Comparative Study of Replication Techniques for Distributed Applications

Shilpa Nupur
Dept.of Computer Science & Electronics, AIM &
ACT Banasthali University, Tonk (Rajasthan), India

Dipanwita Thakur
Dept.of Computer Science & Electronics, AIM &
ACT Banasthali University, Tonk (Rajasthan), India

ABSTRACT
Distributed system is a collection of computers which communicate among themselves through message passing and each system has its own separate memory; however the user is made to believe that it is a single user system. This single system image is projected through transparency. Most of the applications are of distributed nature thus their performance is an important issue. Replication transparency provides both fault tolerance as well as better performance. Replication is a method of creating multiple copies of data at multiple sites to increase the availability of the system. It is mainly used in distributed systems and distributed databases. Replication is required for making a reliable, scalable and fault tolerant system. In case of distributed databases replication is done to enhance the performance. Intensive research is being carried out in this area and many replication techniques have been proposed so far. This paper presents a brief study of some of the replication techniques and their performance is compared.

Keywords
Distributed System, Distributed Database, Replication.

1. INTRODUCTION
Replication refers to the existence of number of replicas of a data item. It helps to make the system available and hence more reliable. Replication is carried out in distributed systems and databases for achieving fault tolerance and enhancing performance respectively. Its importance has recently been further increased because of the role it plays in achieving elasticity at the database layer. [1] Proper degree of replication must be chosen for a system. Low replication degree may degrade the performance whereas high degree of replication will make the maintenance of replicas costly. Replication control includes determining number and location of replicas of a replicated file. In a replication transparent system the replication control is handled entirely automatically in user transparent manner; this is also called as implicit replication. When the user is given the flexibility to control the replication process it is referred to as explicit replication. [2]

Two most common replication techniques are Active Replication and Passive replication. In Active replication the request from the client is multicast to all the replicas, the replicas process the requests independently starting from the state and send response to the client. The replicas are grouped together and act as state machines. Since they are state machines they all end up with same state as one another after each request. [3]Active replication guarantees quick response even if some of the replicas fail, thus the system performance is not hampered, and more over it can tolerate byzantine faults efficiently.

In Primary Backup or passive replication a primary server is associated with each data item x which is responsible for coordinating the read and write operations on x. All the red and write operations are carried out on a single server. Read operations are performed locally and all write operations are forwarded to a fix primary copy. As soon as the primary copy has updated the local copy of x it sends an acknowledgement to the backup servers to perform the updates. If the primary server fails, one of the backup servers takes its place. Passive replication may be used even for non deterministic processes. The disadvantage of passive replication compared to active is that in case of failure the response is delayed. [4] This paper provides a brief discussion of the replication techniques based on active replication and compare their performance.

2. REPLICATION TECHNIQUES

2.1 Practical Byzantine Fault Tolerance
This algorithm aims to tolerate byzantine faults in an asynchronous environment. It improves the response time with some optimizations. This algorithm is more practical as compared to previous algorithms which were primarily designed for synchronous environments. Since malicious attacks and software errors can cause faulty nodes to exhibit Byzantine (i.e., arbitrary) behavior, Byzantine-fault-tolerant algorithms are increasingly important. [5]
Castro presents this new algorithm for state machine replication for providing byzantine fault tolerance. It ensures safety and liveness provided at most n-1/3 out of a total of 'n' replicas are simultaneously faulty. This means that clients eventually receive replies to their requests and those replies are correct according to linearizability. [5]
Here byzantine failure model is used assuming independent node failures. The algorithm is implemented as state machine replication where the services are modeled as state machine. These state machines are replicated along different nodes in the distributed systems.
The replicas maintain service state and perform service operations. R is the set of replicas where R=3f+1, f is maximum number of faulty nodes.
Working of the algorithm is as follows:
1. A client sends request to primary for service invocation.
2. The primary multicasts the request to the backups
3. Replicas execute the request and provide the response to client
4. When the client receives a total of f+1 reply from all the replicas with same result, this is termed as the result of the operation

The replicas should be deterministic and should start in the same state, which is a basic requirement in every state machine replication.
In case the primary becomes faulty then the view change protocol assigns a new primary so that the system functions smoothly. Safety is provided if all the correct replicas follow the same sequence number of requests committed locally.
Three optimizations were used to reduce the communication cost. Replicas send the digest of the result instead of the result itself, which reduces the bandwidth consumption and CPU overheads. Second is minimizing message delays for invoking an operation. Third is to improve the performance of read only operations which does not change the service state.

The main advantage of this approach is its practical implementation in asynchronous environments, reduced response time and communication cost.
The major disadvantage is that it cannot mask software errors at all the replicas simultaneously. Number of replicas may be less to increase the performance.

2.2 Zyzzyva –Speculative Byzantine Fault Tolerance

Zyzzyva is a protocol which applies speculation to make the design of Byzantine Fault Tolerant state machine replication simple and cost effective. It is based on three sub-protocols: (1) agreement (2) view change (3) checkpoint .The agreement protocol orders requests for execution by the replicas. The view change protocol coordinates the election of a new primary when the current primary is faulty or the system is running slowly. The checkpoint protocol limits the state that must be stored by replicas and reduces the cost of performing view changes. [6]

The replicas send the response to clients' request before executing the three phase commit protocol for agreement on the order of requests processing. They follow the order proposed by the primary and respond back to the client instantaneously which may lead to inconsistencies in the replicas. The client identifies and resolves the consistencies by making the replicas merge on single total ordering of requests. This allows Zyzzyva to reduce replication overheads.

It uses a total of 3f+1 replica where f is number of faulty nodes (replicas or clients). The major steps of the agreement protocol are: client request is sent to the primary; request is received and assigned a sequence number and the ordered request is forwarded to the replicas. Upon receiving the request replicas execute them speculatively and send back the response to the client. Client collects the responses. A request is complete when client receives 3f+1 matching responses without any faulty nodes.
If the replicas are faulty the client may receive response from less than 3f+1 replica. Client receives between 2f + 1 and 3f matching responses, assembles a commit certificate, and transmits the commit certificate to the replicas. Replica receives a commit message from a client containing a commit certificate and acknowledges with a local-commit message. Client receives a local-commit messages from 2f + 1 replicas and completes the request.[6] If the client receives fewer than 2f+1 matching responses then it resends the request to all the replicas which is forwarded to the primary for assigning the request with a sequence number and ensuring its execution. If the responses display inconsistent ordering by primary then the client sends the information to replicas and asks for view change to replace the faulty primary.
Some optimizations were implemented to enhance the performance and to cut back the cost. These are separation of agreement from execution, batching of requests, caching of out of order requests, state of read only request not stored in the request history , execution response from single replica.

Zyzzyva has many advantages such as reduced costs, less number of replicas, simpler semantics, improved throughput, suitable for non deterministic services and ability to support high contention workloads.
Limitation of Zyzzyva is that a faulty primary makes the request processing slow. High contention workloads introduce extra latency.

2.3 BASE Using Abstraction to Improve Fault Tolerance

BASE makes use of abstraction to reduce the cost and enhance the ability to mask software errors in Byzantine Fault Tolerance algorithms. It reuses off-the-shelf service implementation. Availability is improved as each faulty replica is repaired periodically using an abstract view of the state stored by the correct replicas. Probability of common mode failures are reduced as each replica runs different service implementation. This technique is used in practice and has better performance than the implementations reused.

BASE stands for BFT with Abstract Specification Encapsulation. [7] It is an extension of BFT library. BASE combines BFT with data abstraction which aids in masking of software errors. A total of 'n' replicas are used for running different off-the-shelf implementations to avoid simultaneous failures. Although these implementation have same functionality but a different specification is used for each one of them with the help of distinct service states. To achieve deterministic behavior a common abstract specification is defined. These specifications are treated as black boxes, which enables the use of existing implementations.

The next step is to implement the conformance wrapper. The conformance wrapper implements the common specification. The last step is to implement the abstract function and its inverse. It is required for state transfer among the replicas and for repairing the faulty replicas.

BASE provides advantages such as reuse of existing code, software rejuvenation through proactive recovery and opportunistic N-version programming.

Disadvantages of this technique are: code complexity would introduce new bugs. Interface is too narrow, unexpected behavior.

3. OTHER REPLICATION TECHNIQUES

3.1 Byzantine Fault-Tolerant Deferred Update Replication

Deferred update replication is used for database replication. Pedone et al [8] describes this method for tolerating byzantine faults. This method has better throughput and scalability than state machine replication primary backup replication as all servers behave as primary and execute the transactions locally then the transaction modifications are forwarded to all the servers.

One of the important features of this technique is that read-only transactions are executed through a single server without communicating with other servers. This attribute helps in reducing latency and increase scalability with number of servers in the system.

In deferred update replication, transactions pass through two phases the execution phase and the termination phase. The execution phase starts when the client issues the first transaction command; it finishes with a client's request to commit or abort the transaction, when the termination phase starts. The termination phase finishes when the transaction is committed or aborted. A total of 3f+1 replica are used as in BFT algorithms. [8]

Advantages of this technique are reduced latency, improved scalability and faster read-only transactions.
Disadvantages: snapshot isolation cannot be used easily in this method, strict certification process to ensure consistency

3.2 Replication Degree Customization for High Availability

Object replication is a conventional method to improve the availability of distributed data intensive services and storage systems. These systems have highly skewed object request probability distributions. Zhong et al [9] describes an object replication degree customization scheme that maximizes the expected service availability under given object request probabilities, object sizes, and space constraints. It says that the optimal replication degree of an object should be linear in the logarithm of its popularity-to-size ratio. [9] When the theoretical value of data object popularity distribution is known the proposed customization increases the system availability to uniform replication. Results show that the scheme requires less amount of replica creation or removal overhead.

Object replication is generally applied in data centric services on wide area networks and wireless networks for increasing the system availability. While assessing the object replication degree the object request popularity is not being considered in the previous availability oriented schemes. Usually all objects are assigned equal number of replicas but due to highly distorted data popularity distribution the system availability may be improved using more number of replicas for popular objects and vice versa. This may enhance overall expected service availability with space cost remaining constant.

In this paper the scheme described is applied on two common distributed system environments: decentralized wide area networks and centrally managed local area clusters. Object sizes are also considered when object popularity and size distributions are correlated.

The optimal number or replicas for each object i to maximize availability of the system is given by

$Ki = K + Ds||r \cdot logp1/2+ \log 1/p\ ri\ /si,\ 1 \leq i \leq n$ [9]

Here k_i is number of objects used for object i, p is machine failure probability, ri request probability of object I, si size of object i (normalized by total data size), n number of objects.

The scheme is advantageous as availability is more in comparison to uniform replication, use same amount of storage space.

Disadvantages: It is not suitable for write intensive applications, more replica adjustment and maintenance overhead.

4. COMPARISON OF REPLICATION TECHNIQUES

The above discussed replication techniques are compared on basis of some parameters such as Availability, Liveness, Scalability, Latency and Throughput. The table displays which parameters are covered by the different replication techniques discussed in the previous sections. A rough estimate is done to assess which of the techniques cover all these parameters based on which further work is to be carried out on one of the technique having the best performance among these techniques.

Table 1. Comparative study of Replication Techniques

Replication Techniques	Availability	Liveness	Scalability	Latency	Throughput
Practical Byzantine Fault Tolerance	No	Yes	No	Yes	No
Zyzzyva: Speculative Byzantine Fault Tolerance	No	Yes	No	Yes	Yes
BASE Using Abstraction to Improve Fault Tolerance	No	Yes	No	No	Yes
Byzantine Fault-Tolerant Deferred Update Replication	Yes	Yes	Yes	Yes	No
Replication Degree Customization For High Availability	Yes	No	No	No	No

5. CONCLUSION

The replication techniques discussed in the paper are namely Practical Byzantine Fault Tolerance, Zyzzyva – Speculative Byzantine Fault Tolerance, BASE Using Abstraction to Improve Fault Tolerance Byzantine Fault-Tolerant Deferred Update Replication, and Replication Degree Customization for High Availability. First three techniques are active replication techniques which aim to mask byzantine faults. Deferred update replication also reduces the byzantine fault and it is more scalable than the active and primary backup replication techniques. The other technique explains object replication customization to improve the service availability of the distributed systems which deals with high volume of data.

To compare these techniques five parameters are considered that is Availability, Liveness, Scalability, Latency and Throughput.

Here a rough estimate is done to assess which of the techniques cover all these parameters based on which further work is to be carried out on the technique having the best performance among these. design and implementations Pg 173-186 Practical Byzantine Fault Tolerance.

6. REFERENCES

[1] Kemme B., Alonso G 2006 Database Replication a Tale of Research across Communities.

[2] Sinha P K, PHI, 1997 Distributed Operating Systems.

[3] Coulouris G, Dollimore J, Kindberg T Pearson Education, 4th ed 2001 Distributed Systems Concepts and Design.

[4] Tanenbaum Andrew S., van Steen M., Pearson Education.2002 Distributed Systems Principles and Paradigms.

[5] Castro M, Liskov B in proceeding OSDI'99 Proceeding of the third symposium on Operating Systems

[6] Kotla R., Alvisi L., Dahlin M., Clement A. and Wong E. 2009 Zyzzyva: Speculative Byzantine Fault Tolerance.

[7] Rodrigues R., Castro M, and Liskov B. In ACM Transactions on Computer Systems (TOCS) Volume 21 Issue 3, August 2003 Pg 236-269 BASE: Using Abstraction to Improve Fault Tolerance.

[8] Pedone F, Schiper N, Enrique Armend´ariz-I˜nigo J 2010 Byzantine Fault-Tolerant Deferred Update Replication

[9] Zhong M., Shen K., Seiferas J., 2008 Replication Degree Customization for High Availability.