoft Way, Redmond, WA 98052. 1998.

[4] Cap C. H. "Distributed Systems with Data Replication: A Non-technical Survey." Technical Report ifi-90.11, Department of Computer Science, University of Zürich, 190, Winterthurstraße CH-8057 Zürich, Switzerland. 1990.

[5] Chahine, A. "A Synchronous/Asynchronous Multi-Master Replication Method". Master's Thesis. Lebanese American University. 2007.

[6] Coulon, C., Pacitti, E., Valduriez, P. "Scaling up the Preventive Replication of Autonomous Databases in Cluster Systems." International Conference on High Performance Computing for Computational Science (VecPar -2004), Valencia, Spain. 2004.

[7] Garmany, J. And Freeman, R. "Multi-Master Replication Conflict Avoidance and Resolution." Oracle Replication, Rampant TechPress, 4th Qtr 2004, pp. 9-15.

[8] Keating, B. "Challenges Involved in MultiMaster Replication." http://www.dbspecialists.com. 2001.

[9] Maney K. "Microsoft Shifts its Focus to Security. USA Today. 2002.

[10] Neumann P. G. "The Risk Digest: Forum on Risks to the Public in Computers and Related Systems." ACM Committee on Computers and Public Policy, 21(87). 2002.

[11]Pacitti, E., Ozsu, M. T., Coulon, C., Valduries, P. "Preventive Replication in a Database Cluster," Distributed and Parallel Databases, 18,223-251. 2005.

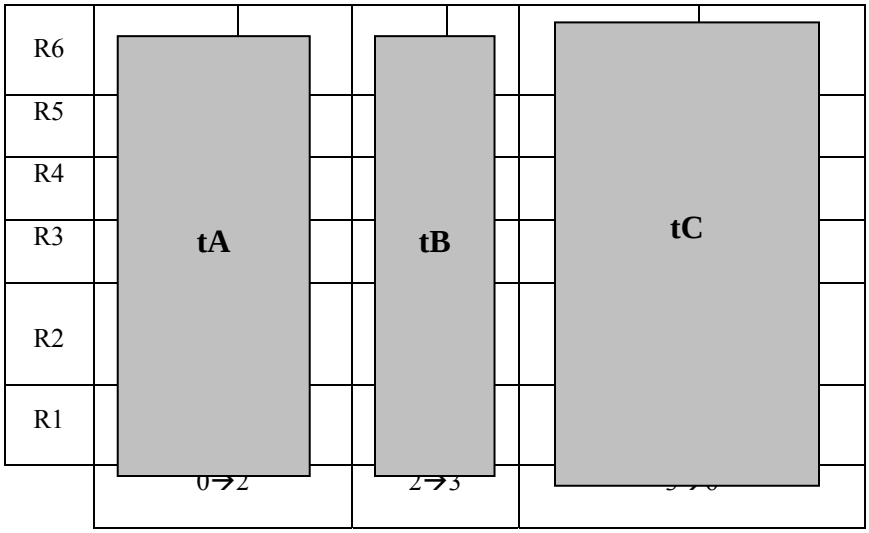

Figure 1: System during Synchronous Approach

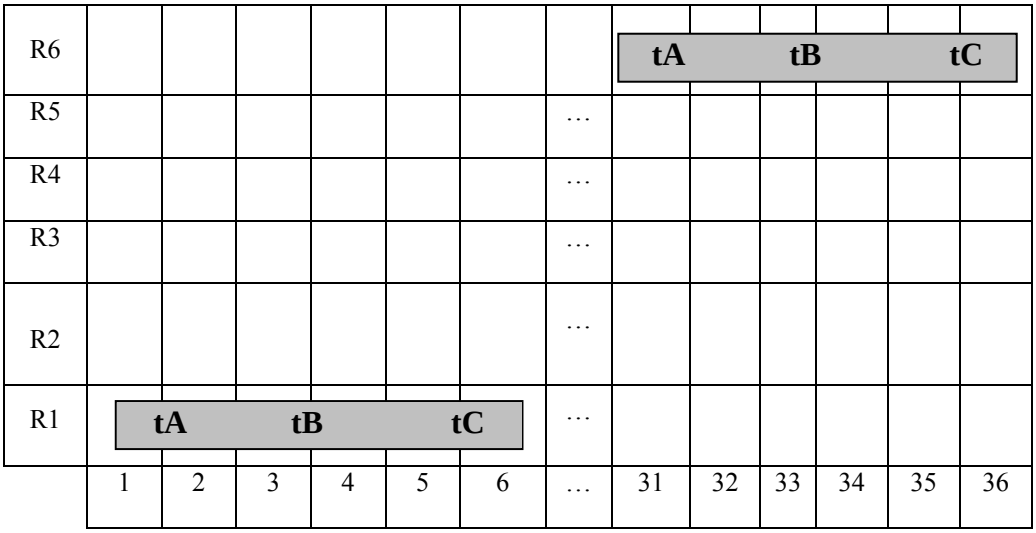

Figure 2: System during Asynchronous Approach

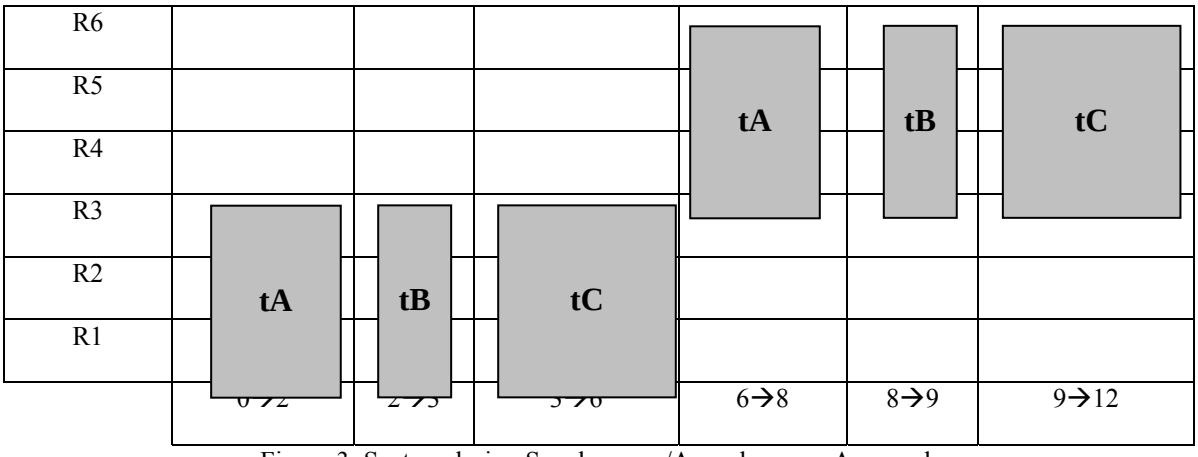

Figure 3: System during Synchronous/Asynchronous Approach

Table 1: Comparison among the Various Replication Techniques

	Synchronous	**Asynchronous**	**Synchronous/ Asynchronous**
Total Number of connections	T*R	R	T*R + C
Period for connections	$\Sigma^{T}_{t=1} x_t$	$R*\Sigma^{T}_{t=1} x_t$	$\Sigma^{C}_{t=1} \Sigma^{T}_{t=1} x_t$
Usage of Resources	High	Low	Moderate
Message Exchange	1 Transaction-to- N Replica	N Transactions-to- 1 Replica	1 Transaction-to- N/C Replica

ACKNOWLEDGEMENTS: This work is funded by the Lebanese American University.