An Efficient Optimal Algorithm for Minimizing the Overall Communication Cost in Replicated Data Management

Xuemin Lin and Maria E. Orlowska

Department of Computer Science, University of Queensland, QLD 4072, Australia
email: {lxue, maria}@cs.uq.oz.au

Abstract. Quorum consensus methods have been widely applied to managing replicated data. In this paper, we study the problem of vote and quorum assignments for minimizing the overall communication cost of processing the typical demands of transactions. This problem is left open, even restricted to a uniform network, and we shall show that it can be solved by an efficient polynomial time algorithm.

1 Introduction

The problem of managing replicated copies of data in a distributed database has received a great deal of attention [2, 3, 4, 5, 6, 9] throughout the last decade. The main issue is focussed on providing high availability of data through replicating data. Meanwhile, the replicated copies of data must be kept mutually consistent by synchronizing transactions at different sites so that a global serialization order can be ensured. A number of standard methods for replicated data concurrency control are based on the formation of quorums [1, 2]. They are called *quorum consensus* methods.

In this paper, we follow the model where replicated data is represented by multiple copies and the transactions are simple read/write. A quorum consensus (QC) method has several common features as follows.

- A vote v_i (integer) is assigned to each site s_i.
- Two threshold values (integers) are assigned: one is referred to as read threshold Q_r (called *read quorum size*), and the other is referred to as write threshold Q_w (called *write quorum size*) .
- Each read (write) transaction must assemble a read (write) quorum of sites such that the votes of all of the sites in the quorum add up to a value not less than Q_r (Q_w).
- *Two quorum intersection invariants* are assigned: $Q_r + Q_w > \sum_{i=1}^{n} v_i$, and $Q_w > \frac{\sum_{i=1}^{n} v_i}{2}$, where n is the number of sites.

If a 2-phase locking mechanism [1] is applied, these two quorum intersection invariants ensure that a read and a write cannot take place simultaneously on different copies of the same data, and neither can two writes.

The QC method described above is basic. Several variations [1, 3, 7, 8, 9] of the basic QC method have been proposed, aimed at extending it to enhance the performance of a QC method. The QC method described above is also static, that is, Q_r and Q_w are fixed "a priori". A number of dynamic QC methods, in which the votes and the quorum sizes Q_w and Q_r are adjustable as sites fail and recover, may be found in [4, 5]. We refer to the above QC method as a BSQC method.

Recently, minimizing communication costs in replicated data management, through a QC method, has been taken into account [6]. Kumar and Segev, taking a BSQC method as an example, showed a tradeoff between the overall communication cost, for processing the typical demands of transactions, and the availability of data. Several optimization problems, along with various optimal algorithms, have been proposed in their paper [6]. But, without forcing the output to meet some specified requirements, the problem of minimizing the overall communication cost for processing the typical demands of transactions, by a BSQC method, is left open in [6]. This applies even if the problem is restricted to a case where networks under consideration are "uniform" networks (see Section 2 for the definition). Only heuristics may be found, in [6], for the optimization problem restricted to a uniform network. We denote this problem by MCCU, which stands for "Minimizing Communication Cost through Uniform networks".

We shall show, in this paper, that MCCU can be solved in $O(n^2 \log n)$ with respect to an improved transaction management model in comparison to that in [6]. Here n is the number of sites in a network. Meanwhile, we show that MCCU, restricted to the same transaction management mode in [6], can be solved in $O(n)$ (as mentioned above, this problem is left open in [6]).

The rest of the paper is organized as follows. In Section 2, we present a formalization of the problem MCCU, along with the transaction management models. Section 3 gives solutions to MCUU. This is followed by remarks and conclusions.

2 A Formalization of MCCU

In this paper, we assume that the networks under consideration consist of n distributed processes (sites) which are fully connected. Each pair of processes can communicate only by messages, and do not share memory. We restrict our research, in this paper, to *uniform* networks where the *communication cost* between each pair of sites is the same c. By communication cost, we mean either the dollar cost of a unit data shipping or the time of a unit data shipping.

Without loss of generality, we assume full replication in our environment; that is, a copy of each replicated file exists at all n sites.

Suppose that $V = (v_1, v_2, ..., v_n)$ is a vote assignment such that v_i is the vote of site s_i. With respect to V, an assignment of write quorum size Q_w and read quorum size Q_r is *valid* if:

(2.1) $\sum_{i=1}^{n} v_i > Q_w$, $\sum_{i=1}^{n} v_i > Q_r$, $Q_r + Q_w > \sum_{i=1}^{n} v_i$, and $Q_w > \frac{\sum_{i=1}^{n} v_i}{2}$.

In the rest of the paper, we restrict our interests only to a valid assignment of Q_w and Q_r. An assignment of Q_w and Q_r, whenever mentioned, is always a valid assignment.

A site s_j is a *key* site with respect to $(v_1, v_2, ..., v_n)$, Q_w, and Q_r, if $\sum_{i=1, i \neq j}^{n} v_i < Q_w$. This means that every write must get a vote from each key site.

In a BSQC method, a read quorum S_j^r for each site s_j, which is a subset of the site set, has been assigned such that $\sum_{s_i \in S_j^r} v_i \geq Q_r$. Similarly, there is a write quorum S_j^w for each site s_j such that $\sum_{s_i \in S_j^w} v_i \geq Q_w$.

We use a similar transaction management model, as in [6], to perform a BSQC method for processing a transaction in a distributed environment. We assume that there is a *transaction manager* (TM) at each site. A write w (read r) is processed as follows:

- The transaction manager (TM) at the issuing site s_j of w (r) acts as the coordinator.
- The coordinator site first obtains locks on the desired set of records in its local file. Then the coordinator sends messages to the remote TMs which are in the write (read) quorum, requesting them to either send their versions of the corresponding records (if the coordinator is not a key site), or to send only their replies to conform that they have locked the corresponding records (if the coordinator is a key site). Each remote TM upon receiving a message must lock its own copy of the relevant records, and either
  1. read them and send them to the coordinator if the coordinator is not a key site, or
  2. send only a confirmation about the implementation of a lock to the coordinator if the coordinator is a key site.

  After receiving the reply messages from all the sites in the write (read) quorum S_j^w (S_j^r), the coordinator will update the the relevant records if necessary, and will run the transaction. Upon completion of the transaction, the coordinator will commit the transaction locally, release locks on the local copies, and send messages to the TMs at all other sites in the write quorum (read quorum) so that they can commit the transaction and release locks on their respective copies. For write transactions, the new image of the records is also sent along with the commit message.

The traffic volume for a write w (or a read r) is $X_{1w} + X_{2w} + X_{3w}$ (or $X_{1r} + X_{2r} + X_{3r}$) if the coordinate site is a key site, otherwise it is $X_{1w} + X_{2'w} + X_{3w}$ (or $X_{1r} + X_{2'r} + X_{3r}$). Here

- $X_{1r}(X_{1w})$ is the size of the request message from the coordinator to a remote site;
- X_{2r} (X_{2w}) is the size of the reply message from the remote site to the coordinator if the coordinator site is a key site;
- $X_{2'r}$ ($X_{2'w}$) is the size of the reply message from the remote site to the coordinator site if the coordinator site is not a key site;

- X_{3r} is the size of the release lock and commit message from the coordinator to the remote site (for read operation); and
- X_{3w} is the size of the update record, release lock, and commit message from the coordinator to the remote site.

Note that for the same transaction r (w),
(2.2) $X_{2'r}$ is usually larger than X_{2r}, and $X_{2'w}$ is usually larger than X_{2w}.

For the same transaction, each size of the reply message from a remote site may be different with respect to different sites if the coordinator is not a key site. We use the same approximate treatment here as in [6] by viewing them as the same $X_{2'r}$ ($X_{2'w}$). In [6], the authors assume that a transaction from a key site is processed in the same way as those from a non-key site, that is, $X_{2r} = X_{2'r}$ and $X_{2w} = X_{2'w}$. We drop this restriction in the paper, since a key site keeps all the update information, and we don't need any remote site to send its version of a relevant record of a file for processing a transaction from a key site.

For each site s_j, let: 1) r_j denote the summation of all $X_{1r} + X_{2r} + X_{3r}$ for all reads from s_j, representing the total data volume of read traffic from s_j in case that s_j acts as a key site, and 2) r'_j denote the summation of all $X_{1r} + X_{2'r} + X_{3r}$ for all reads from s_j, representing the total data volume of read traffic from s_j in case that s_j dose not act as a key site, and 3) w_j denote the total data volume of write traffic from s_j in case that s_j is assigned as a key site, and 4) w'_j denote the total data volume of write traffic from s_j in case that s_j is not assigned as a key site.

Note that $r'_i \geq r_i$ and $w'_i \geq w_i$, since (2.2) holds.

A BSQC method corresponds to a vote assignment $V = (v_1, ..., v_n)$, an assignment of Q_w and Q_r (with respect to V), and an assignment of $\{S_i^r : 1 \leq i \leq n\}$ and $\{S_i^w : 1 \leq i \leq n\}$. To minimize the overall communication cost in a uniform network, the following restrictions can be added:

- $s_i \in S_i^r, S_i^w$ for each i, and
- for each i and each $s_j \in S_i^r$, $v_j \geq v_k$ if $j \neq i$ and $s_k \notin S_i^r$, and
- for each i and each $s_j \in S_i^w$, $v_j \geq v_k$ if $j \neq i$ and $s_k \notin S_i^w$.

Therefore, in this paper, we study only the assignments of all S_i^r and S_i^w with these restrictions. The problem of MCCU can be expressed precisely as follows:
INSTANCE: given $\{r_i, r'_i, w_i, w'_i : 1 \leq i \leq n\}$ such that for each i, $r'_i \geq r_i$ and $w'_i \geq w_i$.
QUESTION: find a vote assignment $V = (v_1, ..., v_n)$, an assignment of a write quorum size Q_w and a read quorum size Q_r, and an assignment of $\{S_i^r : 1 \leq i \leq n\}$ and $\{S_i^w : 1 \leq i \leq n\}$ such that the following value is minimized:

$$\sum_{i=1}^{n}(|S_i^r| - 1)c(\delta(v_i, Q_w)r_i + (1 - \delta(v_i, Q_w))r'_i) +$$
$$\sum_{i=1}^{n}(|S_i^w| - 1)c(\delta(v_i, Q_w)w_i + (1 - \delta(v_i, Q_w))w'_i). \quad (1)$$

Here for each i, $\delta(v_i, Q_w) = 1$ if s_i is a key site with respect to V and Q_w, otherwise $\delta(v_i, Q_w) = 0$.

Note that in this paper, we study a more general optimization problem than the optimization problem in [6]. In [6], they assume that for each i, $r_i = r'_i$ and $w_i = w'_i$, and for each i, $\sum_{j \in S_i^r} v_j = Q_r$ and $\sum_{j \in S_i^w} v_j = Q_w$. We may expect that the overall communication cost with respect to a solution of MCCU is never greater than that with respect to a solution of the restricted MCCU in [6], since the problem domain of MCCU is larger than that of the restricted MCCU.

3 An Efficient Solution to MCCU

Obviously, a trivial exhaustive search for solving MCCU will be exponential time bounded. In this section, by characterizing the properties of a solution of MCCU we provide an algorithm OPT, with time bound $O(n^2 \log n)$, for solving the problem MCCU.

An assignment A of votes $V = (v_1, ..., v_n)$, quorum sizes Q_r and Q_w, and read and writ quorums $\{S_i^r, S_i^w : 1 \leq i \leq n\}$ is a ***key site based assignment*** if there is an positive integer l such that

- $v_{j_i} = n - l + 1$ for $1 \leq i \leq l$, and $v_{j_i} = 1$ for $l + 1 \geq i \geq n$, and
- $Q_r = n - l + 1$ and $Q_w = l(n - l + 1)$, and
- for $1 \leq i \leq l$, $S_{j_i}^r = \{s_{j_i}\}$ and $S_{j_i}^w = KEY = \{s_{j_x} : 1 \leq x \leq l\}$, and
- for $l + 1 \geq i \geq n$, $S_{j_i}^r$ consists of s_{j_i} and a site in KEY, and $S_{j_i}^w = \{s_{j_i}\} \cup KEY$.

We call KEY the ***key site set*** of A. Thus, the communication cost as described in (1), with respect to A, can be re-written as (called the *cost* of A):

$$c \sum_{i=1}^{n} (r'_i + |KEY| w'_i) - c \sum_{s_i \in KEY} (r'_i + w'_i + (|KEY| - 1)(w'_i - w_i)). \qquad (2)$$

The algorithm OPT will choose an appropriate key site based assignment as a solution to MCCU. The algorithm OPT consists of the following two steps:

Step 1: For $1 \leq k \leq n$, let KEY_k consist of k sites s_i whose $r'_i + w'_i + (k - 1)(w'_i - w_i)$ are the first k largest values among sites s_i $(1 \leq i \leq n)$, that is, $r'_i + w'_i + (k-1)(w'_i - w_i) \geq r'_j + w'_j + (k-1)(w'_j - w_j)$ if $s_i \in KEY_k$ and $s_j \notin KEY_k$. Then construct a key site based assignment A_k, whose key site set is KEY_k, of votes, quorums and quorum sizes. Go to Step 2.

Step 2: For $1 \leq k \leq n$, find a A_k such that its cost is minimized within $\{A_i : 1 \leq i \leq n\}$. Output A_k.

In Algorithm OPT, the most expensive procedure is to find the first k largest values of $r'_i + w'_i + (k-1)(w'_i - w_i)$, for each k, among all sites s_i. Here, we apply a simple implementation of this procedure. To find the first k largest values for each k, every time we first carry out sorting. Thus, Step 1 takes $O(n^2 \log n)$.

Meanwhile, Step 2 takes only $O(n)$. It follows that Algorithm OPT runs in $O(n^2 \log n)$.

We are now proving that the algorithm OPT gives a solution to the problem MCCU. First, we show the following fact.

Lemma 1. *Suppose that s_i is not a key site with respect to an assignment of votes $V = (v_1, ..., v_n)$ and a write quorum size Q_w. Then, for any assignment of a read quorum size Q_r, $v_i < Q_r$. (In other words, a read from a non-key site must be processed to access at least one remote site.)*

Proof. Suppose that $v_i \geq Q_r$. From $Q_w + Q_r > \sum_{j=1}^{n} v_j$, it follows that $Q_w > \sum_{j=1, j \neq i}^{n} v_i$. Thus, s_i is a key site. Contradiction! □

Lemma 2. *Suppose $V = (v_1, ..., v_n)$ is an assignment of votes, Q_w and Q_r are respectively a write quorum size assignment and a read quorum size assignment, and π is the set of key sites. Further for each site s_i, S_i^w is a write quorum assignment. Then $\pi \subseteq S_i^w$.*

Proof. This Lemma follows immediately from the definitions of a key site and a write quorum. □

From Lemma 2 and Lemma 1, we can prove the following fact.

Lemma 3. *Suppose that A1 is an assignment of votes $V = (v_1, ..., v_n)$, quorum sizes of Q_w and Q_r, and quorums $\{S_i^r, S_i^w : 1 \leq i \leq n\}$. π is the set of key sites with respect to V and Q_w. Further suppose that A is a key site based assignment A with π as its key site set. Then, the cost of A is smaller than or equal to the cost of A1.*

Proof. From Lemma 2 and Lemma 1, it follows that the cost of any assignment, with π as the set of key sites, of votes $V = (v_1, ..., v_n)$, quorum sizes Q_w and Q_r, and quorums $\{S_i^r, S_i^w : 1 \leq i \leq n\}$ is smaller than or equal to

$$c \sum_{s_i \in \pi} (|\pi| - 1) w_i + c \sum_{s_i \notin \pi} (r_i' + |\pi| w_i').$$

This deduces the Lemma. □

Lemma 4. *Suppose that with respect to an assignment of votes V, quorums, and quorum sizes Q_w and Q_r, there are no key sites. Then each $v_i < Q_w$ for $1 \leq i \leq n$. (This means that each update from any site must be implemented to access at least one remote site.)*

Proof. We are proving this Lemma through the approach of a reduction to absurdity. Suppose that there is a site s_i such that $v_i \geq Q_w$. From $Q_w > \frac{\sum_{i=1}^{n} v_i}{2}$ and $v_i \geq Q_w$, it follows that $Q_w > \frac{Q_w}{2} + \frac{\sum_{j=1, j \neq i}^{n} v_j}{2}$. This implies that $Q_w > \sum_{j=1, j \neq i}^{n} v_j$.

Hence, s_i is a key site. This contradicts the assumption that there are no key sites. □

From Lemma 4 and Corollary 1, we have the following Corollary.

Corollary 5. *In any assignment A of votes, quorums and quorum sizes, if there are no keys, then the cost of any key site based assignment A_1, whose key site set consists only one site, is smaller than or equal to the cost of A.*

Proof. From Lemma 4 and Corollary 1, we have that the cost of A is not smaller than $c\sum_{i=1}^{n}(r_i'+w_i')$. Meanwhile, the cost of a key site based assignment, whose key site set consists of only site s_j, is $cw_j + c\sum_{i=1,i\neq j}^{n}(r_i'+w_i')$.

Note that each $w_j \leq w_j'$. The Corollary follows immediately. □

From Corollary 5 and Lemma 3, it follows that we need only to choose an appropriate key site based assignment as a solution to MCCU. Next, we show what kinds of sites we should choose for a key site based assignment, whose key site set cardinality is k.

Lemma 6. *Among key site based assignments, with k key sites, of votes, quorums and quorum sizes, a key site based assignment A_k, such that the key site set consists of those k key sites s_i whose $r_i'+w_i'+(k-1)(w_i'-w_i)$ are the first k largest values, has the minimal cost.*

Proof. To prove this Lemma, we need only prove the following fact.

Let A^1 and A^2 are two key site based assignments with k sites. Suppose that KEY^1 and KEY^2 are the corresponding key site sets of A^1 and A^2 such that KEY^1 consists of π (a set of $k-1$ sites) and s_i, KEY^2 consists of π and s_j, $r_i'+w_i'+(k-1)(w_i'-w_i) \geq r_j'+w_j'+(k-1)(w_j'-w_j)$. Then, the cost of A^1 is not greater than that of A^2.

By using (2), we may immediately verify this fact. □

From Corollary 5, Lemma 3, and Lemma 6, it follows:

Theorem 7. *Algorithm OPT gives a solution to the problem MCCU.*

A Remark to MCCU

If we apply the same transaction management model as that in [6], we may speed up our algorithm OPT for solving the problem MCCU. In that transaction management model, it is assumed that a transaction from a key site is processed in the same way as those from a non-key site. That is, at each site s_i, $X_{2r} = X_{2r}'$ for a read r and $X_{2w} = X_{2w}'$ for a write w. This implies that at each s_i, $r_i = r_i'$ and $w_i = w_i'$. We use SMCCU to denote the problem MCCU, restricted the transaction management model in [6]. Here, SMCCU stands for "Simple MCCU".

All the Lemmas and Corollaries, proven earlier, still hold for solving SMCCU. Further, we are able to characterize explicitly how many key sites we need and what kind of site can be a key site.

Lemma 8. *Suppose that A is an arbitrary key site based assignment of votes, quorums and quorum sizes, and KEY is key site set of A. Then, a key site based assignment A_1, with one of the following two properties, will never lead to a larger communication cost to that of A:*

1. *either the key site set* KEY_1 *of* A_1 *is* $KEY \cup \{s_{i_0}\}$ *where* $s_{i_0} \notin KEY$ *and* $r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j \geq 0$, *or*
2. *the key site set* KEY_1 *of* A_1 *is* $KEY - \{s_{i_0}\}$ *where* KEY *has at least two elements,* $s_{i_0} \in KEY$, *and* $r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j < 0$.

Proof. Noting the formula (2), we have that the communication cost with respect to the assignment A is:

$$c\sum_{i=1}^{n}(r_i + |KEY|w_i) - c\sum_{s_i \in KEY}(r_i + w_i), \tag{3}$$

and the communication cost with respect to the assignment A_1 is:

$$c\sum_{i=1}^{n}(r_i + |KEY_1|w_i) - c\sum_{s_i \in KEY_1}(r_i + w_i). \tag{4}$$

In case that the assignment A_1 has the property 1, the formula (4) can be re-written as:

$$c\sum_{i=1}^{n}(r_i + |KEY|w_i) - c\sum_{s_i \in KEY}(r_i + w_i)(r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j). \tag{5}$$

It follows that the Lemma holds for a key site based assignment A_1 with the property 1.

In case that the assignment A_1 has the property 2, the formula (4) can be re-written as:

$$c\sum_{i=1}^{n}(r_i + |KEY|w_i) - c\sum_{s_i \in KEY}(r_i + w_i)(r_{i_0} + w_{i_0} - \sum_{j=1}^{n} w_j). \tag{6}$$

It follows that the Lemma holds for a key site based assignment A_1 with the property 2. □

Thus, we obtain a more efficient algorithm OPTS, than the algorithm OPT, to solve SMCCU. The algorithm OPTS proceeds as follows, to find an appropriate key site based assignment:

(A) If there are some sites such that $r_i + w_i - \sum_{j=1}^{n} w_j \geq 0$, the algorithm will choose those sites s_i, with $r_i + w_i - \sum_{j=1}^{n} w_j \geq 0$, to form a site set KEY, and then output the key site based assignment with KEY as its key site set. Otherwise go to (B).

(B) The algorithm will choose the site s_i such that $r_i + w_i - \sum_{j=1}^{n} w_j$ is maximized. And then it will output the key site based assignment with $\{s_i\}$ as its key site set.

It is clear that we can scan all sites only once to implement the algorithm OPTS. This means that the algorithm OPTS takes $O(n)$.

4 Conclusion

In this paper, we investigate the quorum consensus methods for managing replicated data in distributed database systems. The network environment considered in this paper is a uniform network with n sites. We present an algorithm, $O(n^2 \log n)$, to produce an optimal solution for minimizing the overall communication cost of processing the typical demands of transactions by quorum consensus methods. This takes the form of an improved transaction management model in comparison with that in [6]. Meanwhile, we also show that the optimization problem, restricted to the same transaction management model in [6], can be solved in $O(n)$. A possible future study may be carried out through a general network.

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. M. Herlihy, Dynamic Quorum Adjustment for Partitioned Data, *ACM Transactions on Database Systems*, 12(2), 170-194, 1987.
5. S. Jajodia and D. Mutchler, Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database, *ACM Transactions on Database Systems*, 15(2), 230-280, 1990.
6. A. Kumar and A. Segev, Cost and Availability Tradeoffs in Replicated Data Concurrency Control, *ACM Transactions on Database Systems*, 18(1), 102-131, 1993.
7. X. Lin and M. E. Orlowska, A Fault-Tolerant Hybrid Hierarchical Quorum Consensus Method for Managing Replicated Data, *Manuscript*, 1994.
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.