Design of Distributed Voting Replication Model for Replica Control*

Kyeongmo PARK
Computer Science and Information Engineering
The Catholic University of Korea, ETRI
Gyeonggi, 420-743, South Korea

and

Miyoung LEE and Han NAMGOONG
Cloud Computing Research Department, Software Laboratory
Electronics and Telecommunications Research Institute (ETRI)
Daejeon, 305-700, South Korea

ABSTRACT
The objective of replication in data-centric distributed systems is to improve a service's performance, to increase its availability, or to make it fault-tolerant and reliable. In this paper, we present a replication extension model based on quorum-based voting that enforces consistent operations in a distributed system. The model uses a hybrid combing the static and dynamic voting, to configure the trade-off between availability and replica consistency. We contribute with an enhanced voting replication system with fault tolerance in the presence of server and network failures. The performance of our voting replication protocol is evaluated. As a part of the experimental studies, the effects of varying the voting frequencies and synchronization delays versus voting trade-off are examined. The effects of the latency of operations are also studied. Our preliminary findings are presented.

Keywords: Quorum, Voting, Replication, Replica Control, Fault Tolerance, Distributed Systems

1. INTRODUCTION
Replication is a technique used widely for enhancing services of distributed systems. This technique is a key to providing high availability, performance and fault tolerance in distributed systems. The objective of replication is to improve a service's performance, to increase its data availability, or to make it fault-tolerant and reliable. Replication is providing multiple identical instances of the same system or subsystem, directing requests to all of them in parallel, and choosing the correct result on the basis of a quorum. One of the major problems in distributed systems is maintaining the integrity and consistency of replicated data.

In this paper, we present an enhanced replication model based on quorum-based voting that enforces consistent operations in the distributed system. The system must have the capability to support many users without sacrificing performance. Higher availability and fault tolerance in the presence of server and network failures are also requirements of mission-critical and other important systems. These requirements are often at odds with each other, resulting in a balance and fair trade-off between performance, availability, and consistency. The trade-off between consistency and availability or performance has been a hot design issue for many years.

Traditional pessimistic replication systems are based on the principle of one-copy serializability proposed by Bernstein and Goodman [1], i.e., users should observe the system to behave as if there was only one copy of the data. Strong single-copy consistency provides the most desirable correctness and the most favorable guarantees. Examples of strong consistency include quorum consensus [2], primary-copy algorithms [3], and atomic broadcast protocols [4]. However, strong constraints on system integrity seriously limit the availability or performance.

Many replication systems take optimistic replication [5], also known as lazy replication [6], in which replicas are allowed to diverge. Optimistic replication does away with this in favor of eventual consistency, meaning that replicas are guaranteed to converge only when a system is idle. Replication systems, such as Bayou [7] and Coda [8] provide single-copy availability and non-serialized performance at the expense of data consistency and they guarantee eventual consistency.

The remainder of this paper is organized as follows. Our system model is introduced in Section 2. Section 3 explains a hybrid replica control method combining static and dynamic voting. The simulation results in Section 4 evaluate the voting performance under different access patterns. Finally we conclude in Section 5.

2. SYSTEM MODEL
We consider data-centric distributed systems with a small number of host server nodes. Server nodes host objects which are replicated to other server nodes to achieve fault tolerance. Both node and link failures, network partitions are considered.

We assume a partially synchronous system, where clocks are not synchronized, but message time is bound. A group membership service is assumed in the system, which provides a single view of the nodes within a partition. We assume the presence of a group communication service which provides multicast to groups with configurable delivery and ordering guarantees.

One of the goals in this work is to provide fault tolerance through selective agent replication. Multiagent applications

---

* This work was supported by the IT R&D program of MIC/IITA, Korea (2007-S-016-02, Development of Cost Effective and Large Scale Global Internet Service Solution).
reply on the collaboration among agents. If one of the involved agents fails, the whole computation can get damaged. The solution to this problem is replicating specific agents. Replication may often be expensive in both computation and communication. A software element of the application may lose at any point in progress. It is important to be able to go back to the previous choices and replicate other elements.

In Gifford’s quorum-based voting for replica control [11], a generalization of majority voting as originally proposed by R.H. Thomas [12], each copy of a replicated data item is assigned a vote. Each operation then has to obtain a read quorum \( V_r \) or a write quorum \( V_w \) to read or write a data item, respectively. If a given data item has a total of \( V \) votes, the quorums have to obey the following rules:

\[
V_r + V_w > V \quad (1)
\]

\[
V_w > \frac{V}{2} \quad (2)
\]

The first rule ensures that a data item is not read and written by two transactions concurrently avoiding the read-write conflict. The second rule ensures that two write operations from two transactions cannot occur concurrently on the same data item avoiding the write-write conflict. Thus the two rules ensure that one-copy serializability is maintained. For simplicity, we assume that all replicas have equal votes of the same value one and each node in the system hosts one replica. The total number of votes \( V \) is the total number of nodes in the system.

In static quorum techniques, such as weighted voting [11], it is possible to balance the cost of read against write operations by changing the sizes of the read and write quorums appropriately. In response to network failures the quorums are not reconfigured. Intervention is not necessary, when nodes recover and network partitions rejoin.

Static voting blocks operations if quorums cannot be built. However, as discussed in Section 1, in some systems, constraint consistency can be temporarily relaxed during degraded situations. The traditional voting is enhanced by allowing non-critical operations even if no quorums exist. Operations that violate tradable constraints but do not affect non-tradable constraints are allowed.

Non-tradable constraints mean that they cannot be traded for higher availability during degradation. Tradable constraints can be temporarily relaxed during degraded situations. The adaptive approach to quorum consensus for balancing data availability with replica consistency is called Adaptive Voting (AV) [14]. The replication system modes of operation can be described as the three modes: normal mode, degraded mode, and reconciliation mode.

3. A HYBRID REPLICA CONTROL METHOD

A quorum is the minimum number of votes that a distributed transaction must obtain to be allowed to perform an operation in a distributed system. A quorum-based approach is implemented to enforce consistent operation in a distributed database system.

Quorum-based voting can be used as a replica control method, as well as a commit method to ensure transaction atomicity in the presence of network partitions [9, 10]. In replicated databases, a data object has copies present at several sites. To ensure serializability, no two transactions should be allowed to read or write a data item concurrently. In case of replicated databases, a quorum-based replica control protocol can be used to ensure that no two copies of a data item are read or written by two transactions concurrently.
updates, the version histories need to be calculated based on the version histories of the nodes. Either stepwise rollbacks or repair actions are again performed to establish data integrity in case of consistency violations. The quorums are re-adjusted according to the size of the merged partition and the histories are cleaned up.

4. PERFORMANCE EVALUATION

To explore voting performance issues, we have performed experiments. The system we tested consists of two-node Pentium 4 PCs with 3.2 GHz CPU and 3GB RAM. An agent moving from node to node was simulated by sending a message across these nodes. We looked at behavior of 1, 4, 8 replicas and examined the effects of varying voting frequencies with uniform and non-uniform and synchronization delays. The effects of the latency of operations are also studied.

In this experiment, we examine the cost of voting in the case that host speeds are uniform and that how synchronization delay can be amortized by voting less frequently. Agents visited a sequence of $N$ hosts before voting, rather than voting at the end of each stage.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig2.png}
\caption{Time per host visited vs. number of hosts visited with various frequencies.}
\end{figure}

Fig. 2 is the graph of the average time per host visit when $N$ ranges from 1 to 32. The x-axis is the number of hosts visited between votes and the y-axis is time per host visit, normalized to speed of a single replica. The data depicted reports averages from run 160 rounds. The time spent per host had a variance .2 percent. We found improvements as $N$ advanced from 1 to 8. For $N$ greater than 8, the further improvements were not significant. It is interesting to note synchronization delay versus voting trade-off. As voting are infrequent, replica completion time drift apart and the synchronization delay increase.

A voter need wait for a correct majority, so a vote-delimited stage will complete as soon as the median correct replica votes. Therefore, the completion time for a replicated computation that votes infrequently should approximate the completion time when there is a single replica.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig3.png}
\caption{Time for delay vs. number of replicas under uniform and non-uniform delays.}
\end{figure}

Fig. 3 shows the performance of an agent voted five moves. The probability of encountering a slow host to 1 percent was set. The addition of replicas further reduces the slowdown, and such a host was 195 times slower than a normal host. Voting performance under uniform and non-uniform delays is shown in this figure.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig4.png}
\caption{Comparison between three operations for the latency ROWA in normal mode.}
\end{figure}

In this experiment, we have measured the latency of three operations in normal mode: (i) a write operation with constraint checking (CC), (ii) a write operation without CC, and (iii) a read operation for different node numbers.

The performance of the hybrid method is evaluated using the following scenario. The state of an object a/b of class A/B is represented by an integer value $x$. Three constraints are:

\begin{align*}
C-1: & \quad a.x < \text{constant-1} \\
C-2: & \quad b.x < \text{constant-1} \\
C-3: & \quad a.x + b.x < \text{constant-2}
\end{align*}

$V_w$ = number
of nodes, $V = 1$. The voting applies a write operation locally first and then propagates the whole object state of 1 kilobyte to a write quorum of replicas. Write operations with CC are slower than write operations without CC since they involve read operations as well.

However, in case of ROWA, the difference is negligible since the reads can be performed locally. A local read requires about 1.5 milliseconds. Read operations can always be performed locally in a ROWA scheme while latency of write operations directly depends on the number of nodes. ROWA is best for read-intensive applications.

5. CONCLUSIONS

Replication is a key to providing high availability, performance and fault tolerance in data-centric distributed systems. We have presented a replication extension model based on quorum-based voting, in which a hybrid combing the static and dynamic voting is used to configure the trade-off between system availability and replica consistency. This requires kinds of optimistic replication protocols that allow replicas to diverge during degradation to achieve higher availability. We have discussed the concept of the voting replication model that supports the configuration of the trade-off and have evaluated our replication protocols by simulation.

As a part of the experimental studies, the effects of varying the voting frequencies and synchronization delays versus voting trade-off have been examined. As voting are infrequent, replica completion time drift apart and the synchronization delay increase. Synchronization delays by voting could be made insignificant by making voting less frequent, replica completion time drift apart and the synchronization delays by voting could be made insignificant by making voting less frequent. Voting replication improve performance by ensuring that slow hosts do not make progress difficult. For the effects of the latency of operations, in short, a read-one, write-all scheme performs best for read operation. Future work includes more experiments on the performance of the replication and reconciliation approach.

REFERENCES