A Highly Fault-Tolerant Quorum Consensus Method for Managing Replicated Data

Xuemin Lin[1] and Maria E. Orlowska[2]

[1] Department of Computer Science, University of Western Australia
Nedlands, WA 6907, Australia. e-mail: lxue@cs.uwa.oz.au.
[2] Department of Computer Science, The University of Queensland
QLD 4072, Australia. email: maria@cs.uq.oz.au.

Abstract. The main objective of data replication is to provide high availability of data for processing transactions. Quorum consensus (QC) methods are frequently applied to managing replicated data. In this paper, we present a new QC method. The proposed QC approach has a low message overhead: 1) In the best case, each transaction operation process needs only to communicate with $O(\sqrt{n}\log^{1-\frac{1}{2\log_3 2}} n)$ ($\approx O(\sqrt{n}\log^{0.208} n)$) remote sites ($n$ is the number of sites storing the manipulating data item). 2) In the worst case, each transaction operation process may be forced to communicate with $O(\sqrt{n}\log^{\frac{1}{2\log_3 2}} n)$ ($\approx O(\sqrt{n}\log^{0.792} n)$) remote sites. Further, we can show that the proposed QC method is highly fault-tolerant. The proposed approach is also fully distributed, that is, each site in a distributed system bears equal responsibility.

Key words: concurrency control, distributed computing, fault tolerance, quorum consensus method.

1 Introduction

Through the replication of data, distributed database system reliability can be increased if an effective approach is found for managing replicated data. Meanwhile mutual consistency among the replicated copies of data should be maintained by synchronizing transactions at different sites, so that a global serialization order can be ensured. Thus, the problem of replicated data management involves a compromise between two conflicting goals: maximizing data availability and maintaining the consistency of data. *Quorum consensus* (QC) methods [2,3] are widely used in managing replicated data.

Using a QC method, an operation of a transaction issued at a site in a distributed database system can proceed only if permission is granted by a group of other sites storing the replicas of the manipulating data. A general protocol of a QC method for processing a transaction operation can be described as follows.

Given a data item (object), at each site s_i, the regulations for forming a read quorum group S_i^r and a write quorum group S_i^w are assigned, where S_i^r and S_i^w are both in terms of a subset of the set of the sites storing replicated copies of

the object, so that the intersection of a read quorum group and a write quorum group is not empty, neither is the intersection of two write quorum groups. These two *intersection invariants* can be formally defined as:

> for each pair of sites s_i and s_j, and any formed S_i^r, S_i^w, S_j^r and S_j^w, $S_i^w \cap S_j^w \neq \emptyset$, $S_i^r \cap S_j^w \neq \emptyset$, and $S_j^r \cap S_i^w \neq \emptyset$.

A read (write) operation should get permission from each site in S_i^r (S_i^w) for processing it.

If a 2-phase locking mechanism [1] is applied, a QC method will force, through the intersection invariants, the situation that a write and a read cannot take place simultaneously on different copies of the same object, and similarly, neither can two writes. Thus, mutual consistency can be maintained.

Recent trends in developing new QC methods include coupling high data availability with a low communication cost for processing transactions. The achievements of a low communication cost are either

1. through the minimization of the number of remote sites which a transaction operation process has to communicate with [4,8,9], or
2. through the minimization of the total communication cost for processing a given set of transactions [5,6].

In this paper, we restrict our interests to 1. Interested readers may refer to [5,6] for a detailed discussion about minimizing the total communication cost.

To maximize the utilization of a distributed system, and then to enhance the overall performance of transaction processes, a *fully distributed* QC approach is investigated in [4,8,9]. In a fully distributed QC approach, each site bears almost equal responsibility in performing a QC approach. The lower bound of a quorum group size in a fully distributed QC method has been shown as $\sqrt{n}$ [8], where n is the number of sites storing a given object. This lower bound has been achieved by the QC method in [8].

A rigorous analysis [9] shows that by the QC method in [8], the data availability for processing an individual operation gets smaller (falls asymptotically to 0), when the number of replicas of each data item is increased. This defeats the main objective of data replication, i.e., increasing data availability through replication. [9] provides a fully distributed QC method which guarantees that the data availability is increased (asymptotically to 1) as the number of replicas increases. But each operation process is forced to communicate with at least $\Omega(\sqrt{n}\log^{0.5} n)$ remote sites.

In this paper, we will present a new QC method. The proposed method is also fully distributed, and guarantees that the data availability goes asymptotically to 1. Further, we can show that using the proposed approach, each operation process may communicate to less than $\Omega(\sqrt{n}\log^{0.5} n)$ remote sites:

- If at the time when an operation is being processed, there are no (or a few) sites with failures, then the operation process will communicate with only $O(\sqrt{n}\ \log^{0.208} n)$ remote sites.

- If many sites with failures, an operation process will be forced to communicate with $O(\sqrt{n}\ \log^{0.792} n)$ remote sites.

The rest of the paper is organized as follows. In Section 2, we first give our environment assumptions, and then introduce a mathematical model to justify the degree of fault-tolerance of a QC method. In Section 3, we present our QC method. Section 4 provides a sketch of a rigorous performance analysis of our QC approach, and a comparison between our QC method and the related approaches. This is followed by a conclusion.

2 Preliminaries

In our distributed system environment, we assume that communication between different sites is through exchanging messages. Detection of a failure of a site by another site happens through sending a message, but receiving no reply. To simplify the analysis, we assume that communication links never fail, and that each site failure probability is independent. The networks are *fully connected*; that is, a message can be sent directly between any pair of sites. We follow the model where replicated data is represented by multiple copies and the transactions are either *simple* read or *simple* write (that is, each read or write manipulates only one object). So, in this paper we need only to consider a QC method with respect to one object. Without loss of generality, we assume *full data replication*; that is, a copy of each object exists at all sites.

Suppose that in a distributed system N, the probability of each site being alive is known. Given a site i and a QC method A, let $R_{r,A,i}$ ($R_{w,A,i}$) be the probability of assembling successfully a read (write) quorum group at i by the QC method A. The QC method A has a *high site resilience* [9] if

$$\min_{i \in N, q \in \{r,w\}} R_{q,A,i} \rightarrow 1 \text{ when } n \rightarrow \infty,$$

where n is the number of sites in the network N. Note that a QC method with a high site resilience can guarantee that in the asymptotical case, an alive site is always able to assemble a read (or write) quorum group to process the transactions issued at it; that is, the data availability is increased through increasing the number of replicas of data, unlike the one in [8].

Note that we are concerned only with the fault tolerance of transaction synchronization algorithms. So, as far as we are concerned, an operation process cannot successfully assemble a read (or write) quorum group *only* due to site failures. We do not consider the situation where an operation process fails because some other operation has already been granted permission to proceed.

3 A New QC method

Inspired by the QC method in [9], our QC method is built on the top of Kumar's approach [4] and Maekawa's approach [8]. We first briefly review the methods in [4,8], and then present our QC algorithm.

3.1 Kumar's Method

Suppose that in a distributed system, there are n sites. [4] proposed a hierarchical quorum consensus (HQC) method for update synchronization. Here, we introduce our special implementation of the HQC method, which will be adopted in our new QC method.

A logical organization T_n of n sites in a network is a *Kumar* tree if:

- T_n is a rooted 3-way tree, that is, each node has at most three children; and the leaf set of T_n consists of the n sites.
- The levels of the tree are ordered such that the root is located at level 0, and the level number of a parent is one level lower than that of its children. (L_i denotes the set of nodes at level i.)
- T_n has $m = \lceil \log_3 n \rceil + 1$ levels (i.e., the last level is L_{m-1}.)
- Each node in L_i, for $0 \leq i \leq m-3$, has exactly three children; and each node in L_{m-2} either is a leaf or has at least two children.

Figure 1 illustrates the Kumar trees for $n = 6, 9$.

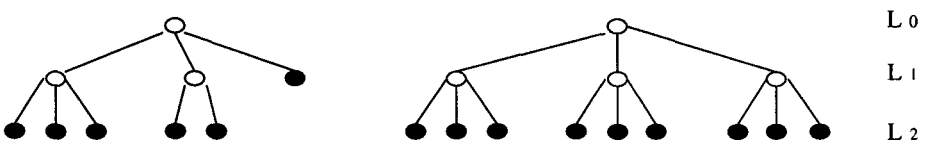

Fig. 1. Kumar trees

Our implementation of HQC performs as follows with respect to each site j:

Step 1: The n sites are logically organized as a Kumar tree. Each site keeps the Kumar tree as a map for assembling its read/write quorum groups. Go to Step 2.

Step 2: The regulations for assembling a read quorum and a write quorum group are the same:

> The root asks (randomly) a pair of its children to take part in assembling the quorum group, and marks these two children. Recursively, each marked node at L_i (randomly) marks a pair of its children at L_{i+1}, and asks them to participate in making the quorum group. (If the number of children of a marked node is smaller than 3, then the marked node should mark all its children.)

Go to Step 3.

Step 3: A quorum group is formed by all the marked leaves. In case there are some marked leaves with a failure, we need to perform a quorum group again. In order to save communication cost among remote sites, we should keep as

many marked non-failure leaves as possible in a new forming quorum group. So, instead of performing a new quorum group by Step 2 from the scratch, we can modify the failed formed quorum group to a new quorum group:

> Suppose that a marked site l has a failure. The site j looks up the Kumar tree to find the parent $pa1$ of site l. Then re-do the Step 2 with respect only to $pa1$, where keeping another originally marked child of $pa1$ being marked in the new forming quorum group. After this, if we find that it is impossible to assemble the quorum group with respect to $pa1$, then j looks up the Kumar tree again to find the parent $pa2$ of $pa1$. Then re-do the Step 2 with respect to $pa2$, where keeping another originally marked child of $pa2$ being marked in the new forming quorum group, and so on. Finally, we can determine whether or not it is possible to assemble a quorum group with alive sites.

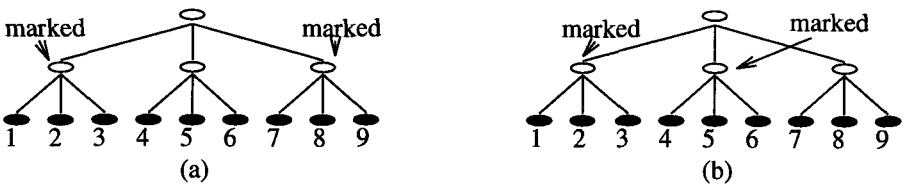

Fig. 2. HQC method

For instance, consider Figurer 2. It shows a collection of 9 sites organized into a three level Kumar tree. The various possible quorum groups are: $\{1, 2, 8, 9\}$ (see Figure 2(a)), $\{2, 3, 4, 6\}$ (see Figure 2(b)), etc. Suppose that $\{1, 2, 8, 9\}$ is initially formed, where the site 9 is not available. So by HQC, $\{1, 2, 7, 8\}$ is then formed. Again, we find site 7 is not available. By HQC, then one of $\{1, 2, 4, 5\}$, $\{1, 2, 4, 6\}$, and $\{1, 2, 5, 6\}$ will be formed.

It has been shown that the size of a quorum group created by using HQC is bounded by (not greater than) $2^{\lceil \log_3 n \rceil}$ ($\approx O(n^{0.63})$). This implies that a transaction process needs to communicate with $O(n^{0.63})$ remote sites to get a read (write) quorum group, if there are no sites with a failure.

Suppose that at the time when a transaction is issued, some sites are not available in the network due to their failures. As mentioned earlier, in a distributed environment a site failure is usually detected through sending a message. A transaction process, by HQC, may be forced to communicate with more than half of the sites in the network, through sending messages, to know whether or not it can successfully get a read (write) quorum group. Thus, in the worst case a read (write) quorum group is formed by communicating with $O(n)$ remote sites.

3.2 Maekawa's Method

[8] proposed another fully distributed quorum consensus method. A simple version of the method in [8] is to organize n sites into an approximate grid square, where the possible empty sites are placed at the grid positions as up and right as possible (see Figure 3). A (read or write) quorum group S_i of a site s_i $(1 \leq i \leq n)$ consists of the sites which are, with respect to the grid, in the row and the column occupied by s_i. (Figure 3 illustrates the quorum groups of site 4 and site 1.) Clearly, each S_i has at most $2\lceil\sqrt{n}\,\rceil - 1$ sites, any two quorum groups have non-empty intersection, and each site belongs to $O(\sqrt{n})$ groups. If a failure of a site in S_i happens, then the process of a transaction from site s_i has to wait until the site recovers.

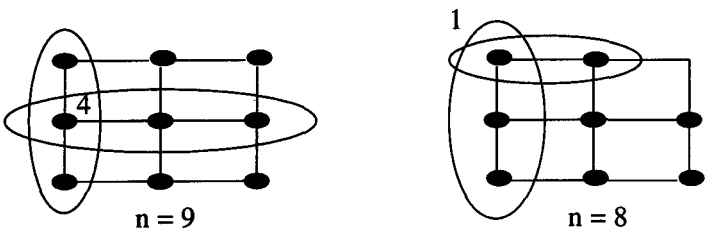

Fig. 3. Maekawa's method

3.3 Our QC Method: KMQC

In this sub-section, we present a new quorum consensus method - KMQC method, which is built on the top of Kumar and Maekawa's methods. In the KMQC method, the regulations for forming read and write quorum groups are the same. We first group n sites $\{s_i : 1 \leq i \leq n\}$ into k disjoint groups $\{G_i : 1 \leq i \leq k\}$ such that we try to make the sizes of these k groups as equal as possible. A quorum group is formed, in KMQC, through two layer constructions. KMQC first performs Maekawa's method on these k groups to obtain a "quorum group" A - a subset of $\{G_i : 1 \leq i \leq k\}$. By the application of HQC to each element G_i in A, we get a quorum group Q_i corresponding to each G_i in A; the union of all these Q_i forms a quorum group in KMQC. It can be precisely described as follows.

Step 1: Group the n sites into k disjoint subgroups $\{G_i : 1 \leq i \leq k\}$ such that

$$\lfloor \frac{n}{k} \rfloor \leq |G_i| \leq \lceil \frac{n}{k} \rceil, \tag{1}$$

and $\cup_{i=1}^{k} G_i = \{s_i : 1 \leq i \leq n\}$. A mapping g_1, from $\{s_i : 1 \leq i \leq n\}$ to $\{G_i : 1 \leq i \leq k\}$, is constructed, such that $g_1(s_i) = G_j$ if $s_i \in G_j$. Go to Step 2.

Step 2: By viewing each element G_i in $\{G_i : 1 \leq i \leq k\}$ as a "site", and by applying Maekawa's method [8] to these k "sites", we obtain the k "quorum groups" A_1, A_2, ..., A_k (i.e, each A_i is a subset of $\{G_i : 1 \leq i \leq k\}$) corresponding to theses k "sites" G_i for $1 \leq i \leq k$. Note $G_i \in A_i$. A mapping g_2 is constructed to identify the "quorum group" of "site" G_i: $g_2(G_i) = A_i$. (Note that $1 \leq i \leq k$, $|A_i| \leq 2\lceil\sqrt{k}\,\rceil - 1$.) Go to Step 3.

Step 3: For each site s_i, its (read or write) quorum group is formed, which consists of certain sites in group $g_2(g_1(s_i))$ (say A_j), such that for each element G_x in A_j, we use HQC to form a quorum (read or write) group Q_x^i in G_x. Then the quorum group of s_i is:

$$\cup_{G_x \in g_2(g_1(s_i))} Q_x^i.$$

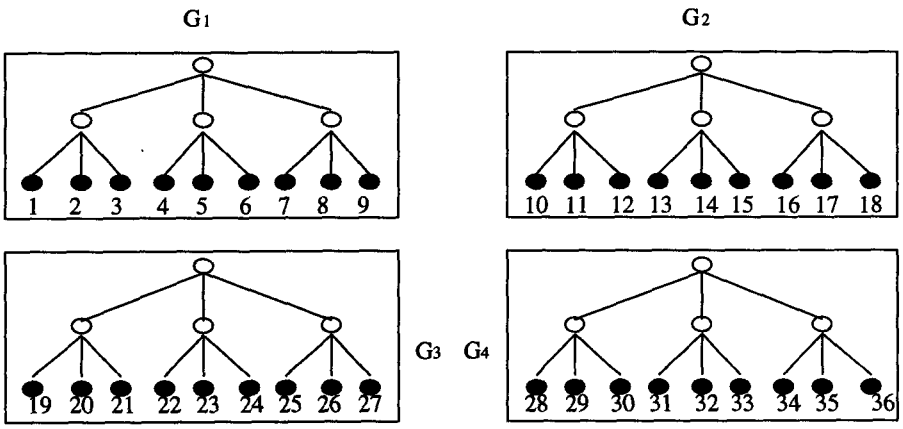

Fig. 4. KMQC approach

For example, consider Figure 4. There are 36 sites. Let $k = 4$, $G_1 = \{s_i : 1 \leq i \leq 9\}$, $G_2 = \{s_i : 10 \leq i \leq 18\}$, $G_3 = \{s_i : 19 \leq i \leq 27\}$, $G_4 = \{s_i : 28 \leq i \leq 36\}$. (Note that $g_1(s_{9i+k}) = G_{i+1}$ for $0 \leq i \leq 3$ and $1 \leq k \leq 9$.) By the application of Maekawa's algorithm, we have that $A_1 = \{G_1, G_2, G_3\}$, $A_2 = \{G_1, G_2, G_4\}$, $A_3 = \{G_1, G_3, G_4\}$, $A_4 = \{G_2, G_3, G_4\}$. (Note that $g_2(G_i) = A_i$ for $1 \leq i \leq 4$.) A transaction from site s_1 requires to form a read (write) quorum group consisting of certain sites in $g_2(g_1(s_1)) = A_1$, that is, certain sites in G_1, G_2, and G_3. Particularly, the quorum groups will be

$\{s_2, s_3, s_5, s_6, s_{13}, s_{14}, s_{16}, s_{18}, s_{19}, s_{20}, s_{22}, s_{23}\}$,

$\{s_4, s_5, s_7, s_8, s_{11}, s_{12}, s_{16}, s_{17}, s_{19}, s_{20}, s_{25}, s_{26}\}$, etc.

Our KMQC is a fully distributed method, since it is a combination of the method in [4] and the method in [8] which are fully distributed. It can be immediately verified that KMQC is also correct (i.e, each two quorum groups have at least one common site), based on the facts that both of the methods in [4,8] are correct.

Now, we estimate how many remote sites will be accessed by a transaction process, using KMQC. Clearly, if there are no sites with failures, a transaction

process needs to access at most $(2\lceil\sqrt{k}\,\rceil - 1)2^{\lceil(\log_3\lceil\frac{n}{k}\rceil)\rceil}$ remote sites.

$$O(\lceil\sqrt{k}\,\rceil 2^{\lceil(\log_3\lceil\frac{n}{k}\rceil)\rceil}) = O(\sqrt{k}\,2^{\log_3(\frac{n}{k})}) = O(\sqrt{k}\,(\frac{n}{k})^{\log_3 2}) \quad (2)$$

If many sites are not available in the network, then to perform a quorum group Q_x^i with respect to each relevant G_x, $O(|G_x|)$ remote sites may be communicated with. So in the worst case, a transaction process will communicate with the following number of remote sites:

$$O(\lceil\sqrt{k}\,\rceil\lceil\frac{n}{k}\rceil) = O(\sqrt{k}\,\frac{n}{k}). \quad (3)$$

4 A Performance Analysis of KMQC

Regarding to the performances of KMQC, the main result is:

Theorem 1. *In a distributed system N, suppose that the maximum value p of the site failure probabilities over all sites is less than 18.35%, and k is chosen as*

$$\min\{\lfloor\frac{n\log_2^{\frac{1}{\log_3 2}}\frac{1}{3p(2-p)}}{\log_2^{\frac{1}{\log_3 2}} n}\rfloor, n\}, \quad (4)$$

where n is the number of the sites in N. Then,

1. $\min_{i\in N, q\in\{r,w\}} R_{q,KMQC,i} \to 1$ *as* $n \to \infty$.
2. *The above (2) is* $O(\sqrt{n}\log^{1-\frac{1}{2\log_3 2}} n)$ $(\approx O(\sqrt{n}\log^{0.208} n))$, *while the above (3) is* $O(\sqrt{n}\log^{\frac{1}{2\log_3 2}} n)$ $(\approx O(\sqrt{n}\log^{0.792} n))$.

Theorem 1 says:

> In a class π of distributed systems where the maximum value of site failure probabilities through the whole class is p and $p \leq 0.1835$, the KMQC method has a *high site resilience* if with respect to any n sites distributed system in π, k is always chosen as (4). Given any n sites distributed system in π, each transaction process needs only to communicate with $O(\sqrt{n}\log^{0.208} n)$ remote sites in the best case, and needs to communicate with $O(\sqrt{n}\log^{0.792} n)$ remote sites in the worst case.

The proof of 2 in Theorem 1 is very straight forward according to the choice of k, (2), and (3). The proof of 1 requires a detailed mathematic estimation. Due to the space limitation, we do not include the detailed proof in this paper. The interested readers may refer to [7] for details. A sketch of the proof 1 is as follows:

> We first prove that HQC in [4] has a high site resilience through a mathematical estimation about the probability of failing to form any quorum groups at a site. Then we use the obtained mathematical estimation to prove 1. The main technique is to use some standard mathematic estimation approaches.

Next, we make a comparison between our KMQC method and the other three fully distributed algorithms in [4,8,9]. To simplify the description, we use HQC to denote the method in [4], GQC to denote the method in [8], HMV to denote the method in [9].

First, KMQC, HQC and HMV all have a high site resilience if the failure probability of each site is smaller than 18.35%, but GQC dose not have. So, there is only one loser, GQC, according to this index.

HQC requires to communicate with at least $\Omega(n^{0.63})$ remote sites, GQC requires $\Omega(\sqrt{n})$ remote sites, HMV requires at least $\Omega(\sqrt{n}\log^{0.5} n)$ remote sites. KMQC requires to communicate with at least $\Omega(\sqrt{n}\log^{0.208} n)$ remote sites. According to this index, GQC is the best, KMQC is the second best, HMV is the third.

Based on the above comparison, KMQC should be a winner, regarding to an integrated whole of remote sites number and fault-tolerance, in the reliable distributed systems where site failure probabilities are small.

5 Conclusion

In this paper, we present a fully distributed quorum consensus method, KMQC. KMQC method has a high site resilience like that in [9], while we can expect that in reliable distributed systems, KMQC should have a lower message overhead than that in [9].

Possible future studies could be on either further improving the message overhead or addressing the communication link failures.

Acknowledgement: The work of the first named author is partially supported by the grant IRG at UWA.

References

1. P. Bernstein, V. Hadzilocs and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Reading, Mass., 1987.
2. S. B. Davidson, H. Garcia-Molina and D. Skeen, Consistency in Partioned Networks, *ACM Computing Surveys*, 17(3), 341-370, 1985.
3. H. Garcia-Molina and D. Barbara, How to Assign Votes in a Distributed Systems, *J. ACM*, 32(4), 841-860, 1985.
4. A. Kumar, Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data, *IEEE Transactions on Computers*, 40(9), 996-1004, 1991.
5. A. Kumar and A. Segev, Cost and Availability Tradeoffs in Replicated Data Concurrency Control, *ACM Transactions on Database Systems*, 18(1), 102-131, 1993.
6. X. Lin and M. E. Orlowska, An Optimal Voting Schema for Minimizing the Overall Communication Cost in Replicated Data Management, to appear in *Journal of Parallel and Distributed Computing.*
7. X. Lin and M. E. Orlowska, On High Resilience and Low Message Overhead in Replicated Data Management, *Technical Report*, Computer Science Department, The University of Western Australia, Australia.

8. M. Maekawa, A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems, *ACM Transactions on Computer Systems*, 3(2), 145-159, 1985.
9. S. Rangarajan, S. Setia and S. K. Tripathi, A Fault-tolerant Algorithm for Replicated Data Management, *IEEE Proceedings of the 8th International Conference on Data Engineering*, 230-237, 1992.