ELSEVIER

The Journal of Systems and Software 53 (2000) 73–82

The Journal of Systems and Software

www.elsevier.com/locate/jss

Optimal binary vote assignment for replicated data

Her-Kun Chang [a,*], Shyan-Ming Yuan [b]

[a] Department of Information Management, Chang Gung University, 259 Wen-Hwa 1st Road, Kwei-Shan, Tao-Yuan 333, Taiwan
[b] Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan

Received 16 April 1999; received in revised form 10 August 1999; accepted 16 December 1999

Abstract

Data replication can be used to improve the availability of data in a distributed database system. In such a system, a mechanism is required to maintain the consistency of the replicated data. Weighted voting is a popular solution for maintaining the consistency. The performance of a voting system is determined by the vote assignment. It was shown in previous studies that finding the optimal solution among nonnegative integer vote assignments requires $O(2^{N^2})$ time. So it is not suitable to find the optimal integer vote assignment (OIVA) for large systems. In this paper, we propose the optimal binary vote assignment (OBVA) considering only binary vote assignments. It is shown that OBVA can achieve nearly the same availability of OIVA. On the other hand, OBVA requires only $O(N^2)$ time, which is preferred for large systems.

Keywords: Distributed systems; Data replication; Availability; Weighted voting; Vote assignment

1. Introduction

In a distributed database system, replicated copies of the same data are kept in different sites in order to increase the probability that an arriving operation can be performed, i.e., to increase the *availability* of the data.

The issue of replica control occurs when multiple copies of replicated data are stored at different sites. The goal of replica control is to maintain the consistency among replicated copies of the same data, i.e., to guarantee that the multiple copies of the same data behave like a single copy. Discussions concerned with the consistency of replicated databases can be found in Bernstein and Goodman (1981, 1985) and Davidson et al. (1985).

Weighted voting (Gifford, 1979) is a popular solution for synchronizing read and write operations to replicated data. With weighted voting:

- Every site is assigned some number of votes.
- A read operation must collect a read quorum of at least r votes.
- A write operation must collect a write quorum of at least w votes.
- To ensure consistency, $r + w$ must be greater than the total number of votes assigned to all copies.

The values of r and w are called *read quorum* and *write quorum*, respectively. When the quorum for each operation is a majority of all votes assigned, it is called *majority voting* or *majority consensus* (Thomas, 1979).

Vote and quorum assignments can deeply influence the availability of replicated data. The problem of vote and/or quorum assignments has been discussed widely in the literature. The key research subjects are involved in determining the vote and quorum assignments to maximize the system performance (Garcia-Molina and Barbara, 1985; Barbara and Garcia-Molina, 1987; Ahamad and Ammar, 1989; Cheung et al., 1989; Tang and Natarajan, 1989, 1993; Tong and Kain, 1991; Spasojevic and Berman, 1994; Amir and Wool, 1998). For instance, Barbara and Garcia-Molina (1987) derived optimal vote assignments for several homogeneous systems with majority voting. Then Tong and Kain (1991) presented an $O(2^N)$ algorithm to compute an optimal real number vote assignment and an $O(N)$ approximation algorithm to find a near-optimal integer vote assignment with majority voting.

Recall that majority voting is a special case of weighted voting. With majority voting, the read and write operations are treated as the same. On the other hand, weighted voting has the flexibility to tune the read

* Corresponding author.
E-mail address: hkchang@mail.cgu.edu.tw (H.-K. Chang).

0164-1212/00/$ - see front matter
PII: S0164-1212(00)00056-X

and write quorums to achieve a better performance according to the mixed ratio of read and write operations.

Ahamad and Ammar (1989) considered the problem of optimal quorum assignment by choosing values for read quorum r and write quorum w to maximize the availability under uniform vote assignment (i.e., each site is assigned a single vote). Later, Cheung et al. (1989) studied the problem of finding the optimal solution from nonnegative integer vote assignments and quorums for read and write operations. The authors presented an algorithm to enumerate a sequence of sets of vote assignments from which the optimal one can be found by exhaustively searching and evaluating the enumerated vote assignments. Although their algorithm can reduce the number of enumeration by eliminating duplicated, dominated and isomorphic assignments, the number of enumeration is still lower bounded by $O(2^{N^2})$.

In this study, we propose the optimal binary vote assignment (OBVA) considering only binary vote assignments (i.e., the vote assigned to each site is either zero or one). An $O(N^2)$ algorithm is presented to find the optimal binary vote assignment for replicated copies. In the following, the optimal solution found by Cheung et al. (1989) is said to be the optimal integer vote assignment (OIVA). We compare OBVA with OIVA for the system availability, the computational complexity and the number of copies required. Experimental results show that OBVA can achieve nearly the same availability of OIVA. On the other hand, OBVA require only $O(N^2)$ time, which is preferred for large systems as compared to $O(2^{N^2})$ time required by OIVA. Moreover, a binary vote assignment can be viewed as an allocation of replicated copies such that a copy is allocated to a site if and only if the vote assigned to the site is one. For a given set of sites, OBVA usually uses less number of copies than that required by OIVA because there may be sites having vote 'zero'.

The remainder of this paper is organized as follows. The model and formulation are described in the next section and the algorithm for OBVA is provided in Section 3. Some experimental results are shown in Section 4. The final section summarizes this paper with some concluding remarks.

2. Model and formulation

A distributed database system consists of a set of sites, which may store multiple copies of the replicated data. In this study, the sites of the system are assumed to be fully connected with perfect links. A site is either *operational* or *failed* and the state (operational, or failed) of each site is statistically independent to the others. The *availability* of a site is the probability that the site is operational at any time instant. When a site is operational, the copy at the site is available; otherwise, it is unavailable.

Let N be the number of sites in the system and the sites in the system are labeled from 1 to N. Let p_i denote the availability of site i, we assume in the following that

$$0 \leqslant p_N \leqslant \cdots \leqslant p_1 \leqslant 1.$$

With weighted voting, each site in the system is assigned some number of votes. In this study, we consider binary vote assignment, i.e., the vote assigned to each site is either zero or one. A binary vote assignment Z is a function such that

$$Z(i) \in \{0,1\}, \quad 1 \leqslant i \leqslant N,$$

where $Z(i)$ is the vote assigned to site i. A binary vote assignment can be treated as an allocation of replicated copies and a vote assigned to a site results in a copy allocated at the site. That is,

1 vote $\equiv$ 1 copy.

Let

$$L_Z = \sum_{i=1}^{N} Z(i).$$

Then, L_Z is the total number of votes assigned to all sites in the system and L_Z is equal to the number of copies allocated in the system.

Let r and w denote the read quorum and write quorum, respectively. Using weighted voting for replica control, an arriving read operation can be performed if at least r copies are available. On the other hand, when a write operation arrives, it can be performed only if at least w copies are available.

To ensure that any pair of read and write operations are not able to be performed concurrently and that a read operation always gets the up-to-date value, $r + w$ must be greater than the total number of copies (votes) assigned to all sites. The following conditions are used to ensure consistency:

$$1 \leqslant r \leqslant L_Z, \quad 1 \leqslant w \leqslant L_Z, \tag{1}$$

$$r + w = L_Z + 1. \tag{2}$$

Conditions (1) and (2) ensures that there is a nonempty intersection of copies between every pair of read and write operations. Thus the conditions ensure that a read operation can access to a most recently updated copy (updated by the last write operation) of the replicated data. Timestamps can be used to determine which copies are most recently updated. [1] Under condition (2), a read

[1] In some case, it is not allowed to perform two write operations at the same time. An additional condition, $2w > L_Z$, is used to fill the requirement and version numbers may be used to find the most recently updated copies.

quorum r can determine a unique write quorum w such that

$$w = L_Z + 1 - r. \tag{3}$$

Let $S(Z)$ be the set of sites at which replicated copies are stored corresponding to the assignment Z. Then

$$S(Z) = \{j \mid Z(j) = 1,\ 1 \leqslant j \leqslant N\}.$$

For a quorum q, a *quorum group* is any subset of $S(Z)$ whose size is greater than or equal to q. The collection of all quorum groups is defined as the *quorum set*.[2] Formally, let $Q(Z,q)$ be the quorum set with respect to the assignment Z and quorum q, then

$$Q(Z,q) = \{G \mid G \subseteq S(Z) \text{ and } |G| \geqslant q\}.$$

For example, let $N = 4$, consider an assignment Z such that

$$Z(1) = Z(2) = 1, \quad Z(3) = 0, \quad Z(4) = 1.$$

Then

$$L_Z = Z(1) + Z(2) + Z(3) + Z(4) = 3$$

and

$$S(Z) = \{1,2,4\}.$$

If $r = 1$ and $w = L_z - r + 1 = 3$, the quorum sets for read and write operations are $Q(Z,1)$ and $Q(Z,3)$, respectively, where

$$Q(Z,1) = \{\{1\},\{2\},\{4\},\{1,2\},\{1,4\},\{2,4\},\{1,2,4\}\}$$

and

$$Q(Z,3) = \{\{1,2,4\}\}.$$

For any assignment Z and quorum q, define $\alpha(Z,q)$ to be the probability that at least q sites in $S(Z)$ are available, then

$$\alpha(Z,q) = \Pr\{\text{at least } q \text{ sites in } S(Z) \text{ are available}\} = \sum_{G \in Q(Z,q)} \left(\prod_{j \in G} p_j \prod_{j \in S(Z)-G} (1 - p_j) \right).$$

The probabilities of successful read and write operations are $\alpha(Z,r)$ and $\alpha(Z,w)$, respectively. Let $\mathrm{Av}(Z,r,w)$ denote the availability corresponding to the assignment Z, read quorum r and write quorum w. If the probability that an arriving operation is read and the probability that an arriving operation is write are f and $1-f$, respectively, then

$$\mathrm{Av}(Z,r,w) = f\alpha(Z,r) + (1-f)\alpha(Z,w).$$

The notations used in this analysis are summarized as follows:

- N: the number of sites in the system;
- p_i: availability of site i;
- $Z(i)$: the vote assigned to site i, $Z(i) \in \{0,1\}$, $1 \leqslant i \leqslant N$;
- L_Z: total number of votes assigned to the sites in the system, $L_Z = \sum_{i=1}^{N} Z(i)$;
- r: read quorum;
- w: write quorum, $w = L_Z - r + 1$;
- $S(Z)$: the set of sites with vote 1, $S(Z) = \{j \mid Z(j) = 1,\ 1 \leqslant j \leqslant N\}$;
- $Q(Z,q)$: quorum group for the quorum q, $Q(Z,q) = \{G \mid G \subseteq S(Z) \text{ and } |G| \geqslant q\}$;
- $\alpha(Z,q)$: the probability that at least q sites in $S(Z)$ are available,
$$\alpha(Z,q) = \sum_{G \in Q(Z,q)} \left(\prod_{j \in G} p_j \prod_{j \in S(Z)-G} (1 - p_j) \right);$$
- f: read probability;
- $\mathrm{Av}(Z,r,w)$: system availability (availability of data), $\mathrm{Av}(Z,r,w) = f\alpha(Z,r) + (1-f)\alpha(Z,w)$.

Giving N, f and $p_1, \ldots, p_N$ ($N \geqslant 1$, $0 \leqslant f \leqslant 1$, $0 \leqslant p_N \leqslant \cdots \leqslant p_1 \leqslant 1$), the problem of OBVA can be formulated as

$$\begin{array}{lll} P^0: & \text{Maximize} & \mathrm{Av}(Z,r,w) \\ & \text{Subject to} & Z(i) \in \{0,1\},\ 1 \leqslant i \leqslant N, \\ & & 1 \leqslant r \leqslant L_Z,\ 1 \leqslant w \leqslant L_Z, \\ & & r + w = L_Z + 1. \end{array}$$

3. Optimal binary vote assignment

The size of the solution space of P^0 is 2^N, since there are 2^N distinct assignments. In this section, we show that the number of assignments that need to be considered for optimization is N. That is, the solution space can be reduced from 2^N to N.

Lemma 1. *Let $\{\pi_1, \ldots, \pi_i\}$ be a subset of $\{1, \ldots, N\}$ such that $1 \leqslant \pi_1 < \cdots < \pi_i \leqslant N$. For $1 \leqslant k \leqslant i \leqslant N$, let Z_1, Z_2 be two assignments such that $S(Z_1) = \{1, \ldots, k-1, \pi_k, \ldots, \pi_i\}$ and $S(Z_2) = \{1, \ldots, k, \pi_{k+1}, \ldots, \pi_i\}$, then $\mathrm{Av}(Z_1,r,w) \leqslant \mathrm{Av}(Z_2,r,w)$, for any pair of read quorum r and write quorum w.*

Proof. Since $1 \leqslant \pi_1 < \cdots < \pi_i \leqslant N$, thus $\pi_k \geqslant k$ and $p_k \geqslant p_{\pi_k}$, for all $k = 1, \ldots, i$. Let Z_0 be the assignment

[2] The concept of quorum sets defined in this paper is basically the same as the concepts of coteries (Garcia-Molina and Barbara, 1985; Cheung et al., 1989) and acceptance sets (Tang and Natarajan, 1989).

such that $S(Z_0) = S(Z_1) - \{\pi_k\} = S(Z_2) - \{k\}$, then, for any quorum q

$$\begin{aligned}\alpha(Z_2,q) &= \Pr\{\text{at least } q \text{ sites in } S(Z_2) \text{ are available}\}\\ &= \Pr\{(\text{site } k \text{ and at least } q-1 \text{ sites in } S(Z_0) \text{ are available})\\ &\quad \text{or (site } k \text{ fails and at least } q \text{ sites in } S(Z_0) \text{ are available})\}\\ &= \Pr\{\text{site } k \text{ and at least } q-1 \text{ sites in } S(Z_0) \text{ are available}\}\\ &\quad + \Pr\{\text{site } k \text{ fails and at least } q \text{ sites in } S(Z_0) \text{ are available}\}\\ &= p_k\alpha(Z_0,q-1) + (1-p_k)\alpha(Z_0,q)\\ &= \alpha(Z_0,q) + p_k(\alpha(Z_0,q-1) - \alpha(Z_0,q)).\end{aligned}$$

Similarly,

$$\alpha(Z_1,q) = \alpha(Z_0,q) + p_{\pi_k}(\alpha(Z_0,q-1) - \alpha(Z_0,q))$$

and

$$\alpha(Z_2,q) - \alpha(Z_1,q) = (p_k - p_{\pi_k})(\alpha(Z_0,q-1) - \alpha(Z_0,q)) \geqslant 0.$$

That is $\alpha(Z_1,q) \leqslant \alpha(Z_2,q)$, for any quorum q. Therefore, for any pair of read quorum r and write quorum w, $\mathrm{Av}(Z_1,r,w) = f\alpha(Z_1,r) + (1-f)\alpha(Z_1,w) \leqslant f\alpha(Z_2,r) + (1-f)\,\alpha(Z_2,w) = \mathrm{Av}(Z_2,r,w)$. □

For $1 \leqslant i \leqslant N$, define Z^i to be the assignment such that $S(Z^i) = \{1,\ldots,i\}$.

Theorem 1. *Z^i is optimal in the set of assignments $A = \{Z \,||S(Z)| = i\}$, $1 \leqslant i \leqslant N$.*

Proof. Let Y be any assignment in A and $S(Y) = \{\pi_1, \pi_2, \ldots, \pi_i\}$ with $1 \leqslant \pi_1 < \pi_2 < \cdots < \pi_i \leqslant N$. Let Y^k be the assignment such that $S(Y^k) = \{1,\ldots,k, \pi_{k+1},\ldots,\pi_i\}$, $1 \leqslant k \leqslant i$. From Lemma 1, $\mathrm{Av}(Y,r,w) \leqslant \mathrm{Av}(Y^1,r,w) \leqslant \cdots \leqslant \mathrm{Av}(Y^i,r,w) = \mathrm{Av}(Z^i,r,w)$, for any pair of read quorum r and write quorum w. □

According to Theorem 1, the original problem can be reduced to

$$\begin{array}{lll} P': & \text{Maximize} & \mathrm{Av}(Z^i,r,w)\\ & \text{Subject to} & 1 \leqslant i \leqslant N,\\ & & 1 \leqslant r \leqslant L_Z,\ 1 \leqslant w \leqslant L_Z,\\ & & r + w = L_Z + 1.\end{array}$$

Theorem 2. *The optimal solution of P' is also optimal for P^0.*

Proof. Let Y^0 and Y' be optimal solutions of P^0 and P', respectively. Suppose $|S(Y^0)| = i$ and $\mathrm{Av}(Y^0,r,w) > \mathrm{Av}(Y',r,w)$, by Theorem 1, $\mathrm{Av}(Y^0,r,w) \leqslant \mathrm{Av}(Z^i,r,w) \leqslant \mathrm{Av}(Y',r,w)$. It's a contradiction. □

Table 1
$(p_1,p_2,p_3,p_4,p_5) = (0.9,0.8,0.8,0.8,0.8)$

f	Optimal integer vote assignment	r^1	α^1	Optimal binary vote assignment	r^2	α^2	α^2/α^1
0.001	(1,1,1,1,1)	5	0.9992	(1,1,1,1,1)	5	0.9992	1.0
0.1	(1,1,1,1,1)	4	0.9741	(1,1,1,1,1)	4	0.9741	1.0
0.2	(2,1,1,1,1)	4	0.9677	(1,1,1,1,0)	3	0.9574	0.9894
0.3	(2,1,1,1,1)	4	0.9613	(1,1,1,1,1)	3	0.9574	0.9960
0.4	(1,1,1,1,1)	3	0.9574	(1,1,1,1,1)	3	0.9574	1.0
0.5	(1,1,1,1,1)	3	0.9574	(1,1,1,1,1)	3	0.9574	1.0
0.6	(1,1,1,1,1)	3	0.9574	(1,1,1,1,1)	3	0.9574	1.0
0.7	(2,1,1,1,1)	3	0.9613	(1,1,1,1,1)	3	0.9574	0.9960
0.8	(2,1,1,1,1)	3	0.9677	(1,1,1,1,0)	2	0.9574	0.9894
0.9	(1,1,1,1,1)	2	0.9741	(1,1,1,1,1)	2	0.9471	1.0
0.999	(1,1,1,1,1)	1	0.9921	(1,1,1,1,1)	1	0.9992	1.0

Table 2
$(p_1,p_2,p_3,p_4,p_5) = (0.9,0.9,0.8,0.8,0.8)$

f	Optimal integer vote assignment	r^1	α^1	Optimal binary vote assignment	r^2	α^2	α^2/α^1
0.001	(1,1,1,1,1)	5	0.9993	(1,1,1,1,1)	5	0.9993	1.0
0.1	(2,2,1,1,1)	5	0.9839	(1,1,1,1,0)	3	0.9796	0.9956
0.2	(2,2,1,1,1)	5	0.9741	(1,1,1,1,0)	3	0.9699	0.9957
0.3	(3,3,2,2,2)	7	0.9729	(1,1,1,1,1)	3	0.9699	0.9969
0.4	(3,3,2,2,2)	7	0.9718	(1,1,1,1,1)	3	0.9699	0.9981
0.5	(2,2,1,1,1)	4	0.9713	(1,1,1,1,1)	3	0.9699	0.9986
0.6	(3,3,2,2,2)	6	0.9718	(1,1,1,1,1)	3	0.9699	0.9981
0.7	(3,3,2,2,2)	6	0.9729	(1,1,1,1,1)	3	0.9699	0.9969
0.8	(2,2,1,1,1)	3	0.9741	(1,1,1,1,0)	2	0.9699	0.9957
0.9	(2,2,1,1,1)	3	0.9839	(1,1,1,1,0)	2	0.9796	0.9956
0.999	(1,1,1,1,1)	1	0.9993	(1,1,1,1,1)	1	0.9993	1.0

Table 3
$(p_1, p_2, p_3, p_4, p_5, p_6, p_7) = (0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65)$

f	Optimal integer vote assignment	r^1	α^1	Optimal binary vote assignment	r^2	α^2	α^2/α^1
0.001	(3,3,2,2,1,1,1)	11	0.9995	(1,1,1,1,0,0,0)	4	0.9994	0.9999
0.1	(7,5,4,3,2,2,1)	15	0.9909	(1,1,1,1,0,0,0)	3	0.9872	0.9963
0.2	(13,10,8,6,5,4,3)	28	0.9872	(1,1,1,1,1,1,0)	4	0.9806	0.9933
0.3	(10,8,6,5,4,3,2)	21	0.9852	(1,1,1,1,1,0,0)	3	0.9768	0.9915
0.4	(8,6,5,4,3,2,2)	16	0.9840	(1,1,1,1,1,0,0)	3	0.9768	0.9926
0.5	(7,5,4,3,3,2,1)	13	0.9836	(1,1,1,1,1,0,0)	3	0.9768	0.9931
0.6	(8,6,5,4,3,2,2)	15	0.9840	(1,1,1,1,1,0,0)	3	0.9768	0.9926
0.7	(10,8,6,5,4,3,2)	18	0.9852	(1,1,1,1,1,0,0)	3	0.9768	0.9915
0.8	(13,10,8,6,5,4,3)	22	0.9872	(1,1,1,1,1,1,0)	3	0.9806	0.9933
0.9	(7,5,4,3,2,2,1)	10	0.9909	(1,1,1,1,0,0,0)	2	0.9872	0.9963
0.999	(3,3,2,2,1,1,1)	3	0.9995	(1,1,1,1,0,0,0)	1	0.9994	0.9999

Table 4
$(p_1, p_2, p_3, p_4, p_5, p_6, p_7) = (0.9, 0.9, 0.6, 0.6, 0.6, 0.6, 0.6)$

f	Optimal integer vote assignment	r^1	α^1	Optimal binary vote assignment	r^2	α^2	α^2/α^1
0.001	(1,1,1,1,1,1,1)	7	0.999	(1,1,1,1,1,1,1)	7	0.999	1.0
0.1	(1,1,0,0,0,0,0)	2	0.972	(1,1,0,0,0,0,0)	2	0.972	1.0
0.2	(5,5,1,1,1,1,1)	10	0.955	(1,1,0,0,0,0,0)	2	0.954	0.999
0.3	(4,4,1,1,1,1,1)	8	0.943	(1,1,0,0,0,0,0)	2	0.936	0.993
0.4	(3,3,1,1,1,1,1)	6	0.933	(1,1,0,0,0,0,0)	2	0.918	0.984
0.5	(3,3,1,1,1,1,1)	6	0.933	(1,1,1,0,0,0,0)	2	0.918	0.984
0.6	(3,3,1,1,1,1,1)	6	0.933	(1,1,0,0,0,0,0)	1	0.918	0.984
0.7	(4,4,1,1,1,1,1)	6	0.943	(1,1,0,0,0,0,0)	1	0.936	0.993
0.8	(5,5,1,1,1,1,1)	6	0.955	(1,1,0,0,0,0,0)	1	0.954	0.999
0.9	(1,1,0,0,0,0,0)	1	0.972	(1,1,0,0,0,0,0)	1	0.972	1.0
0.999	(1,1,1,1,1,1,1)	1	0.999	(1,1,1,1,1,1,1)	1	0.999	1.0

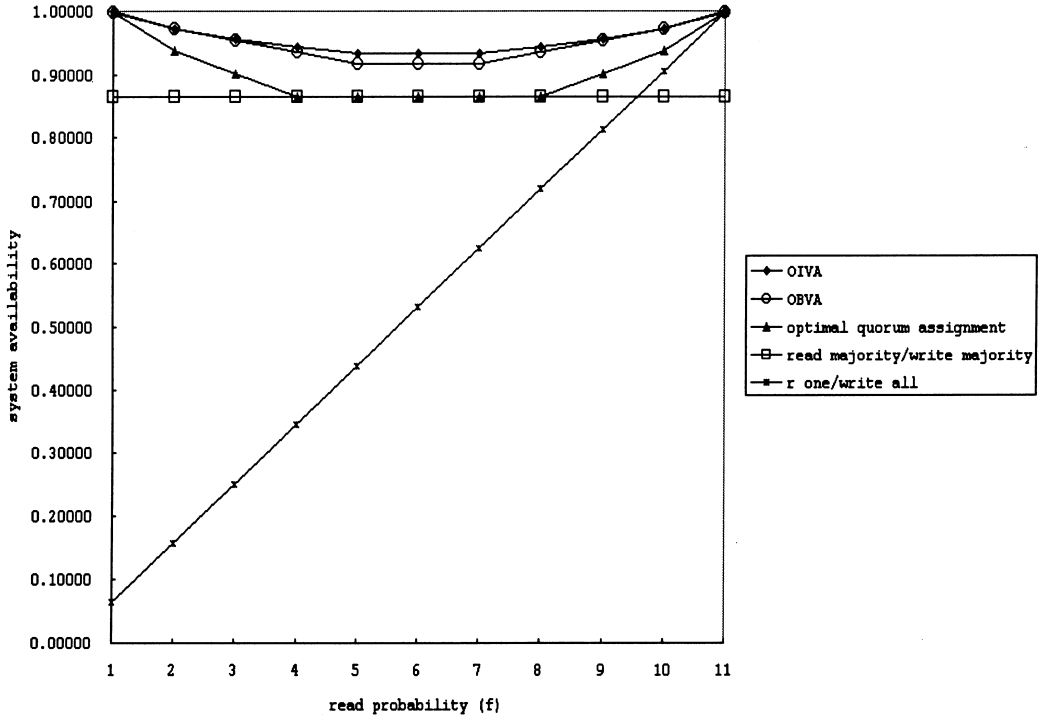

Fig. 1. Comparison of system availability for $(p_1, \ldots, p_7) = (0.9, 0.9, 0.6, 0.6, 0.6, 0.6, 0.6)$.

According to Theorem 2, we can find an optimal assignment from the set $\{Z^i \mid i = 1, \ldots, N\}$. Recalling that for $1 \leqslant q \leqslant i$

$$\alpha(Z^i, q) = \Pr\{\text{at least } q \text{ sites in } S(Z^i) \text{ are available}\}.$$

Thus

$$\alpha(Z^i, 1) = \Pr\{\text{at least 1 sites in } S(Z^i) \text{ are available}\} = 1 - \prod_{j=1}^{i}(1 - p_j)$$

and

$$\alpha(Z^i, i) = \Pr\{\text{at least } i \text{ sites in } S(Z^i) \text{ are available}\} = \prod_{j=1}^{i} p_j.$$

Note that $\alpha(Z^1, 1) = p_1$. For $i > 1$ and $1 < q < i$, $\alpha(Z^i, q)$ is given as following.

Theorem 3. *For all* $i > 1$ *and* $1 < q < i$,

$$\alpha(Z^i, q) = p_i\alpha(Z^{i-1}, q-1) + (1 - p_i)\alpha(Z^{i-1}, q).$$

Proof. By definition,

$$\begin{aligned}
\alpha(Z^i, q) &= \Pr\{\text{at least } q \text{ sites in } S(Z^i) \text{ are available}\} \\
&= \Pr\{(\text{site } i \text{ and at least } q-1 \text{ sites in } S(Z^{i-1}) \text{ are available}) \\
&\quad \text{or } (\text{site } i \text{ fails and at least } q \text{ sites in } S(Z^{i-1}) \text{ are available})\} \\
&= \Pr\{\text{site } i \text{ and at least } q-1 \text{ sites in } S(Z^{i-1}) \text{ are available}\} \\
&\quad + \Pr\{\text{site } i \text{ fails and at least } q \text{ sites in } S(Z^{i-1}) \text{ are available}\} \\
&= p_i\alpha(Z^{i-1}, q-1) + (1 - p_i)\alpha(Z^{i-1}, q). \qquad \square
\end{aligned}$$

To assist in evaluating the values of $\alpha(Z^i, q)$'s, we define $\alpha(Z^i, 0) = 1$, for all $i \geqslant 1$ and $\alpha(Z^i, q) = 0$, for all $1 \leqslant i < q$. Then the values of $\alpha(Z^i, q)$'s, for all $1 \leqslant i \leqslant N$ and $1 \leqslant q \leqslant i$, can be evaluated by the following procedure:

Procedure *Alpha*
$\alpha(Z^1, 1) = p_1$
for $i = 2$ to N do
  for $q = 1$ to i do
    $\alpha(Z^i, q) = p_i\alpha(Z^{i-1}, q-1) + (1 - p_i)\alpha(Z^{i-1}, q)$
  end-for
end-for
end-*Alpha*

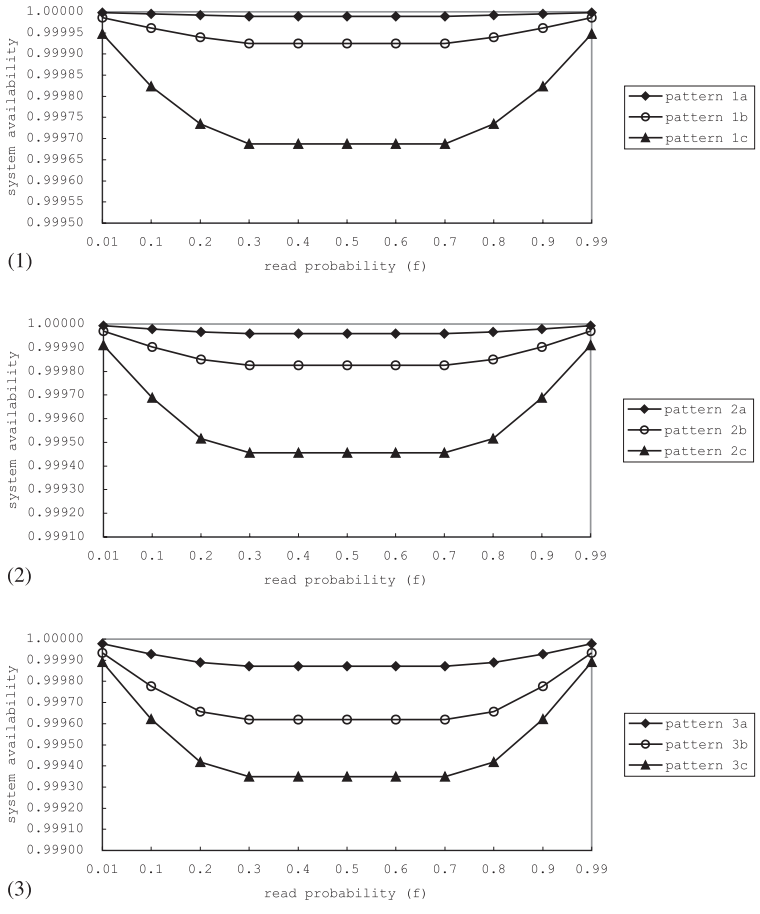

Fig. 2. Comparison of system availability for 12-site systems. (1) Patterns 1a, 1b, 1c. (2) Patterns 2a, 2b, 2c. (3) Patterns 3a, 3b, 3c.

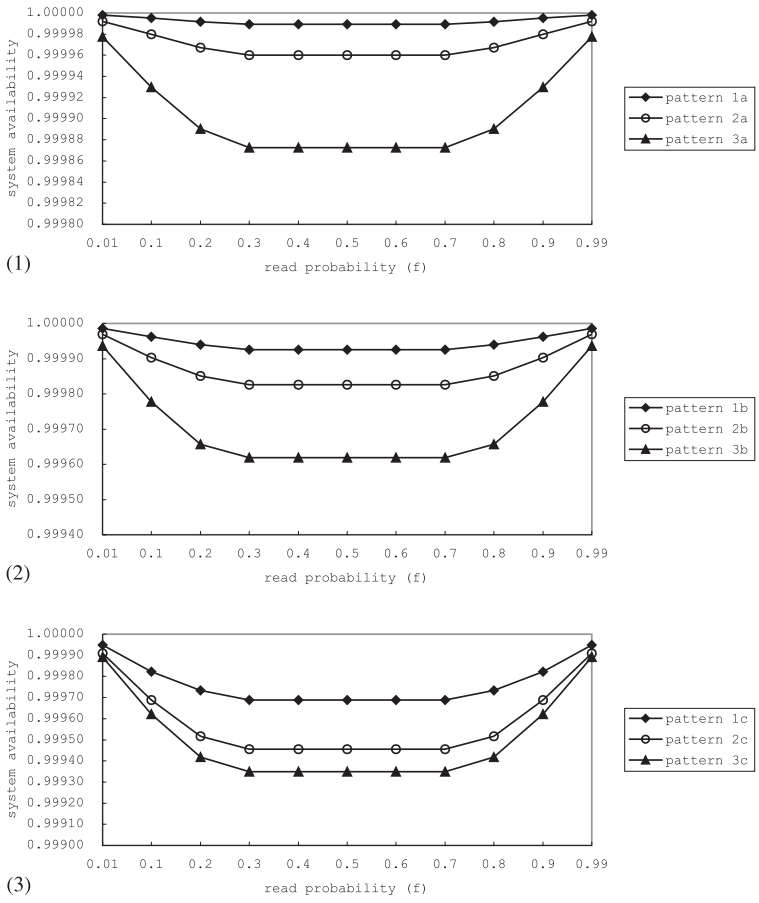

Fig. 3. Comparison of system availability for 12-site systems. (1) Patterns 1a, 2a, 3a. (2) Patterns 1b, 2b, 3b. (3) Patterns 1c, 2c, 3c.

The complexity of Procedure *Alpha* is $O(N^2)$ since the nested two "for" loops spend at most N^2 time units.

Given $0 \leqslant p_N \leqslant \cdots p_1 \leqslant 1$, and $0 \leqslant f \leqslant 1$, the following algorithm can be used to find an optimal binary vote assignment and corresponding quorums.

Algorithm *Opt-Binary*
call Procedure *Alpha*
$A_{\max} = p_1, L_{\text{opt}} = 1, R_{\text{opt}} = 1$
for $i = 2$ to N do
for $r = 1$ to i do[3]
$w = i + 1 - r$
$AVB = f\alpha(Z^i, r) + (1 - f)\alpha(Z^i, w)$
if $AVB > A_{\max}$ then
$A_{\max} = AVB$
$L_{\text{opt}} = i$
$R_{\text{opt}} = r$
end-if
end-for
end-for
end-*Opt-Binary*

[3] When condition $2w > i$ is added, this statement may be modified as "for $r = 1$ to $\lfloor i/2 \rfloor$ do". The remainder of the algorithm need not be changed.

The final values of $A_{\max}$, L_{opt} and R_{opt} denote the maximized availability, optimal degree of replication (number of copies) and read quorum, respectively. Note that OBVA is $Z^{L_{\text{opt}}}$ and according to Condition (3), the write quorum corresponding to the optimal assignment is $w_{\text{opt}} = L_{\text{opt}} + 1 - r_{\text{opt}}$.

Theorem 4. *Given* $0 \leqslant p_N \leqslant \cdots \leqslant p_1 \leqslant 1$, *and* $0 \leqslant f \leqslant 1$, *Algorithm Opt-Binary can find an optimal assignment and corresponding quorum within* $O(N^2)$ *time.*

Proof. Since Procedure *Alpha* needs $O(N^2)$ time and the nested two "for" loops also requires $O(N^2)$ time. Thus the total complexity is $O(N^2)$. □

4. Experimentation

In this section, we compare OBVA with OIVA proposed by Cheung et al. (1989) and other comparable works. Tables 1–4 illustrate the following measures of several experimental results:

- r^1: read quorum corresponding to OIVA;
- α^1: availability of OIVA;

- r^2: read quorum corresponding to OBVA;
- α^2: availability of OBVA.

The compared data of OIVA is derived from Cheung et al. (1989). The system considered in Table 1 consists of five sites where four sites have the same availability, 0.8, and the availability of the other site is 0.9. It is shown that OBVA and OIVA are nearly the same.

Table 2 considers a system consisting of five sites where three sites have the same availability, 0.8, and the availability of the other two sites is 0.9. α^2/α^1 is not less than 99.56% in Table 2.

In Table 3, a system consisting seven sites is considered. The availability of sites is distributed uniformly from 0.95 to 0.65 and $\alpha^2/\alpha^1 \geqslant 99.15\%$.

A system consisting of seven sites is considered in Table 4. Five sites have the same availability 0.6 and the availability of the other two sites is 0.9. In this case, OIVA is quite different from OBVA and the smallest value of α^2/α^1 is 98.4%, which is lowest in Tables 1–4. However, note that, the number of copies required by OBVA is less than OIVA. This is because OIVA assign the lowly available sites (with availability 0.6) a small vote (one) rather than zero in OBVA.

Several notable observations from Tables 1–4 are:

- The availability of OBVA is not less than 98.4% of OIVA in those cases.
- When the sites have nearly the same availability, OBVA and OIVA are almost the same (Table 1).
- If the availability of the sites is distributed uniformly (Tables 2 and 3), OIVA is different from OBVA but OBVA can achieve almost the same availability of OIVA.
- If some sites are highly available and the others are lowly available (Table 4), the availability of OBVA is a little lower than that of OIVA (within 98.4% in those examples); however, the number of copies required by OBVA is quite less than that required by OIVA.

In Fig. 1, we compare OBVA with OIVA (Cheung et al., 1989), the optimal quorum assignment with uniform vote assignment (Ahamad and Ammar, 1989), read majority/write majority, and read one/write all (Ahamad and Ammar, 1989). The values of the other algorithms shown in Fig. 1 are derived from Cheung et al. (1989). It is shown that OBVA performs very closely to OIVA and better than the others.

The algorithm for OIVA is based on an enumeration and an exhaustive search of all vote assignable read

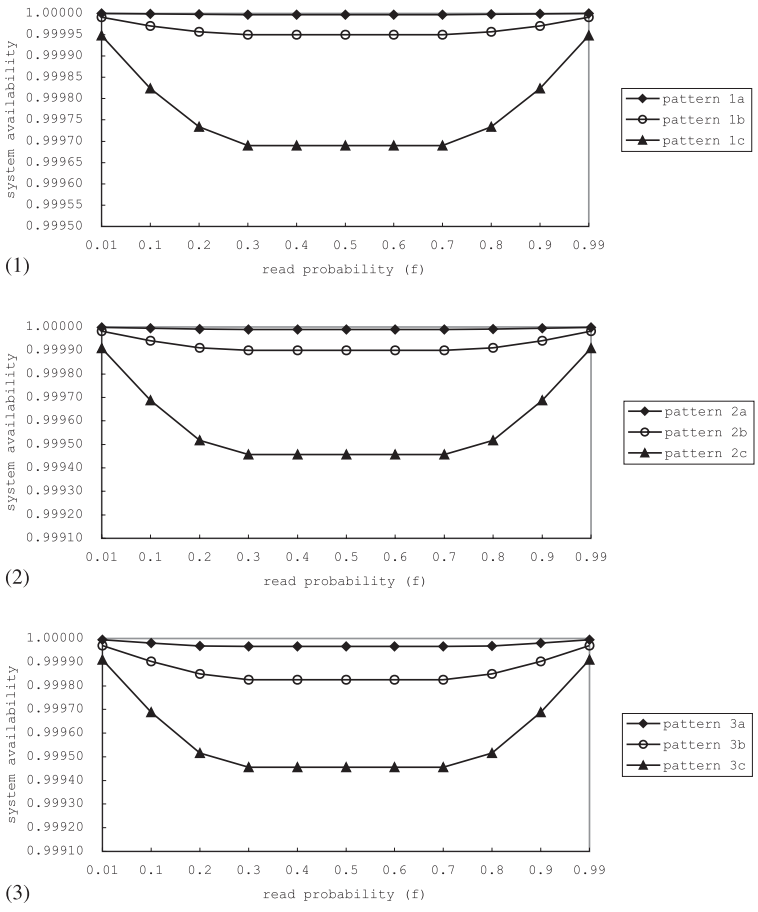

Fig. 4. Comparison of system availability for 18-site systems. (1) Patterns 1a, 1b, 1c. (2) Patterns 2a, 2b, 2c. (3) Patterns 3a, 3b, 3c.

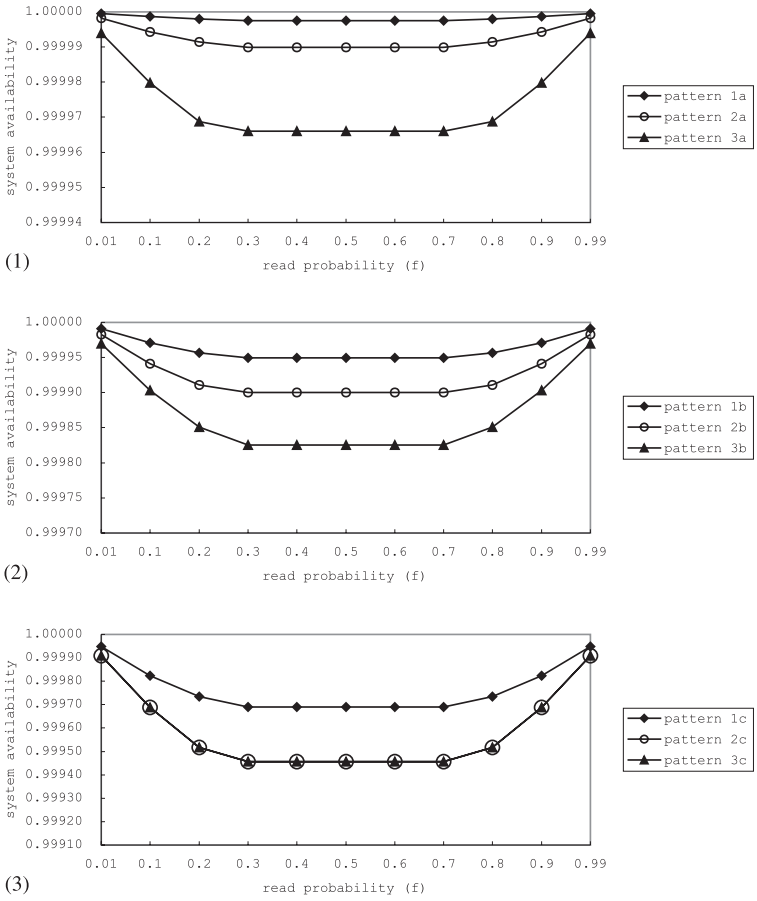

Fig. 5. Comparison of system availability for 18-site systems. (1) Patterns 1a, 2a, 3a. (2) Patterns 1b, 2b, 3b. (3) Patterns 1c, 2c, 3c.

coteries for all valid read quorums. Basically, this follows the work of Garcia-Molina and Barbara (1985), which discussed the problem of how to assign votes to achieving optimal performance with majority voting. It was shown in Garcia-Molina and Barbara (1985) that the number of (majority) vote assignable coteries is $O(2^{N^2})$. Since the algorithm of Cheung et al. (1989) needs an enumeration of all vote assignable read coteries for all read quorums (including majority), the complexity of the enumeration is at least $O(2^{N^2})$. This is unfeasible for large scale systems. In the contrast, our algorithm needs only $O(N^2)$ time. For large systems, our algorithm would be preferred.

To do more experiments with the sensitivity of OBVA, we also apply OBVA to different availability patterns of 12-site and 18-site systems. Four availability patterns used to experiment on 12-site systems are:

1. $p_1 = 0.99$,
   $p_k = p_{k-1} - d$ for $k = 2, \ldots, 12$.
2. $p_1 = 0.99$,
   $p_7 = p_6 - 0.5$,
   $p_k = p_{k-1} - d$ for $k = 2, \ldots 6, 8, \ldots, 12$.
3. $p_1 = 0.99$,
   $p_5 = p_4 - 0.5$,
   $p_9 = p_8 - 0,5$,
   $p_k = p_{k-1} - d$ for $k = 2, \ldots, 4, 6, \ldots, 8, 10, \ldots, 12$.

The availability patterns for 18-site systems are:

1. $p_1 = 0.99$,
   $p_k = p_{k-1} - d$ for $k = 2, \ldots, 18$.
2. $p_1 = 0.99$,
   $p_{10} = p_9 - 0.5$,
   $p_k = p_{k-1} - d$ for $k = 2, \ldots 9, 11, \ldots, 18$.
3. $p_1 = 0.99$,
   $p_7 = p_6 - 0.5$,
   $p_{13} = p_{12} - 0.5$,
   $p_k = p_{k-1} - d$ for $k = 2, \ldots, 6, 8, \ldots, 12, 14, \ldots, 18$.

According to the values of d, each of the above patterns has three subpatterns:

a. $d = 0.01$;
b. $d = 0.015$;
c. $d = 0.02$.

The experimental results are illustrated in Figs. 2–5. Figs. 2 and 3 plot the system availability against the read probability (f) for different patterns of 12-sites systems. Figs. 4 and 5 show similar experiments on 18-sites systems. The figures show that the system availability is highly dependent on the availability pattern of the sites in the system. One notable observation is that the system

availability is very sensitive to the read probability. In other words, read probability is a significant factor that can affect the system availability. We note that, in the case of large f or small f, OBVA performs better than in the other cases. The reason is that, when f is large (or small), OBVA can tune the read (and write) quorum to achieve high performance on the read (or write) operations by sacrificing the performance on the write (or read) operations. On the other hand, when the gap between the read probability and the write probability is close on, the system performance cannot be tuned up by sacrificing any kind of operations.

5. Concluding remarks

The availability is an important metric for replicated data. With voting, the availability is determined by the vote assignment. In previous works, the optimal assignment with integer votes was studied. Although OIVA can achieve the maximal availability, it requires $O(2^{N^2})$ time. In practice, the time required to find OIVA for a system having more than seven nodes is not acceptable. In this paper, we propose OBVA providing an efficient way to find the optimal assignment with binary votes. The proposed algorithm requires only $O(N^2)$ time to find OBVA. The trade-off of the time saving is the decrease in availability. We compare OBVA with OIVA and other related works. It is shown that OBVA can achieve nearly the same availability of OIVA and the availability of OBVA is higher than that of the others. A binary vote assignment can also be viewed as an allocation for the copies of the replicated data. In this aspect, a node with vote 'zero' will not be allocated a replicated copy. That is, replicated copies are allocated only to the nodes having vote 'one'. Thus, for a given set of nodes, the number of copies required by OBVA is usually less than that required by OIVA. As a result, OBVA may also reduce the communication and storage cost.

To sum it up, OIVA can achieve the maximal system availability, but the computational time, $O(2^{N^2})$, is not acceptable for large systems. On the other hand, OBVA can achieve near-optimal availability with less computational time. Consider the example of $N = 7$, OIVA requires more than 10^{14} time units and OBVA requires about only 10^2 time units. Thus OBVA would be preferred for large systems.

In the future, we plan to investigate heuristic algorithms on other voting schemes or on other performance metrics.

Acknowledgements

The authors would like to thank the anonymous reviewers. Their valuable comments lead a significant improvement of the paper. This work was supported partially by the National Science Council on the grant number NSC-88-2520-S-182-005.

References

Ahamad, M., Ammar, M.H., 1989. Performance characterization of quorum consensus algorithms for replicated data. IEEE Trans. Soft. Eng. 15, 492–496.

Amir, Y., Wool, A., 1998. Optimal availability quorum systems: theory and practice. Information Process. Lett. 65, 223–228.

Barbara, D., Garcia-Molina, H., 1987. The reliability of voting mechanisms. IEEE Trans. Comput. 36, 1197–1208.

Bernstein, P.A., Goodman, N., 1981. Concurrency control in distributed database systems. ACM Computing Surveys 13, 185–221.

Bernstein, P.A., Goodman, N., 1985. Serializability theory for replicated databases. J. Comput. Syst. Sci. 31, 355–374.

Cheung, S.Y., Ahamad, M., Ammar, M.H., 1989. Optimizing vote and quorum assignments for reading and writing replicated data. IEEE Trans. Knowledge Data Eng. 1, 387–397.

Davidson, S.B., Garcia-Molina, H., Skeen, D., 1985. Consistency in partitioned networks. ACM Computing Surveys 17, 341–370.

Garcia-Molina, H., Barbara, D., 1985. How to assign votes in a distributed system. J. ACM 32, 841–860.

Gifford, D.K., 1979. Weighted voting for replicated data. In: Proceedings of the Seventh Symposium on Operating System Principles, pp. 150–162.

Spasojevic, M., Berman, P., 1994. Voting as the optimal pessimistic scheme for managing replicated data. IEEE Trans. Parallel Distributed Syst. 5, 64–73.

Tang, J., Natarajan, N., 1989. A static pessimistic scheme for handling replicated databases. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 389–398.

Tang, J., Natarajan, N., 1993. Obtaining coteries that optimizing the availability of replicated databases. IEEE Trans. Knowledge Data Eng. 5, 309–321.

Thomas, R.H., 1979. A majority consensus approach to concurrency control for multiple copy databases. ACM Trans. Database Syst. 4, 180–209.

Tong, Z., Kain, R.Y., 1991. Vote assignments in weighted voting mechanisms. IEEE Trans. Comput. 40, 664–667.