ELSEVIER

Information Sciences 119 (1999) 107–123

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Minimal cost replicated file system with an availability constraint

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

[a] *Department of Information Management, Chang Gung University, 259 Wen-Haw 1st Road, Kwei-Shan, Tao-Yuan 333, Taiwan, ROC*

[b] *Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC*

Received 1 December 1998; received in revised form 24 May 1999; accepted 4 July 1999

Communicated by Frank Shih

Abstract

In a replicated file system, copies of the same file are kept in different servers so that failures of some servers can be tolerant. The goal of a replicated file system is to increase the probability that an operation (read or write) can be performed, i.e., to increase the *availability* of the file. Giving a minimal acceptable availability of a file, the objective is to minimize the total cost required to build a replicated file system that satisfies the availability constraint. In this paper, an efficient algorithm is proposed to find the minimal cost replicated file system.

Keywords: Replicated file; Availability; Cost

1. Introduction

A replicated file system consists of a set of file servers wherein each server stores a copy of the replicated file. The servers in a replicated file system can be unreliable. When a server fails, the copy at the server becomes unavailable. The goal of a replicated file system is to tolerate failures of some servers, or, more precisely, to increase the probability that an operation (read or write) can be performed, i.e., to increase the *availability* of the file.

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

0020-0255/99/$ - see front matter
PII: S0020-0255(99)00024-9

The issue of replica control occurs when multiple copies of a replicated file are stored at different servers. The goal of replica control is to maintain the consistency among copies of the replicated file, i.e., to guarantee that the multiple copies of the replicated file behave like a single copy [3–5,9].

The operations to the file are classified as either *read* or *write*. A replica control algorithm is required to synchronize read and write operations to maintain the consistency of the replicated copies. A survey of replica control algorithms can be found in [5]. In general, a replica control algorithm must ensure two things:

1. Each pair of read and write operations is not allowed to perform at the same time.
2. Each pair of write operations is not allowed to perform at the same time.

Quorum consensus [6] is a popular solution for replica control. To perform a read operation, a read quorum of r copies is required to be accessed. On the other hand, a write operation is required to access w copies. To ensure consistency, $r + w$ and $2w$ must be greater than the total number of replicated copies.

When the quorum of each operation is a majority of all copies, it is called *majority consensus* [12]. With majority consensus,

$$r = w = \left\lceil \frac{n+1}{2} \right\rceil,$$

where n is the number of replicated copies. Note that majority consensus is a special case of the generic quorum consensus algorithm.

The problem of determining the optimal read quorum to maximize the availability of a quorum consensus system was studied by Ahamad and Ammar [1]. In [7], Kumar and Malik considered cost minimization for majority consensus with an availability constraint. Later, Kumar and Segev [8] studied three optimization models for minimizing communication overhead with different constraints. This paper extends and integrates previous works to a new optimization model that considers cost minimization for generic quorum consensus with an availability constraint. The problem is described as follows.

Let p be the availability of a server, i.e., the probability that the server is operational in the long run. In general, the cost of a server with higher availability is more expensive than that of a server with lower availability. That is, for an individual server, higher availability implies higher cost. Let $C(p)$ be the cost of a server with availability p, then $C(p)$ is an increasing function of p. The cost of a replicated file system is determined by the number of servers and the availability of the servers. Let n be the number of servers, then the total cost is $nC(p)$. Giving a minimal acceptable availability, a designer of a replicated file system is required to achieve the availability and to reduce the cost as much as possible. The objective of this study is to design an efficient algorithm that determines the number and the availability of servers to minimize the total cost of a replicated file system for quorum consensus with an availability constraint.

The remainder of this paper is organized as follows. Section 2 describes the model. The algorithm is presented in Section 3. Section 4 illustrates some numerical results of our algorithm. The computational complexity of our algorithm is discussed in Section 5. The final section concludes this paper.

2. Model

The analysis of this paper is based on the following assumptions:

Assumption 1. The file system is homogeneous.

Assumption 2. The number of servers is bounded.

The reasons of these assumptions are:

- A *homogeneous* system consisting of identical servers is preferred to a *heterogeneous* system consisting of various types of servers. One advantage of a homogeneous system is ease to maintain, since any failed server can be repaired by the same manufacturer and the standby spare units, if any, can be used to replace any failed server. On the other hand, maintaining a heterogeneous system is costly, since different servers may be produced by different manufacturers and the spare units are different.
- The communication overhead of a replicated file system increases as the number of servers grows. A designer can set a limit to the number of servers so that the communication overhead is acceptable.

A server is either *operational* or *failed* and the state (operational, or failed) of each server is statistically independent to the others. The *server availability* is the probability that a server is operational at any time instant. When a server is operational, the copy at the server is available; otherwise, it is unavailable. The operations are classified as either *read* or *write*. Quorum consensus is used to synchronized read and write operations in a replicated file system. Thus, read and write operations are required to access at least r and w servers, respectively.

Let n be the number of servers, the following conditions are required to ensure consistency:

$$1 \leqslant r, w \leqslant n, \tag{1}$$

$$r + w = n + 1, \tag{2}$$

$$2w > n. \tag{3}$$

Condition 2 inhibits concurrent execution of any pair of read and write operations and Condition 3 prevents two write operations from concurrently executing. Condition 2 also ensures that a read operation can access to a most

recently updated copy (updated by the last write operation) of the replicated file [6]. Under Condition 2, a read quorum r can determine a unique write quorum w such that

$$w = n + 1 - r. \tag{4}$$

The service time of any operation is assumed to be negligible, thereby the operation finishes instantaneously. If an operation cannot access the required quorum of servers, the operation aborts. The *read* (*write*) *availability* is defined to be the probability that a read (write) operation can access at least r (w) servers, i.e., the probability that $r(w)$-out-of-n servers are operational. Let $\alpha(n,k,p)$ be the probability that k-out-of-n servers are operational for the given server availability p. Then the read availability and write availability are $\alpha(n,r,p)$ and $\alpha(n,w,p)$, respectively.

The *file availability* is defined to be the probability that an operation can access the required quorum of servers at any time instant. If the probabilities that an operation is read and write are f and $1-f$, respectively, then the file availability $A(n,r,p)$, giving n, r and p, is

$$A(n,r,p) = f\alpha(n,r,p) + (1-f)\alpha(n,w,p), \tag{5}$$

where $w = n + 1 - r$.

In this paper, we use the following notation:

1. Constants:
   - m : maximal number of servers,
   - f : the probability that an operation is read,
   - A_0 : availability cut-off, i.e., the minimal acceptable file availability.
2. Variables and functions:
   - n: number of servers, $1 \leqslant n \leqslant m$,
   - r: read quorum,
   - $w = n + 1 - r$: write quorum,
   - p: server availability, $\frac{1}{2} \leqslant p \leqslant A_0$ [1],
   - $\alpha(n,r,p)$: read availability,
   - $\alpha(n,w,p)$: write availability,
   - $A(n,r,p)$: file availability,
   - $r^*(n,p)$: optimal read quorum, giving n and p,
   - $A^*(n,p) = A(n,r^*(n,p),p)$: maximal file availability, giving n and p,
   - $p^*(n)$: minimal server availability s.t. $A^*(n,p^*(n)) \geqslant A_0$, giving n,
   - $C(p)$: availability cost function, i.e., the cost of a server having availability p,
   - $Z^*(n) = n \times C(p^*(n))$: minimal cost for fixed n,
   - n^*: $Z^*(n^*)$ is the minimal one in $\{Z^*(1),\ldots,Z^*(m)\}$.

[1] The range setting of p does not affect the algorithm. Any reasonable range of p can be set.

Using the notation described above, the problem is

$$\begin{aligned} \text{Minimize} \quad & nC(p) \\ \text{Subject to} \quad & 1 \leqslant n \leqslant m \\ & \frac{1}{2} \leqslant p \leqslant A_0 \\ & A(n,p) \geqslant A_0. \end{aligned}$$

3. Algorithm

The algorithm to find the minimal cost replicated file system consists of three levels:

1. Level 1 (the lowest level) computes $A^*(n,p)$, which is the maximal file availability, given n and p.
2. Level 2 approximates $p^*(n)$, for fixed n, such that

   $$A^*(n,p^*(n)) \geqslant A_0$$

   and

   $$A^*(n,p) < A_0, \quad \text{for all } p < p^*(n).$$
3. Level 3 (the highest level) finds n^* such that $Z^*(n^*)$ is the minimal one in $\{Z^*(1),\ldots,Z^*(m)\}$, where $Z^*(n) = n \times C(p^*(n))$.

3.1. Level 1: computing $A^*(n,p)$

For fixed n and p, the optimal read quorum $r^*(n,p)$ that maximizes the file availability is given in [1]:

$$\begin{aligned} r^*(n,p) = 1, \quad \text{if } \frac{f}{1-f} \geqslant \left(\frac{p}{1-p}\right)^{n-2} = n, \qquad \text{if } \frac{f}{1-f} \\ < \left(\frac{p}{1-p}\right)^{2-n} = \left\lceil \frac{n}{2} - \frac{\ln(f/1-f)}{2\ln(p/1-p)} \right\rceil, \text{ otherwise.} \end{aligned} \tag{6}$$

Thus the maximal file availability for fixed n and p is

$$A^*(n,p) = A(n,r^*(n,p),p). \tag{7}$$

3.2. Level 2: approximating $p^*(n)$

Giving n, a bi-section method can be used to approximate $p^*(n)$:

1. Initially, set $p_u = A_0$, $p_l = 1/2$ and $p = (p_u + p_l)/2$.
2. If $A^*(n,p) \geqslant A_0$ then set $p_u = p$ else set $p_l = p$.

3. Set $p = (p_u + p_l)/2$.
4. Repeat steps 2 and 3 i times.
5. Output p_u.

Clearly, $p_u \geqslant p^*(n) > p_l$, after any number of bi-sections. Since $p_u - p_l = (A_0 - (1/2))(1/2)^i \leqslant (1/2)^{i+1}$, after i bi-sections, so

$$p_u - p^*(n) < \left(\frac{1}{2}\right)^{i+1}. \tag{8}$$

Thus, if i is large enough, p_u can approach to $p^*(n)$ as closely as one wish. The selection of i is discussed in Section 5.

3.3. Level 3: finding n^*

Once $p^*(n)$ can be found, the minimal cost for fixed n is

$$Z^*(n) = n \times C(p^*(n)). \tag{9}$$

Then the minimal cost of required replicated file system can be found in $\{Z^*(1), \ldots, Z^*(m)\}$.

4. Numerical examples

The minimal cost replicated file system depends on m, A_0, f and, of course, the cost function, $C(p)$. This section first illustrates some numerical results of our algorithm applying the cost function proposed in [7]. Then we discuss some basic properties of a cost function and propose a new cost function that satisfies the properties. Numerical examples for the new cost function are shown in the last.

The cost function proposed in [7] is:

$$C(p) = K \times e^{Bp-D} \tag{10}$$

where K, B and D are constants. Figs. 1–5 and Table 1 show the numerical results for $K = 15000$, $B = 15$ and $D = 14$:

- Fig. 1 shows the minimal cost plotted against the number of servers for different values of f. Also, the results are compared with majority consensus. The availability cut-off in Fig. 1 is 0.99. It is shown that the minimal cost decreases as f increases and the generic quorum consensus algorithm always performs better than majority consensus, especially for the number of servers is even, regardless the value of f. Note that the difference between the generic quorum consensus algorithm and majority consensus grows as f increases. To explain the results, recall that majority consensus is a special case of quorum consensus. With majority consensus, the read quorum and the write quorum are the same (the majority). On the other hand, quorum

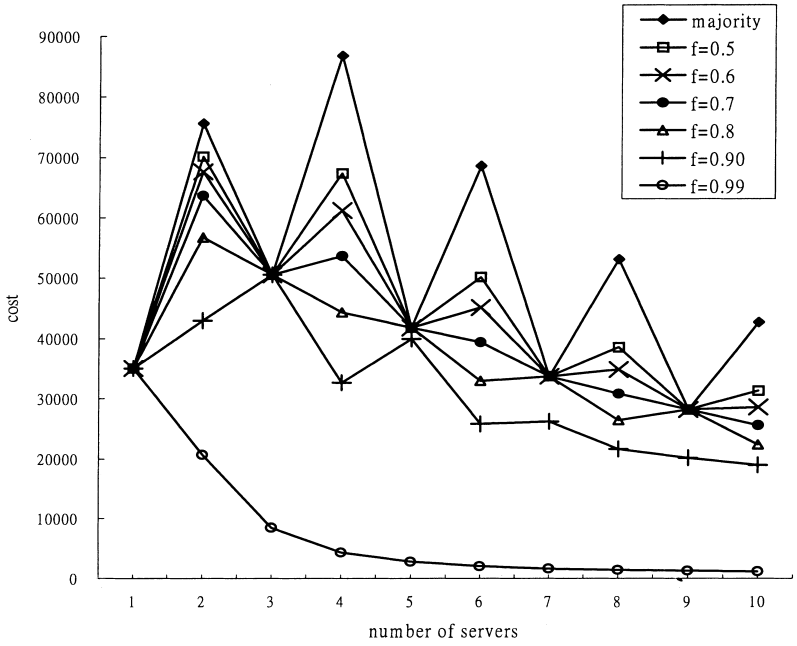

Fig. 1. Cost vs number of servers for different f's, where $C(p) = 15{,}000\ e^{15p-14}$ and cut-off = 0.99.

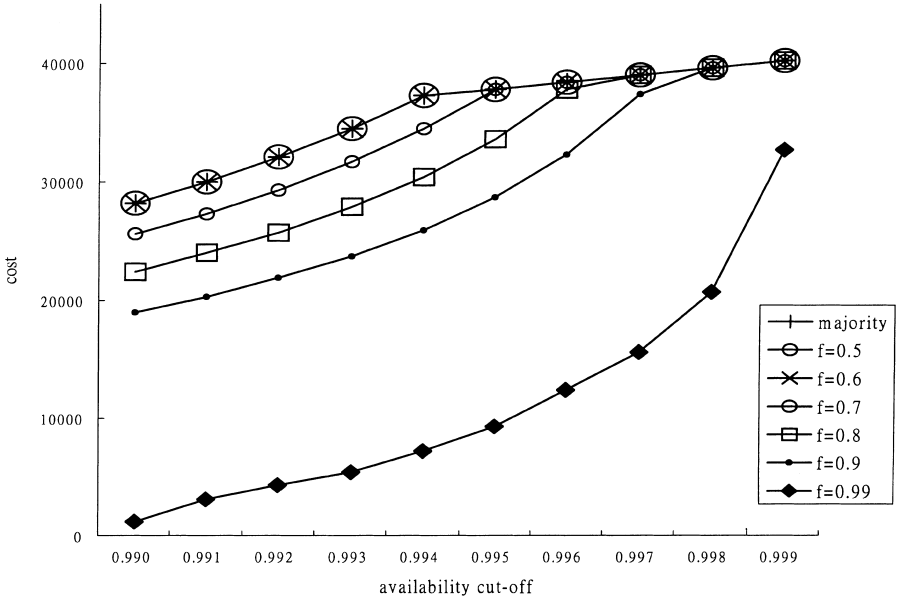

Fig. 2. Cost vs availability cut-off for different f's, where $C(p) = 15{,}000\ e^{15p-14}$ and $m = 10$.

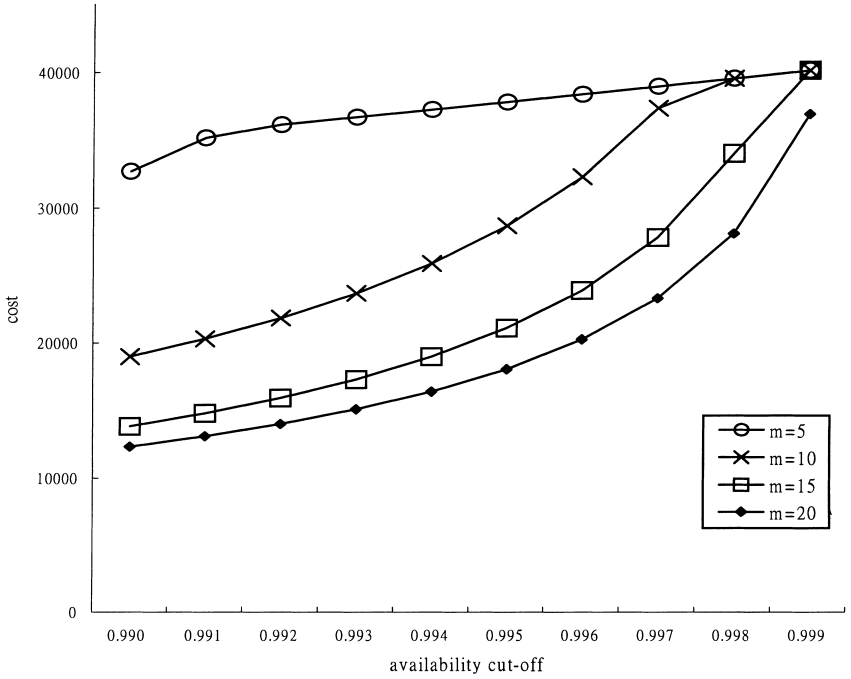

Fig. 3. Cost vs number of servers for different m's, where $C(p) = 15{,}000\ e^{15p-14}$ and $f = 0.9$.

consensus has the flexibility to tune the read (write) quorum to achieve the best performance according to the read probability (f).

- Fig. 2 is a plot of the minimal cost against the availability cut-off, where $m = 10$, for majority consensus and the generic quorum consensus algorithm with different values of f. It is shown that the minimal cost grows as the availability cut-off increases and reduces as f increases. Similar to Fig. 1, the difference between quorum consensus and majority consensus grows as f increases. Note that quorum consensus is dominant for larger f. However, for smaller f, the difference is not obvious.
- Figs. 3 and 4 plot the minimal cost against the availability cut-off for different m's. The values of f in Figs. 3 and 4 are 0.9 and 0.6, respectively. In both figures, the minimal cost decreases as m increases, since a larger m means a larger range of choices.
- Table 1 shows more detailed data, including the actual read quorum, $r^*(n, p^*(n))$, and the server availability, $p^*(n)$, found by the algorithm for $n = 1$ to 20 and different values of f. Fig. 5 depicts the read quorum against f for $n = 5, 10, 15$ and 20. They show that, for each n, the read quorum tends to decrease as f increases. This is because the system availability is a weighted sum of read availability (with weight f) and write availability (with weight

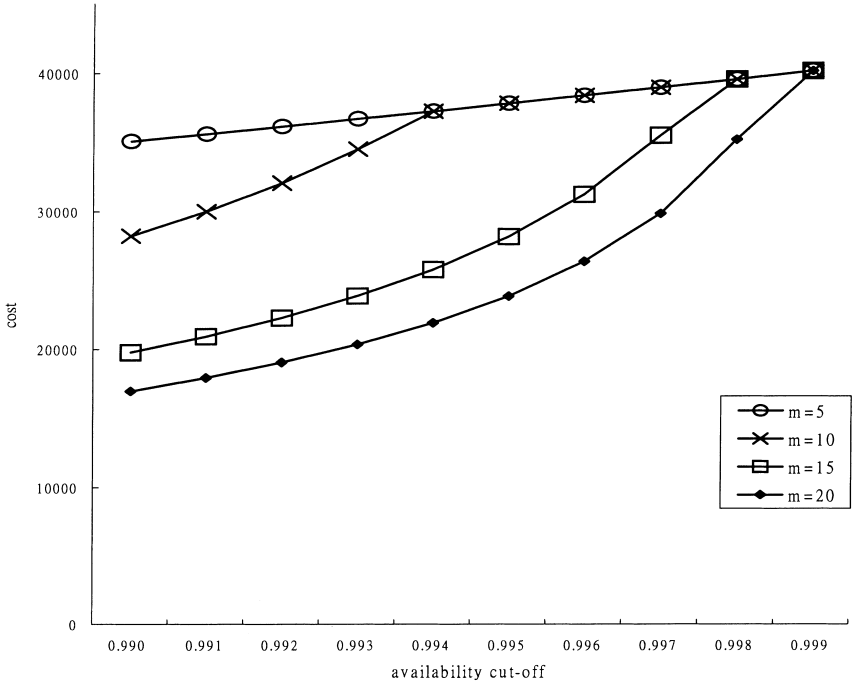

Fig. 4. Cost vs number of servers for different m's, where $C(p) = 15{,}000\ c^{15p-14}$ and $f = 0.6$.

$1 - f$). As the read probability (f) increases, it is better to increase the read availability, by decreasing the read quorum, to achieve a higher system availability. It is worth of note that, for each n, the read quorum drops drastically to 1 as f reaches a critical value (0.99 in the case). The result implies that, if f is large enough (that is most of the operations are intended to read the replicated file system), a group of cheap servers applying read-one/write-all policy ($r = 1$ and $w = n$) is preferred.

The exponential cost function in [7] (Eq. (10)) is more realistic than a linear function, since it increases very steeply as p approaches 1. This means that it is hard to make up a perfect server (with availability 1). There is a little problem concerning with the possibility of having a perfect server. Since Eq. (10) is bounded, it implies that a perfect server is still possible to have. However, it is impossible to obtain a perfect server that never fails. In practice, the cost of a perfect server should be infinite and a server with availability 0 should cost zero. To sum up, the basic properties of a cost function, $C(p)$, are:

1. $C(p)$ is an increasing function of p.
2. $C(0) = 0$.
3. $C(1) = \infty$.

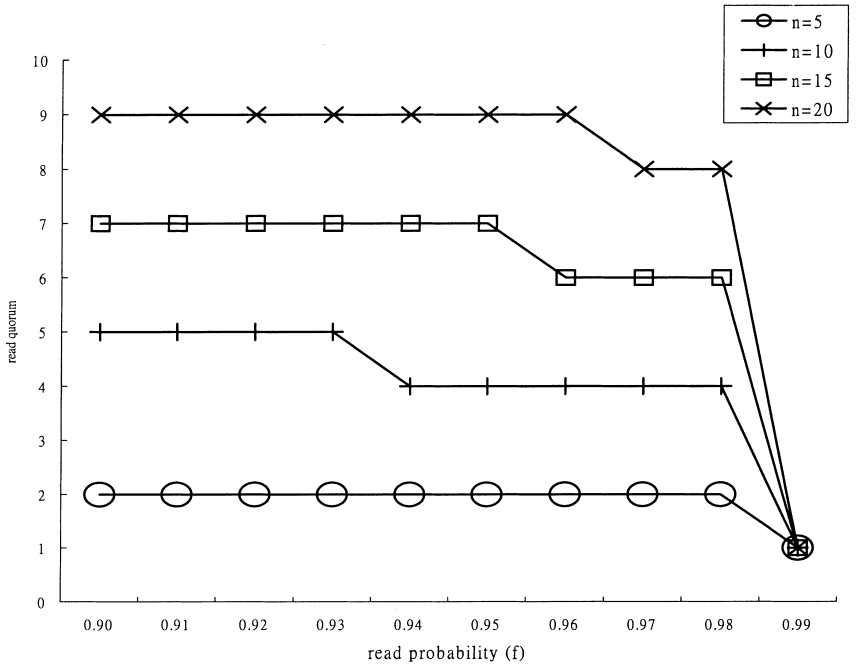

Fig. 5. Read quorum vs read probability (f) for $n = 5, 10, 15, 20$, where $C(p) = 15{,}000\ e^{15p-14}$ and cut-off = 0.990.

Clearly, Eq. (10) satisfies property 1; but it dose not satisfy properties 2 and 3.

We propose a cost function having the required properties:

$$C(p) = \left(\frac{p}{1-p}\right)^{a}, \quad a > 0. \tag{11}$$

Figs. 6–8 show the minimal cost plotted against the number of servers for three examples of Eq. (11), where $a = 1$, $2/3$ and $1/2$, respectively. Several observations are:

- If the cost function grows very fast as the p increases (the case of Fig. 6), or the availability cut-off is very high (cut-off = 0.999 in Figs. 7 and 8), the single-server system is usually more expensive than a multiple-server system.
- If the cost function grows slowly as the p increases and the availability cut-off is low (cut-off = 0.990 and 0.995 in Fig. 8), the cost tends to grow as the number of servers increases.
- In general, the minimal cost replicated file system depends on the cost function and the availability cut-off. Various environments ($C(p)$ and f) and constraints (A_0 and m) can result in different solutions.

Table 1
Read quorum, $r*\ (n, p*\ (n))$, and server availability, $p*\ (n)$, found by the algorithm, where $C(p) = 15000\ e^{15p-14}$ and cut-off $= 0.990$

n	$f = 0.91$		$f = 0.93$		$f = 0.95$		$f = 0.97$		$f = 0.99$	
	$r*$	$p*$	$r*$	$p*$	$r*$	$p*$	$r*$	$p*$	$r*$	$p*$
1	1	0.9900	1	0.9900	1	0.9900	1	0.9900	1	0.9900
2	1	0.9541	1	0.9463	1	0.9364	1	0.9239	1	0.9087
3	2	0.9411	2	0.9411	1	0.9308	1	0.8913	1	0.8223
4	2	0.8903	2	0.8846	2	0.8782	2	0.8711	1	0.7593
5	2	0.8857	2	0.8715	2	0.8525	2	0.8270	1	0.7148
6	3	0.8482	3	0.8440	3	0.8396	2	0.8180	1	0.6826
7	3	0.8359	3	0.8243	3	0.8104	3	0.7937	1	0.6585
8	4	0.8177	4	0.8145	3	0.8096	3	0.7759	1	0.6398
9	4	0.8026	4	0.7935	4	0.7830	4	0.7709	1	0.6249
10	5	0.7943	5	0.7917	4	0.7762	4	0.7508	1	0.6129
11	5	0.7785	5	0.7712	5	0.7629	4	0.7479	1	0.6029
12	6	0.7756	5	0.7688	5	0.7530	5	0.7334	1	0.5946
13	6	0.7600	6	0.7539	6	0.7471	5	0.7264	1	0.5875
14	6	0.7599	6	0.7490	6	0.7359	6	0.7201	1	0.5813
15	7	0.7451	7	0.7399	7	0.7342	6	0.7111	1	0.5760
16	7	0.7430	7	0.7336	7	0.7224	7	0.7094	1	0.5713
17	8	0.7327	8	0.7282	7	0.7216	7	0.6993	1	0.5672
18	8	0.7293	8	0.7211	8	0.7115	7	0.6982	1	0.5635
19	9	0.7222	9	0.7182	8	0.7089	8	0.6899	1	0.5602
20	9	0.7180	9	0.7106	9	0.7023	8	0.6869	1	0.5572

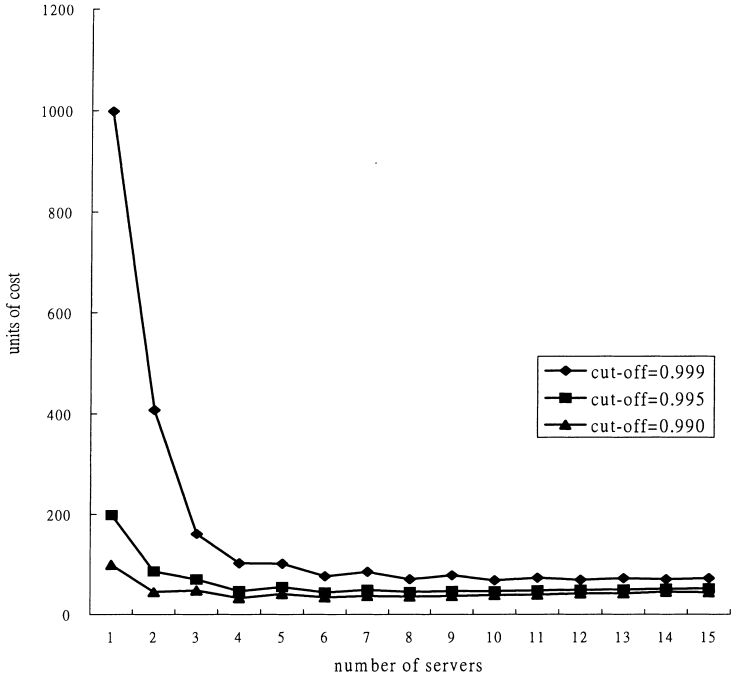

Fig. 6. Cost vs number of servers for different cut-off's, where $C(p) = p/(1-p)$ and $f = 0.9$.

Although various cost functions are possible, our algorithm provides an efficient way, for any cost function, to design a minimal cost replicated file system achieving the required level of availability. The efficiency (complexity) of the algorithm is elaborated in the next section.

5. Complexity analysis

The complexities of the three levels of the algorithm are:

1. $A^*(n,p) = A(n, r^*(n,p), p) = f\alpha(n, r^*(n,p), p) + (1-f)\alpha(n, w^*(n,p), p)$, where $r^*(n,p) + w^*(n,p) = n+1$. Since $r^*(n,p)$ can be found in constant time (Eq. (6)), the complexity of computing $A^*(n,p)$ is equivalent to the complexity of computing $\alpha(n, r^*(n,p), p)$ and $\alpha(n, w^*(n,p), p)$. Note that $\alpha(n,k,p)$ is the availability of a k-out-of-n system. Several $O(nk)$ algorithms for evaluating the availability (or reliability) of a k-out-of-n system can be found in [2,10,11]. Thus the complexity of computing $A^*(n,p)$ is $O(n(r^*(n,p) + w^*(n,p))) = O(n^2)$.

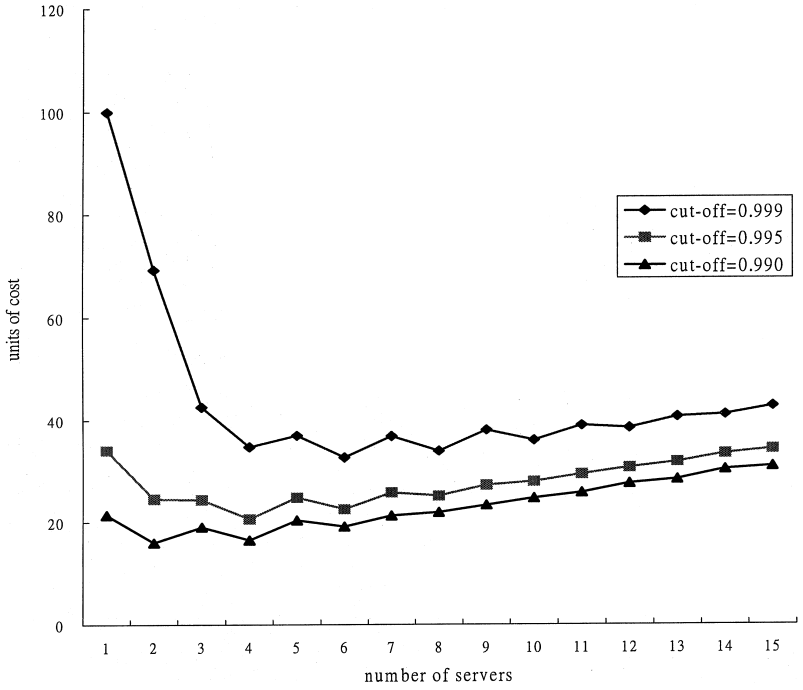

Fig. 7. Cost vs number of servers for different cut-off's, where $C(p) = (p/(1-p))^{2/3}$ and $f = 0.9$.

2. The complexity of approximating $p^*(n)$ is $O(in^2)$, wherein $O(n^2)$ is required to compute $A^*(n,p)$ and i is the number of bi-sections. (The selection of i is discussed later.)
3. Level 3 of the algorithm requires computing $p^*(n)$, for all $n = 1, \ldots, m$. Thus the complexity of the algorithm is $O(i(1^2 + \cdots + m^2)) = O(im^3)$.

The number of bi-sections, i, is determined by the following theorems.

Theorem 1. *For any* $\varepsilon > 0$ *and*

$$C(p) = K \times e^{Bp-D}, \frac{C(p_u) - C(p^*(n))}{C(p^*(n))} < \varepsilon, \quad \text{if } i \geqslant \left\lceil \frac{\ln \ln(1+\varepsilon) - \ln B}{\ln(1/2)} \right\rceil.$$

Proof. The proof is shown in Appendix A. □

Example 1. If $B = 15$, for $\varepsilon = 10^{-3}$ and 10^{-5}, according to Theorem 1, $i \geqslant 13$ and 20, respectively.

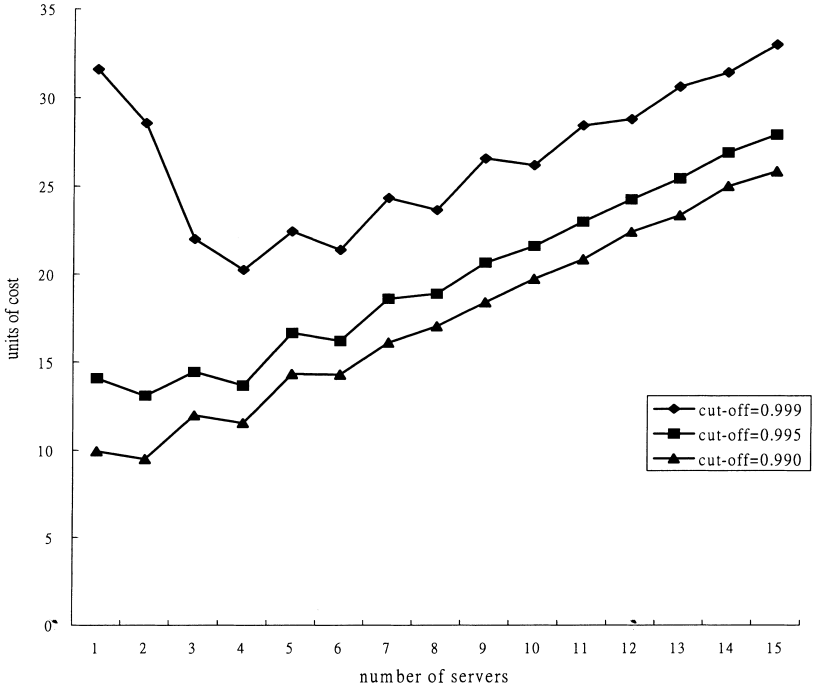

Fig. 8. Cost vs number of servers for different cut-off's, where $C(p) = (p/(1-p))^{1/2}$ and $f = 0.9$.

Theorem 2. *For any* $\varepsilon > 0$ *and*

$$C(p) = \left(\frac{p}{1-p}\right)^{a}, a > 0, \frac{C(p_u) - C(p^*(n))}{C(p^*(n))} < \varepsilon, \quad \textit{if } i \geqslant \left\lceil \frac{\ln \varepsilon - \ln b}{\ln(1/2)} \right\rceil,$$

where

$$b = \frac{aA_0^{a-1}}{(1-A_0)^{a+1}}.$$

Proof. The proof is shown in Appendix A. □

Example 2. If $\varepsilon = 10^{-5}$ and $A_0 = 0.999$, for $a = 1$, $2/3$ and $1/2$, according to Theorem 2, $i \geqslant 37$, 33 and 31, respectively.

Example 3. If $\varepsilon = 10^{-3}$ and $A_0 = 0.990$, for $a = 1$, $\frac{2}{3}$ and $\frac{1}{2}$, according to Theorem 2, $i \geqslant 23$, 21 and 19, respectively.

Above theorems and examples show that the number of bi-sections required to approximate $p^*(n)$ is bounded for giving cost function and required ε.

Therefore i can be trusted as a constant and the complexity of the algorithm is $O(m^3)$. Thus the algorithm can be applied efficiently to design a required system.

6. Conclusion

A replicated file system is used to improve the availability of a file. The availability of the replicated file can be improved by increasing the number or the availability of servers. However, the cost is proportional to the number of servers. Moreover, for each individual server, higher availability implies higher cost. A system analyst usually encounters the problem of designing a minimal cost replicated file system that achieves a required level of availability. Two problems had been studied in previous works: availability maximization for quorum consensus and cost minimization for majority consensus with an availability constraint. This study extends and integrates previous works to a new optimization model considering cost minimization for quorum consensus with an availability constraint. An $O(m^3)$ algorithm is designed to find the optimal solution. Numerical results show that the solution found by our algorithm is always better than that considers only majority consensus. The algorithm can be applied to various cost functions and it can be used to design a new system or redesign an existing system when the environment or the constraint changes.

Acknowledgements

The authors would like to thank the anonymous referees for their constructive comments that lead a significant improvement of the paper.

Appendix A

Proof of Theorem 1.

$$C(p) = K\mathrm{e}^{Bp-D}$$

and

$$\frac{C(p_u) - C(p^*(n))}{C(p^*(n))} = \mathrm{e}^{B(p_u - p^*(n))} - 1.$$

Since $p_u - p^*(n) \leqslant (1/2)^{i+1}$, $(C(p_u) - C(p^*(n)))/(C(p^*(n))) < \varepsilon$, if

$$e^{B(1/2)^{i+1}} < 1 + \varepsilon.$$

That is,

$$i \geqslant \left\lceil \frac{\ln \ln(1+\varepsilon) - \ln B}{\ln(1/2)} \right\rceil. \qquad \square$$

Proof of Theorem 2.

$$C(p) = \left(\frac{p}{1-p} \right)^a$$

and

$$\frac{\mathrm{d}}{\mathrm{d}p} C(p) = a \left(\frac{p}{1-p} \right)^{a-1} \frac{1}{(1-p)^2} = \frac{a p^{a-1}}{(1-p)^{a+1}}.$$

Since $\frac{1}{2} \leqslant p \leqslant A_0$, thus

$$C(p) \geqslant 1$$

and

$$\frac{\mathrm{d}}{\mathrm{d}p} C(p) \leqslant \frac{a A_0^{a-1}}{(1-A_0)^{a+1}}.$$

Let $b = (aA_0^{a-1})/((1-A_0)^{a+1})$, then

$$\frac{C(p_u) - C(p^*(n))}{C(p^*(n))} \leqslant \frac{b(p_u - p^*(n))}{1} = b(p_u - p^*(n)).$$

Thus, $(C(p_u) - C(p^*(n)))/(C(p^*(n))) < \varepsilon$, if $b(p_u - p^*(n)) < \varepsilon$. Since $p_u - p^*(n) \leqslant (1/2)^{i+1}$, $b(p_u - p^*(n)) < \varepsilon$, if $b(1/2)^{i+1} < \varepsilon$, i.e.,

$$i + 1 > \frac{\ln \varepsilon - \ln b}{\ln(1/2)}.$$

Or, equivalently,

$$i \geqslant \left\lceil \frac{\ln \varepsilon - \ln b}{\ln(1/2)} \right\rceil. \qquad \square$$

References

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

[2] R.E. Barlow, K.D. Heidtmann, Computing *k*-out-of-*n* system reliability, IEEE Trans. Reliability 33 (1984) 322–323.

[3] P.A. Bernstein, N. Goodman, Concurrency control in distributed database systems, ACM Comput. Surveys 13 (1981) 185–221.
[4] P.A. Bernstein, N. Goodman, Serializability theory for replicated databases, J. Comput. System Sci. 31 (1985) 355–374.
[5] S.B. Davidson, H. Garcia-Molina, D. Skeen, Consistency in partitioned networks, ACM Comput. surveys 17 (1985) 341–370.
[6] D.K. Gifford, Weighted voting for replicated data, in: Proceedings of 7th Symposium on Operating System Principles, ACM SIGOPS, 1979, pp. 150–162.
[7] A. Kumar, K. Malik, Voting mechanisms in distributed systems, IEEE Trans. Reliability 40 (1991) 593–600.
[8] A. Kumar, A. Segev, Cost and availability tradeoffs in replicated data concurrency control, ACM Trans. Database Systems 18 (1993) 102–131.
[9] M.T. Ozsu, P. Valduriez, Distributed and parallel database systems, ACM Comput. Surveys 28 (1996) 125–128.
[10] A.M. Rushdi, Utilization of symmetric switching functions in the computation of k-out-of-n system reliability, Microelectronics and Reliability 26 (1986) 973–987.
[11] A.K. Sarje, E.V. Prasad, An efficient non-recursive algorithm for computing the reliability of k-out-of-n systems, IEEE Trans. Reliability 38 (1989) 234–235.
[12] R.H. Thomas, A majority consensus approach to concurrency control for multiple copy databases, ACM Trans. Database Systems 4 (1979) 180–209.