Performance Analysis of ROWA Protocol for Controlling Replica in Data Replication

Md. Al Mehedi Hasan, Md. Mamun Ar Rashid, Md. Ayub Hossain Md. Rabiul Islam, Md. Omar Faruqe[1]
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology (RUET)
e-mail: { mehedi_ru, cse_mam, ayub_305, rabiul_cse, faruqe_cse[1]}@yahoo.com

Abstract: *Effective data management in today's competitive enterprise environment is an important issue. Data is information and information is knowledge. Hence first and effective access to data is very important. Replication is one such widely accepted phenomenon in distributed environment, where data is stored at more than one site for performance and reliability reasons. By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find close to the site where the request originated, thus increasing the performance of the system by reducing network traffic. If the application has read-only nature, replication can greatly improve the performance. But if the application needs to process update requests, the benefits of replication can be neutralized to some extend by the overhead of maintaining consistency among multiple replicas. This paper focuses performance of* ***Read One Write All (ROWA)*** *protocol for controlling replica of data replication in distributed database system using* ***ASP.NET web service.***

Key Words: ROWA, ASP.NET, Middleware Software, Distributed Database, Replication.

1. INTRODUCTION

Computing infrastructure and network application technologies have come a long way over the past 20 years and have become more and more on detached from the underlying hardware platform on which they run. At the same time computing technologies has evolved from monolithic to open and then to distributed systems. Both scientific and business applications today are generating large amount of data, typical applications, such as High Energy Physics and bioinformatics, will produce petabytes of data per year. In many cases data may be produces, or required to be accessed or shared, at geographically distributed sites [1,2,4]. Sharing of data in a distributed environment gives rise to many design issue e.g. access permissions, consistency issues, and security. Thus, effective measures for easy storage and access of such distributed data are necessary. One of the effective measures to access data effectively in a geographically distributed environment is replication [1,2]. The reason for data replication in distributed system has the following facts:

- Increased availability
- Increased performance and
- Enhanced reliability

2. THE DATABASE AND NETWORK *SYSTEM*

A replicated database is a distributed database in which multiple copies of some data items are stored at multiple sites. By storing multiple copies, the system can operate even though some sites have failed. Maintaining the correctness and consistency of data is of prime importance in a DBMS. In distributed DBMS [5] it is assumed that a replicated database should behave like a database system managing a single copy of the data .As replication is transparent from user's point of view, users may want to execute interleaved executions on a replicated database to be equivalent to a one-copy database, the criterion commonly known as one-copy serializabilty.We model a distributed database as a collection of files. Files are assumed to be unit fully replicated database. Each replicated site has three components: a source replicated database, replica control manager and a resource manger (RM). The network can be LAN, MAN or WAN. If different network types exist then it must be connected by appropriate component. The ROWA protocol is able to receive read/write request from any network type and back result to appropriate clients using ASP.NET web service [6] as platform of distributed system and read One Write All protocol.

3. READ ONE WRITE ALL (ROWA) PROTOCOL

A read-one write-all (ROWA) protocol [3, 7] is commonly used in distributed systems to improve performance and availability of mostly read data. A read operation is done from the nearest replicated database in distributed system therefore very efficient and faster. To ensure consistency of data, a write operation is done on all replicas in distributed system and hence is very expensive and has low availability. Low write availability is a major limitation of read-one write-all (ROWA) protocol. ROWA protocol is based on the notion of an **epoch**

ISBN 984-300-002131-3

which is a set of nodes that contain replicated database in the distributed system known at some point to be operational and connected. The system periodically runs a special operation, **epoch checking**,[7] that polls i.e. uniquely identify all nodes that is the member in current epoch. . If any members in current epoch are not accessible (failures detected), or any nodes outside of the current epoch are contacted (repairs detected), an attempted is made to form a new epoch. Epochs are distinguished by their epoch numbers. By performing epoch checking one member in current epoch can find out the location of other member in current epoch and also able to verify validity of current epoch membership in distributed system.

3.1 Epoch checking

The algorithm for epoch checking and switching to a new epoch [7] is shown in Algorithm 1.

3.2 Read operation

The algorithm for the read operation is simple. A node that initiates the read of data item x (the co-ordinator) chooses the closest node that has a replica of x (the participant) among the nodes represented in the epoch of the owner transaction. The initiator then sends a request for the data to the chosen replica together with the epoch number of the owner transaction. If the participant's replica marked stale, or if its epoch number is deferent from the epoch number of the owner transaction, the participant rejects the request which causes the initiator to repeat the operation with another participant (or abort if none is available). Otherwise, the participant obtains the local read. lock for the replica, checks its epoch number again and, if it is still equal to the one from the read request, sends the data to the coordinator and releases the read lock.

3.3 Write operation

The algorithm for the write operation [7] is shown in Algorithm 2.

4. EXPERIMENTAL ENVIRONMENT

Experimental evaluation has been conducted by running an actual implementation of the protocol within a simulated environment. We considered eight replicated site. We perform three kind of experiment to measure the performance of ROWA protocol in LAN using windows system. All experiment has been done at site-1. At the time of experiment to measure the performance of ROWA protocol the data transfer rate between the replicated sites was 10 Mbps.

Algorithm 1: To perform Epoch checking

1. Obtain the lock for epoch checking;
2. Multicast (all-nodes-in-the-system, "epoch-check-request");

5. EXPERIMENTAL RESULT

Assuming service getting time of a client computer for read or writes operation is SGT and defined as time between start time of client request and the time when service of request is available to client computer**.**

5.1 Performance of read operation

SGT time in millisecond increases near linearly for read operation when distance of a client increase from a replicated site (from site-1).The ROWA protocol stops any kind of write operation to all sites at the time of read operation.

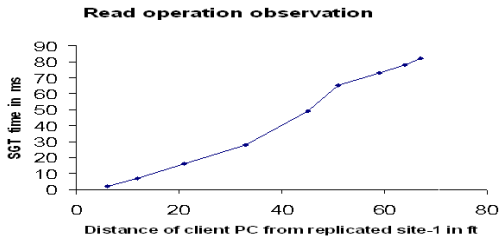

Fig 1: Observation of response time of read operation for various distance of client from site-1

5.2 Performance of write operation

SGT time in millisecond increases near linearly for write operation when distance of a client increase from a replicated site (from site-1).The ROWA protocol stops any kind of read operation from all sites at the time of write operation.

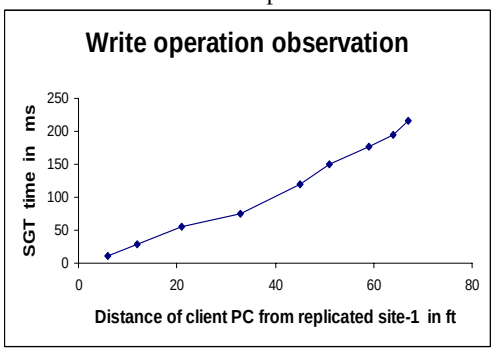

Fig 2: Observation of response time of a write operation for various distance of client from site-1

```
3. Receive RESPONSES [1: k];
4. m: = an index j such that enumber j = max i=1…k (enumber i);
5. if | {node1.....node k} intersection (ELIST)m |>= [ |(ELIST)m|/2] and if {node1.....node k}! =(ELIST)m then
6.      NEW-EPOCH: = {node1..... Nodek};
7.      NEW-NODES: = NEW-EPOCH \ (ELIST) m;
8.      for each node i belong to NEW-NODES do in parallel
9.           STALE-REPLICASi=0;
10.          for each replica x i residing on node i do
11.             if there exists a node p such that p has a replica of x and enumber i <(latest-epoch)m(P) then
12.                  STALE-REPLICAS i := STALE-REPLICASi intersection { xi} ;
13.             endif;
14.          endfor;
15.          if STALE-REPLICASi! = 0; then
16.            send-message (i, (“mark-stale", STALE-REPLICASi));
17.            if i fails to acknowledge that it marked the specified replicas stale then
18.               NEW-EPOCH: = NEW-EPOCH \ {i};
19.            endif;
20.          endif;
21.    endfor;
22.    New-epoch-number: = enumber m + 1;
23.    Do-atomically
24.          multicast (NEW-EPOCH, (“switch-epoch", NEW-EPOCH, new-epoch-number));
25.    end-do-atomically;
26.    Release the lock on epoch checking;
27. end if;
```

Algorithm 2: To perform write operation

```
1. Write(x, new-value, elist (T), enumber (T));
2. PARTICIPANTS: = {node p such that p belongs to elist (T) and p has a replica of x};
3. multicast (PARTICIPANTS, (“write-lock-request", x, enumber (T)));
4. receive RESPONSES [1: k];
5. if all nodes from PARTICIPANTS replied with positive response then
6.    CURRENT-EPOCH-NODES:={participants that reported the epoch number equal to enumber (T)};
7.    OLD-EPOCH-NODES: = {participants that reported epoch numbers smaller than enumber(T)};
8.    multicast(OLD-EPOCH-NODES,(“do-update-and-adopt-new-epoch",x,new-value,enumber(T),elist (T)));
9.    multicast (CURRENT-EPOCH-NODES, (“do-update", x, new-value, enumber (T)));
10.else
11   if some node from PARTICIPANT replied with negative acknowledgement
12.      multicast (PARTICIPANTS, (“unlock-replica", x));
13.      abort the owner transaction;
14.  else
15.       multicast (PARTICIPANTS, (“unlock-replica", x));
16.       repeat the whole operation later or abort the owner transaction;
17.  endif
18. endif
```

5.3 Performance of writes operation by increasing replicated database

SGT time in millisecond increases near exponentially for a write operation at any replicated site (at site-1) if number of replicated database increase

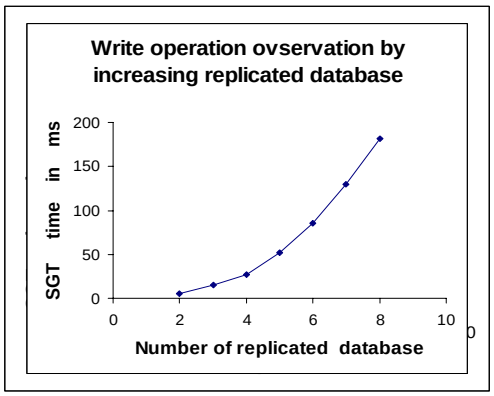

Fig 3: Observation of response time of a write operation at site-1 by increasing number of replicated database.

6. CONCLUSION

In ROWA protocol write can succeed if it can perform on all replicas of the data item. The read can succeed if it can perform on a single current replica of the data item. The data may be available for both read and write accesses even if only one operational the data item remains in the system. Another property of ROWA protocol is that multiple read operation at different replicated database sites are done synchronously without any delay and need not ordering.

From performance analysis of ROWA protocol, we can say that service getting time of read operation from nearest replicated database site is very less. But it takes larger time for write operation which is the major problem of ROWA protocol. So, we can say that ROWA protocol provide best performs if the application in only read only nature.

REFERENCES

[1] Sushant Goel, Rajkumar Buyya, "DATA REPLICATION STRATEGIES IN WIDE AREA DISTRIBUTED SYSTEMS", Grid Computing and Distributed System (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, 2006.
http://www.buyya.com/papers/DataReplicationInDS Chapter2006.pdf

[2]. Andrew S. Tanenbaum, Maarten van Steen, "DITRIBUTED SYSTEMS Principles And Paradigms", pp1-15 New edition 2005-2006.

[3]Michael Rabinovich and Edward D.Lazowska, "An Efficient and Highly Read-One-write-All Protocol for Replicated Data Management", Department of CSE, University of Washington, Seattle, WA 98195.
http://citeseer.ist.psu.edu/cache/papers/cs/2386

[4]. Abraham Silberschatz, Henry F.Korth and S.Sudarshan, "DATABASE SYSTEM CONCEPTS," pp 697-711, Fourth edition, (McGraw-Hill).

[5]. Stefano Ceri, Giuseppe Pelagatti, "DISTRIBUTED DATABASES Principles and Systems", pp 1-30, International edition, (McGraw-Hill).

[6] Creating and consuming a Web Service using Visual Studio .NET
http://www.xefteri.com/articles/show.cfm

[7] Michael Rabinovich and Edward D.Lazowska, "An Efficient and Highly Read-One-write-All Protocol for Replicated Data Management", Department of CSE, University of Washington, Seattle, WA 98195.
http://citeseer.ist.psu.edu/cache/papers/cs/2386