ENHANCED TREE QUORUM ALGORITHM FOR REPLICATED DISTRIBUTED DATABASES

Soon M. Chung
Department of Computer Science and Engineering
Wright State University
Dayton, Ohio 45435, U.S.A.

Abstract

In this paper, a new replica control algorithm, called the enhanced tree quorum (ETQ) algorithm, is proposed to manage replicated data in a distributed database system. This algorithm provides high availability for read and write operations by imposing a logical structure of modified tree with a backup root on data copies. With ETQ algorithm, a read operation is limited to a data copy in the best case, and a write operation is allowed as long as one of the roots and the majority of the children of each node selected are available in the tree. Compared to other algorithms, ETQ requires lower message cost for an operation, while providing higher availability.

1. Introduction

A distributed system consists of a set of computers (called sites) that communicate through a network. A distributed system differs from a centralized system in the nature of failures. The components of a centralized system typically depend on one another to the extent that the failure of a single component disables the entire system. On the other hand, the components of a distributed system are autonomous computers, and the failure of an individual site does not disable the entire system. The other advantage of distributed systems is that valuable data can be stored redundantly at multiple sites, which is commonly called replication of data.

Replication is a useful technique for distributed database systems where reliability is important, such as banking systems, airline reservation systems, etc. Rather than keeping one copy of important data at a single site, multiple copies of the same data can be maintained at different sites. Replication increases the data’s availability: if one copy is temporarily inaccessible, then the transaction can be continued using a different copy. Replication also increases the data’s reliability: if one copy is accidentally destroyed, it can be reconstructed from the other copies. The performance of an operation can be enhanced by permitting the authorized user to work on the copy that can be most easily accessed.

In a replicated database, each read and write operation issued by a transaction on some logical data item must be mapped by the database system to the corresponding operation on physical copies. To be correct, the mapping must ensure that the concurrent execution of transactions on replicated data is equivalent to a serial execution on non-replicated data—a property known as one-copy serializability [BER85]. Multiple copies of a data item should appear as a single copy to the transaction. This requirement is enforced by the replica control algorithm.

In order to ensure the requirement of one-copy serializability, a replicated data item x may be read by reading a quorum of copies, and it may be written by writing a quorum of copies. The quorum for an operation is defined as a set of copies whose number is sufficient to execute that operation. The selection of a quorum is restricted by the quorum intersection property: for any two operations O1(x) and O2(x) on a data item x, where at least one of them is a write, the quorums must have a non-empty intersection. The correctness of our enhanced tree quorum algorithm will be proved by using this property.

Many replica control algorithms have been proposed: Voting algorithms [GIF79, THO79] become popular because they are flexible and easily implemented. Dynamic replica control methods [BAR86, JAJ87a, JAJ87b, JAJ88, PAR88] are derived from voting to achieve high data availability. Available copies [BER84, LON87] and regeneration methods [PU86] achieve even higher data availability using the same number of copies. They do not have to trade off the read availability with the write availability, but the data consistency is not guaranteed when the network is partitioned. To reduce the storage space for copies, voting with witnesses [PAR86] can be used, but the operation availability will be compromised. The tree quorum algorithm [AGR90a, AGR90b] uses quorums that are obtained from a logical tree structure imposed on data copies, and the multiple tree quorum algorithm [CHU92] is based on multiple logical trees imposed on the data copies. Compared to the tree quorum algorithm, the multiple tree quorum algorithm provides higher availability of the write operation, especially when the network is partitioned. A comprehensive survey of replica control algorithms is presented in [DAV85].

The problem of finding quorums to optimize the performance has been addressed by several researchers. Garcia-Molina and Barbara introduced the method that can be used to generate a subset of the quorums which includes all quorums obtained from vote assignments [GAR85]. In [CHE90], Cheung, Ahamad, and Ammar presented an algorithm for generating all vote and quorum assignments that need to be considered in optimizing the read and the write operations on replicated data. Kumar described a hierarchical quorum consensus algorithm [KUM90], that organizes a group of copies into a multi-level hierarchy and extends the quorum consensus algorithm to such an environment. The algorithm for finding a multi-dimensional vote assignment is proposed by Cheung, Ahamad, and Ammar in [CHE90].

This paper is organized as follows: In Section 2, we review the basic replica control algorithms; read-one/write-all, available copies, voting, and tree quorum algorithms. In Section 3, the enhanced tree quorum algorithm is proposed. In Section 4, the performance of the proposed algorithm is analyzed in terms of message cost and availability, and a comparison with other algorithms is given.

† This research was partially supported by National Science Foundation under grant No. IRI 9008694.
2. Review of Replica Control Algorithms

In this section, we review some of the basic replica control algorithms: read-one/write-all, available copies, voting, and tree quorum algorithms, which will be compared with the proposed enhanced tree quorum algorithm.

2.1. Read-One/Write-All

The simplest technique to maintain replicated data is that a read operation is allowed to read any copy, and a write operation is required to write all copies of the data item, which is called read-one/write-all (ROWA) algorithm. This algorithm works correctly since a transaction processes from one correct state to another correct state. The ROWA has the lowest read cost because only one copy is accessed by a read operation. A weakness of this method is the low write availability because a write operation cannot be done at the failure of any copy.

The available copies algorithm [BER84, BER87] is an enhanced version of the ROWA approach in terms of the availability of write operations. Every read is translated into a read of any copy of the data item, and every write is translated into writes of all available copies of that data item. This algorithm handles site failures but not communication failures because they assume that each site is either operational or down and that all operational sites can communicate with each other. Therefore, each operational site can independently determine which sites are down, simply by attempting to communicate with them. If a site does not respond to a message within the timeout period, then it is assumed to be down.

2.2. Voting

The first voting algorithm was the majority consensus algorithm proposed in [THO79]. Here, we describe the generalization of the algorithm proposed in [GIF79]. In this approach, every copy of a replicated data item is assigned a certain number of votes, and a transaction has to collect a read quorum of \( r \) votes to read a data item, and a write quorum of \( w \) votes to write a data item. Quorums must satisfy two constraints: \( r + w \) must be larger than the total number of votes \( v \) assigned to the copies of the data item, and \( w > v/2 \).

The first constraint ensures that there is a non-empty intersection between every read quorum and every write quorum. Therefore, each read quorum is guaranteed to have a current copy of the data item. The second constraint ensures that writes can happen in two different partitions of copies of the same data item. Both constraints together guarantee that each write quorum of a data item has at least one copy in common with every read quorum and every write quorum of the data item.

To help the transaction figure out which copy is up-to-date, we tag each copy with a version number, which is initially 0. When a transaction performs a write, it obtains the current maximum version number of the data, adds one to it, and tags all of the copies that it writes with the new version number. The version number shows how up-to-date each copy is. Each read returns the version number along with the data value, and the transaction selects the copy with the largest version number in the read quorum.

The voting approach ensures the one-copy serializability and works well for the failure caused by the site crash or the network partition. By choosing \( r > v/2 \), a data item is read accessible in only one partition, where it is write accessible. The read availability is given a higher priority by choosing a small \( r \). A weakness of this scheme is that writing a data item is fairly expensive: A write quorum of copies must be larger than the majority of votes, i.e., \( w > v/2 \).

2.3. Tree Quorum Algorithm

The tree quorum (TQ) algorithm [AGR90a, AGR90b] imposes a logical tree structure on the set of copies of a data item. One advantage of this algorithm is that a read operation may access only one copy and the number of copies to be accessed by a write operation is always less than the majority of the copies. Thus, the tree quorum algorithm is much better in terms of the write availability than the read-one/write-all algorithm. Since the logical tree structure does not have to correspond to the physical structure of the network, no reconfiguration is required at the failure of sites.

To explain the selection of read and write quorums, a logical tree with 13 copies is shown in Figure 1. For read operations, a read quorum can be formed by the root or the majority of the children of the root. If any node in the majority of the children of the root fails, it can be replaced by the majority of its children, and so on recursively. In the best case, a read quorum contains only the root, \( \{1\} \). As the root fails, a quorum is formed by the majority of the nodes at level 1, e.g., \( \{2, 3\}, \{3, 4\}, \) or \( \{2, 4\} \). Node 2 or node 3 is replaced by the majority of their children, respectively, if no majority at level 1 is accessible and only node 4 is accessible. Such quorums are \( \{5, 6, 4\} \) or \( \{8, 10, 4\} \). If the nodes at level 0 and level 1 fail, a quorum is formed by the majorities of the children of the selected majority at level 1, e.g., \( \{5, 6, 8, 9\} \) or \( \{8, 9, 11, 13\} \).

For write operations, a write quorum is formed by the root, and the majority of the children of the root, and the majorities of the children of the selected children of the root, etc. The size of a write quorum for a given tree is fixed, but the members can be different. Possible write quorums are \( \{1, 2, 3, 5, 6, 8, 9\} \), \( \{1, 2, 4, 6, 7, 12, 13\} \), etc.

![Figure 1. A Tree Imposed on Data Copies](image-url)
choosing a small read quorum and a large write quorum. The tree quorum method relaxes one constraint of voting. It allows a write quorum can be smaller than the majority of all copies. The fault tolerance is increased if the nodes at the low levels of the tree have less failures. In the next section, the enhanced tree quorum algorithm, based on the tree quorum algorithm, is proposed. The new algorithm has higher read and write availabilities because the failure of the root is tolerated.

3. Enhanced Tree Quorum (ETQ) Algorithm

In this section, we propose the enhanced tree quorum (ETQ) algorithm for managing replicated data items in distributed database systems. In this approach, selection of read and write quorums for a data item is based on a logical tree with a backup root imposed on the copies of the data item. Instead of counting votes, a tree-based protocol is used to select quorums.

Here we assume that a distributed system consists of a set of sites that communicate with each other by sending messages over a communication network. No assumptions are made regarding the speed, connectivity, or reliability of the network. We also assume that sites are fail-stop [SCH83] and communication links may fail to deliver messages. Combinations of such failures may lead to partitioning of the network, where sites in one partition may communicate with each other, but no communication can occur between sites in different partitions. A site may become inaccessible due to site failures or network partitioning.

With ETQ algorithm, copies of a data item are logically organized into a modified tree (with a backup root) of height $h$ and degree $D$, where $D$ is the degree of nodes in the tree. Figure 2 shows a modified tree of height 2 and degree 3 imposed on 14 copies.

![Figure 2. A Tree with a Backup Root Imposed on Data Copies](image)

3.1. Constructing a Read Quorum and a Write Quorum

For a read operation, a read quorum can be formed by one of the roots of the modified tree, or by replacing failed roots with the majority of their children, and so on. In other words, when the two roots fail, a read quorum is obtained for the tree in the same manner as in TQ algorithm. However, the size of the read quorum is one if at least one of the roots is available. For a write operation, a write quorum can be formed by available roots and the majority of children of the roots, and so on. In other words, all the available roots are selected, and the nodes at lower levels of the tree are selected in the same manner as in TQ algorithm. Figures 3 and 4 show the algorithms for constructing the read and the write quorums, respectively.

![Figure 3. Read Quorum Construction in ETQ Algorithm](image)
Function WriteQuorum(Tree): QUORUM;
var
  MajorityQuorum, Majority: QUORUM;
begin
  if Empty(Tree) then
    return (\{\});
  else if at least one of the roots is accessible then
    MajorityQuorum = \{all the available roots\};
    (* collect the majority of subtrees of the roots *)
    for each selected subtree ∈ Majority
      MajorityQuorum = MajorityQuorum \\n      \text{WriteQuorum(Subtree)};
    if unable to collect Majority then
      return (\{\});
    else return (MajorityQuorum);
  else return (\{\});
end;

Figure 4. Write Quorum Construction in ETQ Algorithm.

3.2. Correctness of the Enhanced Quorum Algorithm

The enhanced tree quorum algorithm is based on the tree quorum algorithm. Therefore, the correctness of ETQ can be proved based on the correctness of TQ. With the following theorem proved, we demonstrate that a read quorum and a write quorum constructed by ETQ algorithm always have a non-empty intersection. Note that two write quorums always have a non-empty intersection.

**Theorem:** The enhanced tree quorum algorithm guarantees a non-empty intersection between a read quorum and a write quorum.

**Proof:** The proof is by induction on the height of the tree.

**Basis:** The theorem holds for a tree of height zero since there are only roots in the tree. A read quorum contains an available root and a write quorum contains all the available roots. So both quorums have a non-empty intersection.

**Induction Hypothesis:** Assume that the theorem holds for a tree of height \( h \).

**Induction Step:** Consider a tree of height \( h + 1 \). The read and write quorums constructed for this tree have the following form:

1. Read Quorum:
   \( \{\text{one of the roots}\}, \text{ or } \{\text{read quorums of the majority of the subtrees of height } h\} \).

2. Write Quorum:
   \( \{\text{available roots and the write quorums of the majority of the subtrees of height } h\} \).

If a read quorum consists of one of the roots of the tree, it is guaranteed to have a non-empty intersection with any write quorum. If a read quorum consists of the read quorums of the majority of the subtrees of height \( h \), it is guaranteed to have at least one subtree in common with any write quorum. Since the subtrees are of height \( h \), the induction hypothesis guarantees that read and write quorums have a non-empty intersection. Hence, by induction, ETQ algorithm guarantees a non-empty intersection between a read quorum and a write quorum.

The tree quorum and the enhanced tree quorum algorithms use the same principle of imposing a logical structure to improve the write availability over ROWA and to reduce the quorum size compared to voting. In addition, ETQ allows to construct a write quorum on a modified tree even though one root fails, while TQ does not allow to write when the root fails. Thus the proposed ETQ algorithm enhances the fault-tolerance in write operation compared to TQ. For read, the minimum read quorum size of TQ is \( \lceil (D+1)/2 \rceil \) when the root fails and the degree of nodes is \( D \). However, it is still one with ETQ if the other root is available.

4. Performance Analysis and Comparison

In this section, we analyze and compare the performance of the enhanced tree quorum algorithm and other replica control algorithms: read-one/write-all, voting, and tree quorum, on the message cost and the operation availability.

4.1. Message Cost Analysis

The message cost of an operation is directly proportional to the size of the quorum required to execute the operation. Therefore, we represent the message cost in terms of the quorum size. In addition, we use \( C_X^Y \) to denote the message cost with \( X \) algorithm for \( Y \) operation, which is \( R \) (read) or \( W \) (write).

4.1.1. Read-One/Write-All (ROWA)

In ROWA, a read operation needs only one copy, while a write operation needs to access \( N \) copies if there are \( N \) copies in the system. Thus, the message cost of a read operation is:

\[ C^R_{\text{ROWA}} = 1 \]

and the message cost of a write operation is:

\[ C^W_{\text{ROWA}} = N \]

4.1.2. Voting (VT)

In voting, if the majority consensus method is used, the quorum size for both read and write operations can be the same, i.e., the majority of the total number of votes assigned to the copies. Thus, the cost \( C^R_{\text{VT}} \) and \( C^W_{\text{VT}} \) are:

\[ C^R_{\text{VT}} = C^W_{\text{VT}} = \left\lceil \frac{N+1}{2} \right\rceil \]

where \( N \) is the total number of votes assigned to copies.

If the weighted voting method is used, the cost is dependent on the number of votes preselected for read and write operations.

4.1.3. Tree Quorum (TQ)

In estimating the cost of operations with TQ, \( h \) denotes the height of the tree, \( D \) is the degree of nodes in the tree, and \( M \) is the majority of \( D \), i.e.,
In TQ, the cost of a read operation ranges from 1 to \( M^h \) \[AGR90a\]:

\[ 1 \leq C_{\text{TQ}} \leq M^h \]

When the root is accessible, the read quorum size is 1. As the root fails, the majority of its children replace it, thus the quorum size increases to \( M \). Therefore, for a tree of height \( h \), the maximum quorum size is \( M^h \).

In TQ, all the write quorums have the same size: the root is selected and the majority of the children of each node selected are included, recursively. Thus, the cost of a write operation, \( C_{\text{TQW}} \), can be represented as:

\[ C_{\text{TQW}} = \sum_{i=0}^{h} M^i \]

4.1.4. Enhanced Tree Quorum (ETQ)

The size of a read quorum in ETQ is the same as that of TQ. Thus, it varies from 1 to \( M^h \):

\[ 1 \leq C_{\text{ETQ}} \leq M^h \]

For a write operation, the two roots are accessed and the majority of the children of each node selected are accessed, recursively. Thus, the cost of a write operation can be represented as:

\[ C_{\text{ETQW}} = 2 + \sum_{i=1}^{h} M^i \]

4.1.5. Comparison of Costs

Figures 5 and 6 show the read and the write costs of the four algorithms, respectively, for different total number of copies. We plot the maximum read costs for TQ and ETQ in Figure 5, and for voting, we choose the read quorum and the write quorum as the majority of the total copies. ROWA has the lowest cost for a read because only one copy is read. In the best case, TQ and ETQ also have the lowest read cost. In the worst case, the read costs for TQ and ETQ are almost the same. For example, for ETQ with a modified tree of height 2 on 14 copies, the maximum cost is 4, and for TQ with a tree of height 2 on 13 copies, the maximum cost is also 4.

In voting, if we choose the read quorum and the write quorum as the majority of the total copies, then the read cost is always higher than those of ETQ and TQ because \( M^h < \lceil (N+1)/2 \rceil \).

For a write operation, ROWA has the highest cost. The write cost of ETQ is almost the same as the cost of TQ: for 14 copies, the write cost of ETQ is 8, whereas for 13 copies, the write cost of TQ is 7. The write costs of TQ and ETQ are always lower than that of voting because the number of nodes selected at each level of the tree is less than the majority of the nodes at the level.

4.2. Availability Analysis

In this section, four replica control algorithms are analyzed and compared in terms of the operation availability. We first introduce the k-out-of-N model which will be used for estimating the operation availability.

4.2.1. The k-out-of-N Model

We define the availability of a data item as the probability of the data item being accessible for an operation at any given time, which can be represented using the k-out-of-N model [BAR84]. The assumptions of k-out-of-N model are repeated here:

1. The data item and its copies are in one of the two states: accessible or inaccessible.
2. The states of the copies are changed independently.
3. The data item is available for an operation if at least \( k \) of its \( N \) copies are accessible.

Thus, the k-out-of-N model can be formulated as:

\[ k\text{-out-of}-N = \sum_{i=k}^{N} \binom{N}{i} p^i (1-p)^{N-i}, \quad k \geq 1 \]

where \( p \) denotes the availability (accessibility) of a copy.

This formula gives the availability of a data item for an operation if \( k \) available copies are needed for the quorum. As \( k \) increases, the availability decreases because more accessible copies are required. By using the k-out-of-N model, we can estimate the availability of operations on a
data item given the availability of each copy of the data item. In estimating the availability of operations, all copies are assumed to have the same availability \( p \), and \( A_{XY} \) will represent the availability of \( Y \) operation with \( X \) algorithm.

4.2.2. Read-One/Write-All (ROWA)

For a data item replicated, ROWA requires a read on any one of the copies and a write on all copies. Therefore, the availability for a read operation can be represented as \( 1 - \text{out-of-N} \), and for a write operation as \( N \)-out-of-\( N \). Thus, the read availability \( A_{ROWA} \) is:

\[
A_{ROWA} = \sum_{i=1}^{N} p^i (1-p)^{N-i} = 1 - (1-p)^N
\]

and the write availability \( A_{ROWAW} \) is:

\[
A_{ROWAW} = \sum_{i=N}^{N} p^i (1-p)^{N-i} = p^N
\]

4.2.3. Voting (VT)

For \( N \) copies, voting allows \( N \) choices for the read and the write quorums from (read 1, write \( N \)) to (read \( N \), write 1). If we select a small read quorum, the read availability is high but the write availability becomes low, demonstrating the trade-off between the read and the write availabilities in VT. If we choose the read quorum of \( k \), the read availability \( A_{VT\text{R}} \) is:

\[
A_{VT\text{R}} = \sum_{i=k}^{N} \left( \begin{array}{c} N \\text{ \_i} \\ i \end{array} \right) p^i (1-p)^{N-i}, \ k \geq 1
\]

and the corresponding write availability \( A_{VT\text{W}} \) is:

\[
A_{VT\text{W}} = \sum_{i=N+1-k}^{N} \left( \begin{array}{c} N \\text{ \_i} \\ i \end{array} \right) p^i (1-p)^{N-i}
\]

4.2.4. Tree Quorum (TQ)

The availability of the read and the write operations in TQ can be estimated by using recurrence equations based on the tree height \( h \). Let \( R_h \) and \( W_h \) be the availability of the read and the write operations with a tree of height \( h \), respectively, while \( D \) denotes the degree of nodes in the tree and \( M \) is the majority of \( D \). Then, the availability of a read operation for a tree of height \( h+1 \) can be represented as:

\[
R_{h+1} = p + (1-p) \sum_{i=M}^{D} \left( \begin{array}{c} D \\text{ \_i} \\ i \end{array} \right) R_h^i (1-R_h)^{D-i}
\]

and the availability of a write operation for a tree of height \( h+1 \) is given as:

\[
W_{h+1} = p \sum_{i=M}^{D} \left( \begin{array}{c} D \\text{ \_i} \\ i \end{array} \right) W_h^i (1-W_h)^{D-i}
\]

where \( p \) is the probability that a copy is available, and \( R_0 \) and \( W_0 \) are equal to \( p \).

4.2.5. Enhanced Tree Quorum (ETQ)

We can use the equations developed for TQ to estimate the operation availability in ETQ because the read availability and the write availability of a subtree in ETQ are identical to those in TQ for the subtree. Let \( R_h \) and \( W_h \) denote the read and the write availabilities for a tree of height \( h \), respectively, and let \( E_R \) and \( E_W \) denote the read and the write availabilities in ETQ with a modified tree of height \( h \), respectively. Then, \( E_R_{h+1} \) can be represented as:

\[
E_{R_{h+1}} = [1 - (1-p)^2] + (1-p)^2 \sum_{i=M}^{D} \left( \begin{array}{c} D \\text{ \_i} \\ i \end{array} \right) R_h^i (1-R_h)^{D-i}
\]

The first term corresponds to the case that at least one of the two roots is available, and the second term is for the case that both roots are not available.

The availability of a write operation, \( E_{W_{h+1}} \), is

\[
E_{W_{h+1}} = [1 - (1-p)^2] \sum_{i=M}^{D} \left( \begin{array}{c} D \\text{ \_i} \\ i \end{array} \right) W_h^i (1-W_h)^{D-i}
\]

and

\[
E_{R_0} = E_{W_0} = 1 - (1-p)^2
\]

where \( p \) is the probability that a copy is available.

4.2.6. Comparison of Availabilities

The read and the write availabilities of the four methods, ROWA, VT, TQ, and ETQ, are compared in Figures 7 and 8, respectively. We assume that all data copies have the same availability, and that \( N = 14 \) for ROWA, VT, and ETQ, and \( N = 13 \) for TQ. Here, we use the same quorum size, \( \lceil (N+1)/2 \rceil \), for read and write in VT. The read availabilities of ETQ and TQ are much higher than that of VT because the number of data copies to be read in ETQ and TQ is much smaller than the majority of the copies, required in VT. ETQ has slightly higher read availability than TQ because in ETQ either one of the two roots can be a read quorum, whereas a read quorum in TQ will be the majority of the children of the root when it fails.

The write availability of ETQ is also higher than that of TQ because a write operation is not allowed at the root failure in TQ, whereas a write operation is allowed in ETQ as long as one of the two roots is available. The write availability of VT is lower than those of ETQ and TQ because the write quorum size of VT is selected as \( \lceil (N+1)/2 \rceil \), which is larger than the write quorums of ETQ and TQ almost all the cases.

Considering the operation costs and the availabilities, TQ and ETQ are better than ROWA and VT, and ETQ provides higher operation availability than TQ at the same operation cost.

![Figure 7. Comparison of the Read Availability](image-url)
In this paper, a new replica control algorithm, called the enhanced tree quorum (ETQ) algorithm, is proposed to manage replicated data in distributed database systems. This algorithm provides high availability for read and write operations by imposing a logical structure of a modified tree on data copies. With ETQ algorithm, a read operation is limited to a data copy in the best case, and a write operation is allowed as long as one of the roots of the modified tree and the majority of the children of each node selected are available. That means, no reconfiguration is required in ETQ for the case of site failures. Compared to other algorithms, ETQ requires lower message cost for an operation, while providing higher availability.

**References**


