A Performance Study of a New Grid Protocol and General Grid Structures for Replicated Data

Akhil Kumar[†] [*], Michael Rabinovich[‡] [†], and Rakesh K. Sinha[‡] [‡]
[†] Graduate School of Management
Cornell University
Ithaca, NY 14853
[‡] Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195

Abstract

Recently there has been considerable interest in the study of replica-control protocols which are based on organizing several copies of an object into logical structures, such as rectangular grids. In addition to high availability, another objective in exploiting such structures is to improve the degree of load sharing in a system. In this paper, we extend the scope of grid structures to general grids, which allow holes in various positions of a rectangular structure and are useful to consider because they often produce availabilities that are higher than solid grids, where every position must be occupied by a node. We propose an improvement to the existing grid protocol, prove its optimality, and also compare its performance with the existing protocol in terms of availability. In addition, we also offer new insights into the performance of the grids, from both availability and load sharing points of view. Noting that the write availability of square grids tends to 0 for a large number of nodes N, we conduct an asymptotic analysis, and derive the conditions that must be imposed on the dimensions of the grid for the availability to increase asymptotically with increasing N. Algorithms for designing grids to maximize availability independently, and also in conjunction with a load sharing constraint are given.

Key words: Replicated Databases - Distributed systems - Quorums - Grid Protocol Availability - Load sharing - Algorithms.

1 Introduction

Data is replicated in distributed systems to improve availability and performance. In most cases, the consistency of the data must be maintained despite node and/or communication link failures. This can be achieved by requiring that, in order to succeed, read and write operations obtain permission from certain sets of replicas called read and write quorums. The read and write quorums are defined in such a way that any two write quorums and any pair of read and write quorums have at least one node in common. Then, a read operation can always identify the most recent version of the data, and write conflicts are also

[*]Work was supported in part by NSF Grant IRI-9110880.
[†]This author's research is supported in part by the NSF Grant No. CCR-9200832.
[‡]Research supported by NSF/DARPA under grant CCR-8907960 and NSF under grant CCR-8858799

prevented. Quorums are also used in distributed systems for many other purposes such as mutual exclusion [1, 11], commit protocols [13] and distributed consensus [9].

In addition to higher availability, a second advantage of replication is greater load sharing. If there is only one copy of an object, and all operations go to the node where it is stored, the node may get overloaded and the response time may deteriorate. By maintaining multiple copies, the load can be distributed uniformly across the various copies. In majority voting [14], the load sharing is not very good because half of the copies must be involved in a quorum. Hence, the maximum improvement in load sharing is limited to a factor of 2, and does not change as the number of copies increases. On the other hand, the solutions described in this paper have better load sharing properties.

A desirable property for an algorithm to have is to be *fully distributed*. This is because in such an algorithm, each node plays an equal role, and the load distribution is uniform. More precisely, in a fully distributed solution all write quorums are of equal size, all read quorums are of equal size, and every node is a member of an equal number of read and write quorums. An important goal in designing such an algorithm is to minimize the quorum size as a function of the number of replicas because the cost of performing an operation depends upon quorum size. Provably the best (with regards to quorum size) fully distributed solutions have a quorum size of $\Omega(\sqrt{N})$, where N is the number of replicas of the data item. Although this bound is achieved in *Maekawa's* [11], *Grid* [2, 3], and *Hierarchical Grid* (h-grid) [7] protocols, each of these protocols has some drawback. Maekawa's protocol has the lowest constant among the three, but it gives poor availability. On the other hand, the grid protocol performs better than Maekawa's protocol in terms of availability, while the h-grid protocol has an identical quorum size to the grid protocol and also gives asymptotically high write availability (which the grid protocol does not). However, the higher write availability in the h-grid protocol is obtained at the expense of read availability. This means that one does not dominate the other. Another protocol described in [12] which combines Maekawa's protocol and majority voting in a novel manner has a quorum size of $\sqrt{N \log N}$ and also produces asymptotically high availability.

In the grid protocol [2, 3], N copies are logically organized into a rectangular $m \times n$ grid. A read quorum is formed by assembling one copy from each column of the grid (also called a *column-cover* or *c-cover*); a write quorum is formed by the union of a read quorum and all copies from any one column of the grid. Grid protocols are promising because they are fully distributed, and, moreover, have the potential of satisfying both the load sharing and high availability requirements. In this paper, we describe a modified grid protocol and give new results and analysis that offer fresh insights into the availability and load sharing behavior of grids. We also extend the scope of our results and analysis to *general* grids, by which we refer to arrangements where a large number of copies of an object are organized into a rectangular structure; however, some of the positions in the rectangle are allowed to be empty (we call such positions "holes"). Such grids with holes will be referred to as *hollow* as opposed to *solid* grids which have copies in *all* positions.

Consequently, this paper discusses several issues pertaining to both *solid* and *hollow* grids. First, we modify the grid protocol described in [2, 3] slightly to improve its availability properties, and also derive the expressions for read and write availability in the case of a *general* grid. This exercise is interesting because, as we show in Section 4, often the availability resulting from organizing N copies into an $m \times n$ grid (where $mn = N$) can be improved by placing the N copies in an $m' \times n'$ grid (where $m'n' > N$) and leaving some grid positions empty (i.e., filling them with holes). Of course, permitting holes in a grid diminishes the fully distributed nature of the grid protocol because it means that some nodes play a larger role than others. We resolve this issue by permitting at most one hole in any column of the grid and so the algorithm remains "almost fully distributed". Therefore, for large N, the load is distributed among the nodes almost as evenly as in a fully distributed solution, and yet there is more flexibility in constructing the grid. In fact, for many values of N (e.g., when N is prime), no sensible *fully* distributed solution exists anyway.

The second issue we address is that of availability. We examine it from both the theoretical and empirical points of view. As shown in [7], the write availability for square grids goes asymptotically to 0; however, the question we pose is: If rectangular, non-square grids are permitted, then, would the asymptotic availability go to 1? What must the dimensions of the grid be for this to happen? We find that for this to happen, the dimensions of the grid must be close to $\log N \times N/\log N$, and this produces quorum sizes that are almost linear in N, thereby defeating the main strength of grids. Interestingly enough, empirical calculations show that this asymptotic behavior of grids comes into play only for *very large* values of N. It was found that for $N \leq 23000$, grid dimensions, $\sqrt{N} \times \sqrt{N}$ and the probability p of a node being up 0.99, the availability does increase towards 1, and begins to decline only for values of N larger than 23000. This means that, in spite of the negative asymptotic result mentioned above, rectangular grids are still very promising both from availability and load sharing points of view in most practical situations. We also studied the relationship between availability and N for various values of p, the probability that a node is up.

Another important issue addressed here is that of grid design subject to availability and load sharing constraints. Given a desired degree of load sharing and a minimum availability threshold, what is the smallest number of copies N required to satisfy these requirements? Moreover, how should these copies be arranged in an $m \times n$ grid? An algorithm is proposed to enable a designer to solve this problem and meet the twin objectives of load sharing and availability in an optimal manner.

This paper is organized as follows. Section 2 gives our notation and terminology. Section 3 describes a general grid and derives expressions for computing read and write availability of such a grid. In this section, a modified grid protocol and a proof of its optimality are also given. In section 4, we compare the existing and our modified grid protocols in terms of availability. Section 5 gives an algorithm for constructing a grid with the maximal availability given N copies of an object. Section 6 analyzes availability of general grids in the asymptotic case (for large N), both theoretically and empirically. Next, Section 7 turns to the issue of designing a grid which satisfies both availability and load sharing constraints. Section 8 concludes the paper.

2 Terminology and notation

Throughout this paper, the terms "node" and "site" are used, often interchangeably, to refer to physical network sites where the data is replicated. The term "position", on the other hand, refers to the placeholders in logical grids that may or may not be occupied by physical nodes (Figure 1). A grid in which all positions are occupied by physical nodes is called *solid*. Positions not occupied by nodes are called *holes*. A grid with one or more holes is called a *hollow* grid.

Two grids are equivalent if their read and write quorum sizes and availabilities are the same. A column of a grid is *good* if *all* its nodes are up; it is *alive* if *at least one* of its nodes is up. A column which is not good is *bad*.

The following is standard notation in this paper. The system contains N nodes organized in an $m \times n$ grid, where m is the number of rows and n, the number of columns. A grid may or may not contain all N nodes. As an example, a solid $m \times n$ grid in which $mn < N$ does not use $N - mn$ nodes. A node is assumed to be operational with probability p, and to have failed with probability $q = 1 - p$. Read and write availabilities are denoted by RA and WA, respectively, and represent the probabilities that the corresponding quorums can be formed. Read and write quorum sizes are denoted by Q_R and Q_W. See Table 1 for a summary of the terminology.

Finally, as a measure of the degree of load sharing in the system, we define a *relative read (write) quorum size* to be the ratio of the read (write) quorum size to the total number of nodes used in the grid. The intuition behind this measure is as follows. In a solid grid containing all N nodes, the relative quorum size shows the fraction of nodes participating in a single operation and hence, if different operations choose

Notation	Description
N	total number of nodes in the system
m	number of rows
n	number of columns
p	probability that a given node is up
q	$1-p$
RA	read availability
WA	write availability
Q_R	size of read quorum
Q_W	size of write quorum
column is *good*	*all* nodes in the column are *up*
column is *bad*	*all* nodes in the column are *down*
column is *alive*	at least one node in the column is *up*
relative read quorum size	$\frac{Q_R}{Number\ of\ nodes\ in\ the\ grid}$
relative write quorum size	$\frac{Q_W}{Number\ of\ nodes\ in\ the\ grid}$

Table 1: Summary of terminology used

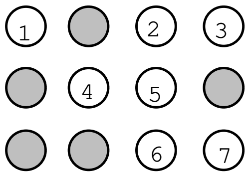

Figure 1: A 3×4 grid with 7 nodes.

quorums randomly, it gives the fraction of the total load carried by a single node. If not all N nodes are included in the grid, the extra nodes never participate in any operation and should not be taken into account in the measure of load sharing.

3 Computing read and write availability for general grids

In Section 3.1, we derive formulas for read and write availability for a general grid, i.e., one in which some positions may not be occupied by nodes (Figure 1 is a 3×4 hollow grid with 7 nodes and 5 holes). In previous works, only solid rectangular grids were considered [2, 3]. However, as we will illustrate in Section 4, hollow grids may provide better availability than solid ones. The computations in Section 3.1 are carried out for the existing grid protocol. Then, Section 3.2 describes the modified grid protocol, and recomputes the read availability for that protocol. Finally, in Section 3.3, we prove that the modified grid protocol is optimal in the sense that it is non-dominated.

3.1 Availability in Existing Protocol

Consider N nodes placed in an $m \times n$ grid where $mn \geq N$. If $mn > N$, then the grid contains $mn - N$ holes. Let n_1 columns contain m_1 nodes, n_2 columns contain m_2 nodes, ..., n_k columns contain m_k nodes, and so on. The read and write availabilities of this grid are calculated as follows.

Prob(a column with m_i nodes is alive) $= 1 - q^{m_i}$;

Prob(all columns in the grid are alive) =

$$(1 - q^{m_1})^{n_1} \cdot (1 - q^{m_2})^{n_2} \cdot \ldots \cdot (1 - q^{m_k})^{n_k};$$

Prob(A column with m_i nodes is bad and alive) =

1− Prob(a column with m_i nodes is good) − prob(a column with m_i nodes is dead) $= 1 - p^{m_i} - q^{m_i}$;

Prob(all columns are bad and alive) =

$$(1 - p^{m_1} - q^{m_1})^{n_1} \cdot (1 - p^{m_2} - q^{m_2})^{n_2} \cdot \ldots \cdot (1 - p^{m_k} - q^{m_k})^{n_k};$$

Write availability WA = Prob(all columns are alive) − Prob(all columns are bad and alive) =

$$\begin{gathered}(1 - q^{m_1})^{n_1} \cdot \ldots \cdot (1 - q^{m_k})^{n_k} - \\ (1 - p^{m_1} - q^{m_1})^{n_1} \cdot \ldots \cdot (1 - p^{m_k} - q^{m_k})^{n_k}. \end{gathered} \tag{1}$$

Read availability RA = Prob(all columns are alive) =

$$(1 - q^{m_1})^{n_1} \cdot (1 - q^{m_2})^{n_2} \cdot \ldots \cdot (1 - q^{m_k})^{n_k}. \tag{2}$$

In the existing grid protocol, there are two tradeoffs: first, between read and write availabilities; and second, between read quorum size (which is a measure of read performance) and write availability [2]. Write availability is better in a grid where the number of rows is much smaller than the number of columns, which means the grid should contain a large number of short columns. However, in a grid with a large number of columns, the size of read quorums is also large (recall that a read quorum is formed by obtaining one node from each column of the grid). Hence, read performance in such grids suffers. Also, having too many short columns makes it harder to collect one operational node from every column and hurts read availability. These tradeoffs make designing a grid especially hard.

3.2 Availability in Modified Protocol

We modified the grid protocol of [2, 3] slightly by redefining a read quorum in the following manner. A read quorum can be formed in one of two ways: as a set consisting of one node from each column of the grid *or* a set of all nodes from any single column. It is straightforward to show that any read quorum defined this way intersects with any write quorum and hence the quorum intersection rules are satisfied. This protocol will be referred to as the *modified grid protocol.* (This modification was also suggested independently by Neilsen [10].) This seemingly minor improvement to the protocol is significant because it increases the read availability without changing the write availability. Notice that now a grid with many short columns provides high write availability and, at the same time, good read availability and performance since a read operation can assemble all nodes from any one of many short columns to form a quorum. To calculate the read availability for the modified grid, note that the probability that a column is bad and alive is $1-p^m-q^m$, where m is the number of nodes in the column. Then, the probability that a read quorum *cannot* be formed is equal to the probability that *all* columns are *bad* minus the probability that all columns are *bad and alive*, which is:

$$\begin{gathered}(1-p^{m_1})^{n_1} \cdot \ldots \cdot (1-p^{m_k})^{n_k} - \\ (1-p^{m_1}-q^{m_1})^{n_1} \cdot \ldots \cdot (1-p^{m_k}-q^{m_k})^{n_k}\end{gathered}$$

Then, read availability RA is

$$\begin{gathered}1-((1-p^{m_1})^{n_1} \cdot \ldots \cdot (1-p^{m_k})^{n_k} - \\ (1-p^{m_1}-q^{m_1})^{n_1} \cdot \ldots \cdot (1-p^{m_k}-q^{m_k})^{n_k})\end{gathered} \tag{3}$$

It is easy to show that the set of quorums defined by the modified grid is optimal, i.e., it is not dominated by any other set.

The notion of *non-dominance* was first introduced in [5] to formalize the intuition behind optimal quorums. Let R and W be sets of read and write quorums. A pair of sets $Q=(R,W)$ (also called a *quorum agreement*) is *dominated* iff there exists another quorum agreement $Q'=(R',W')$ such that (1) $Q \neq Q'$ (i.e., $R \neq R'$ or $W \neq W'$); (2) for each $r \in R$, there exists $r' \in R'$ such that $r' \subseteq r$; and (3) for each $w \in W$, there exists $w' \in W'$ such that $w' \subseteq w$. Otherwise, Q is called *non-dominated.*

The notion of non-dominance characterizes the optimality of the quorums because if Q is dominated, then there exists another quorum agreement Q' whose quorums are subsets of quorums from Q. Then, the protocol based on Q' would have better availability and, possibly, performance properties, since it would be easier for an operation to collect a quorum of permissions.

Quorums defined by the modified grid can be described using the *disjoined set* method for constructing quorum agreements [10] if columns of the grid are chosen as the disjoined sets. Since any quorum agreement

grid dim	write unavail.	old read unavail.	new read unavail.	*old* / *new*
2 x 2	$5.23 \cdot 10^{-2}$	$1.99 \cdot 10^{-2}$	$3.70 \cdot 10^{-3}$	5
2 x 4	$4.05 \cdot 10^{-2}$	$3.94 \cdot 10^{-2}$	$2.53 \cdot 10^{-4}$	155
2 x 6	$5.86 \cdot 10^{-2}$	$5.85 \cdot 10^{-2}$	$1.30 \cdot 10^{-5}$	4489
4 x 2	$1.18 \cdot 10^{-1}$	$2.00 \cdot 10^{-4}$	$6.88 \cdot 10^{-5}$	2
4 x 4	$1.44 \cdot 10^{-2}$	$4.00 \cdot 10^{-4}$	$1.63 \cdot 10^{-5}$	24
4 x 6	$2.25 \cdot 10^{-3}$	$6.00 \cdot 10^{-4}$	$2.88 \cdot 10^{-6}$	207
6 x 2	$2.20 \cdot 10^{-1}$	$2.00 \cdot 10^{-6}$	$9.37 \cdot 10^{-7}$	2
6 x 4	$4.82 \cdot 10^{-2}$	$4.00 \cdot 10^{-6}$	$4.11 \cdot 10^{-7}$	9
6 x 6	$1.06 \cdot 10^{-2}$	$6.00 \cdot 10^{-6}$	$1.36 \cdot 10^{-7}$	44

Table 2: Unavailability of the existing and modified grid protocols with p = 0.90

grid dim	write unavail.	old read unavail.	new read unavail.	*old* / *new*
2 x 2	$1.40 \cdot 10^{-2}$	$4.99 \cdot 10^{-3}$	$4.81 \cdot 10^{-4}$	10
2 x 4	$1.00 \cdot 10^{-2}$	$9.96 \cdot 10^{-3}$	$8.92 \cdot 10^{-6}$	1117
2 x 6	$1.49 \cdot 10^{-2}$	$1.49 \cdot 10^{-2}$	$1.24 \cdot 10^{-7}$	120237
4 x 2	$3.44 \cdot 10^{-2}$	$1.25 \cdot 10^{-5}$	$2.32 \cdot 10^{-6}$	5
4 x 4	$1.21 \cdot 10^{-3}$	$2.50 \cdot 10^{-5}$	$1.60 \cdot 10^{-7}$	156
4 x 6	$7.82 \cdot 10^{-5}$	$3.75 \cdot 10^{-5}$	$8.23 \cdot 10^{-9}$	4553
6 x 2	$7.02 \cdot 10^{-2}$	$3.12 \cdot 10^{-8}$	$8.28 \cdot 10^{-9}$	3
6 x 4	$4.92 \cdot 10^{-3}$	$6.25 \cdot 10^{-8}$	$1.16 \cdot 10^{-9}$	53
6 x 6	$3.46 \cdot 10^{-4}$	$9.38 \cdot 10^{-8}$	$1.22 \cdot 10^{10}$	766

Table 3: Unavailability of the existing and modified grid protocols with p = 0.95

constructed using the disjoined sets method is non-dominated (Theorem 5.2 in [10]), so is the quorum agreement defined by the modified grid.

The described refinement of grid protocol can also be extended easily to the h-grid and hierarchical quorum consensus-2 (HQC2) [6, 7] protocols. The details are omitted here because the focus of the present work is on grids exclusively.

4 The existing vs. modified grid protocols

In this section, we give a comparison between the existing (*old*) grid protocol [2, 3] and our modified (*new*) protocol described above. For purposes of this comparison, we consider grids of the same dimensions as those considered in [3], and recompute the read availability for different values of p. Note that the write availability remains the same in both protocols. The results are shown in Tables 2, 3 and 4, which correspond to p=0.9, 0.95 and 0.99, respectively. For ease of comparison, we follow [3] in expressing fault-tolerance properties in the tables as *unavailabilities* (defined as 1 - availability). Each table gives the grid dimensions, write unavailability (same for *old* and *new* protocol), read unavailability of old protocol, read unavailability of new

grid dim	write unavail.	old read unavail.	new read unavail.	$\frac{old}{new}$
2 x 2	$5.92 \cdot 10^{-4}$	$2.00 \cdot 10^{-4}$	$3.97 \cdot 10^{-6}$	50
2 x 4	$4.00 \cdot 10^{-4}$	$4.00 \cdot 10^{-4}$	$3.13 \cdot 10^{-9}$	127835
2 x 6	$6.00 \cdot 10^{-4}$	$6.00 \cdot 10^{-4}$	$1.85 \cdot 10^{-12}$	324404608
4 x 2	$1.55 \cdot 10^{-3}$	$2.00 \cdot 10^{-8}$	$7.88 \cdot 10^{-10}$	25
4 x 4	$2.45 \cdot 10^{-6}$	$4.00 \cdot 10^{-8}$	$2.45 \cdot 10^{-12}$	16344
4 x 6	$6.37 \cdot 10^{-8}$	$6.00 \cdot 10^{-8}$	$5.66 \cdot 10^{-15}$	10596700
6 x 2	$3.42 \cdot 10^{-3}$	$2.00 \cdot 10^{-12}$	$1.17 \cdot 10^{-13}$	17
6 x 4	$1.17 \cdot 10^{-5}$	$4.00 \cdot 10^{-12}$	$7.77 \cdot 10^{-16}$	5146
6 x 6	$4.02 \cdot 10^{-8}$	$6.00 \cdot 10^{-12}$	0.00	—

Table 4: Unavailability of the existing and modified grid protocols with p = 0.99

protocol, and improvement ratio of read unavailability. The improvement ratio shown in the last column is computed as the read unavailability in the old protocol divided by the read unavailability in the new protocol, and is an indicator of the improvement that results from using the new protocol.

Several conclusions may be drawn from these results. First, in all cases, the read unavailability of the new protocol is significantly lower than in the old protocol. Secondly, the gap or relative difference between the two protocols as indicated by the improvement ratio increases for larger values of p. Thirdly, for all values of p, the relative improvement depends considerably on the grid dimensions, m and n. It is maximum when the number of rows (m) is small and number of columns (n) is large. This makes intuitive sense because the new protocol makes it easy to form a read quorum from all nodes in any one column (since m is small, i.e. each column is short). On the other hand, this is also the case where it is the hardest to form a read quorum in the old protocol by a column-cover since the number of columns is large and each column is short.

Fourthly, another important point that should be noted is that the modified protocol considerably lessens the tradeoffs between read and write availabilities mentioned in Section 3. This means that the same improvement in write availability now results from a much smaller sacrifice in read availability. In the old protocol, both read and write availability are very sensitive to actual grid dimensions m and n for a given total number of nodes N. To see this, consider for example, the 4×6 and 6×4 grids in Table 3. In the existing protocol, the write unavailability is two orders of magnitude lower for the 4×6 grid, while the read unavailability is three orders of magnitude lower for the 6×4 grid. On the other hand, for the modified protocol, the two orders of magnitude decline in the write unavailability for the 4×6 grid causes the read unavailability to increase (worsen) only by a factor of less than 5. Similar arguments apply to Tables 2 and 4 also. Hence, the new protocol, in addition to improving the read availability, also makes it less sensitive to the actual grid dimensions.

Grid Configuration	Read Avail.	Write Avail	Weighted Avail (80% reads)
1×16 solid	$1 - 10^{-16}$	0.185302	0.837060
2×8 solid	0.9999994	0.922746	0.984548
4×4 solid	0.999984	0.985629	0.997113
8×2 solid	0.999999989	0.675632	0.935126
16×1 solid	$1 - 10^{-16}$	0.185302	0.837060
3×5 solid (1 node not used)	0.999973	0.993575	0.998694
4×5 with 4 holes in the bottom row	0.999972	0.994079	0.998797

Table 5: Availability of various grids for $N = 16$ and $p = 0.9$.

Finally, given certain read and write availability requirements, it is possible to satisfy them with a grid of much fewer nodes using the new protocol as compared to the old one. Consider the case where a grid must be designed to the requirements that the read and write unavailabilities must not exceed 10^{-4} and 10^{-5} respectively, given $p = 0.95$. With the existing protocol, no solid grid with fewer than 35 nodes can simultaneously satisfy both these requirements. On the other hand, the new protocol allows these requirements to be met using only 24 nodes, say with a 4×6 grid.

5 Finding grids with the highest availability

This section describes an algorithm that, given N nodes, finds the *almost fully distributed* grid solution with the highest availability. This problem is complicated for two reasons: firstly, hollow grids often have a higher availability than solid grids; and, secondly, in some cases, a higher availability can be achieved if some of the nodes are not used at all. To illustrate these points, consider availabilities of various grids for $N = 16$ computed in Table 5. In this table, read, write and weighted availabilities have been computed. The weighted availability is defined as F·RA + (1-F)·WA, where F is the fraction of read operations. Here F is assumed to be 0.8. Table 5 shows that a solid 4×4 grid gives a *lower* write and weighted availability than a 4×5 grid with four holes in the bottom row. Moreover, a solid 3×5 grid containing only 15 nodes has higher values for both write and weighted availability than any solid grid containing 16 nodes. These improvements become more significant when viewed as decreases in *unavailability*, defined as $1-$ availability, rather than increases in availability.

Algorithm $OptimalWriteAvail$ for finding the grid with the highest write availability, given N nodes, is listed in Figure 2. Since the objective is to design an "almost fully distributed" protocol, our algorithm

```
OptimalWriteAvail(N): (best-avail, best-m, best-n);
    IF (N = 1)
        RETURN (p, 1, 1);
    ENDIF
    m = 1;   n = N;
    cur-best-m = m;   cur-best-n = n;
    cur-best-avail = avail(m, n, N);
    WHILE (n >= m)
        /* find next grid */
        n = n - 1;
        IF (m n < N)
            m = m + 1;
        ENDIF
        IF (avail(m, n, N) >= cur-best-avail)
            cur-best-avail = avail(m, n, N);
            cur-best-m = m;   cur-best-n = n;
        ENDIF
    ENDWHILE
    (prev-best-avail, prev-best-m, prev-best-n) =
        OptimalWriteAvail(N-1);
    IF (prev-best-avail > cur-best-avail)
        RETURN (prev-best-avail, prev-best-m, prev-best-n);
    ELSE
        RETURN (cur-best-avail, cur-best-m, cur-best-n);
    ENDIF
```

Figure 2: The algorithm for finding the grid with the highest write availability.

considers only the grids containing at most one hole in any column. Moreover, since the read and write availabilities are not a function of the exact location of the hole in a given column, it is assumed without loss of generality that the hole is always at the bottom of a column. For N nodes, the algorithm considers all rectangular grids such that: $m \leq n$ and $N \leq mn < N+n$. It does not make sense to consider grids with $mn \geq N+n$ because such a grid must have at least n holes, i.e. the bottom row contains all holes, and so it can be eliminated from consideration. It can also be shown that for any N, m, and n such that $mn < N+n$ and $m > n$, the $m \times n$ grid with all holes in the bottom row has lower availability than the $n \times m$ grid with all holes in the bottom row. Hence, in a search for the best grid, one may only consider grids where $m \leq n$.

Let G_N be the best (i.e. the highest availability) grid found among the grids containing exactly N nodes and WA_N be its write availability. (In an $m \times n$ grid containing N nodes, there are $n-(mn-N)$ columns with m nodes and $mn-N$ columns with $m-1$ nodes. Then, the write availability of this grid can be calculated using expression 1 of Section 3.) Then, the algorithm returns the grid G_N^{opt} as the one with the higher write availability among G_{N-1}^{opt} and G_N.

In considering N-node grids, the algorithm starts with $m=1$ and $n=N$, and then decrements n by 1

N	10	20	30	500	1000
Dim.	3×3	4×6	4×7	11×49	13×80
Write quorum size	5	9	10	59	92
Rel. write quorum size	55.6%	45%	35.7%	11.8%	9.2%

Table 6: Dimensions of grids with maximal write availability.

on every iteration until $n = \lfloor \sqrt{N} \rfloor$. Hence, the number of grids examined to find G_N is $O(N)$. Since a grid with $N-1$ nodes may have a higher availability than one with N nodes, our algorithm must also examine grids in which fewer than N nodes are used. Therefore, the total number of grids examined by the algorithm is $O(N^2)$. As a side result, the algorithm also gives the maximum availability for all grids with the number of nodes in the range $[1, N]$.

Figure 3 gives a plot of the write availability of the best grid and its relative quorum size, for various values of N up to 1000. The write availability is shown along the Y-axis as $(-\log_{10}(1 - WA))$ and can be interpreted as the number of 9's after the decimal place. The dimensions of the best grids and the write quorum sizes are given in Table 6. Figure 3 shows that as N increases, the maximum availability also increases, while at the same time the relative quorum size falls. This means that by increasing N, one can satisfy both a minimum availability requirement as well as a minimum load sharing requirement at the same time. This should be contrasted with the majority voting algorithm where the relative quorum size remains nearly 0.5 for all N.

Another point to be noted from Table 6 is that often the maximum write availability is achieved by not using all N nodes. For instance, in the case of $N = 30$, the best grid is 4×7 and so two nodes are not used. Of course, this could affect the degree of load sharing in a slightly adverse way. However, as we show next, the difference in availability can be very significant.

It was found that, in general, imposing a requirement that *all* N nodes be used can have a considerable negative impact on availability for some values of N, and no impact for other (sometimes neighboring) values of N. For example, for all values of N in Figure 3, the loss of availability is 0. On the other hand, for $N = 5$, the loss in write availability is large (when a 2×2 grid is compared with the best grid containing exactly five nodes). The relationship between loss of availability and N is plotted in Figure 4. Since for large N, write availability is very close to 1, it is more useful to consider the difference in *unavailability*, defined as

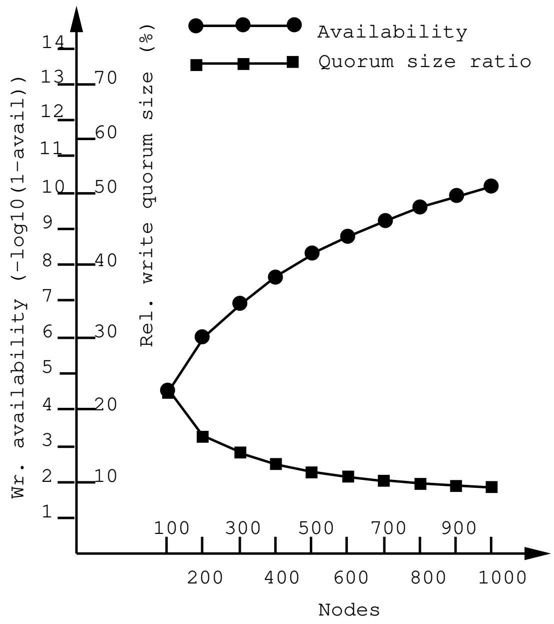

Figure 3: Maximal write availability for a given number of nodes.

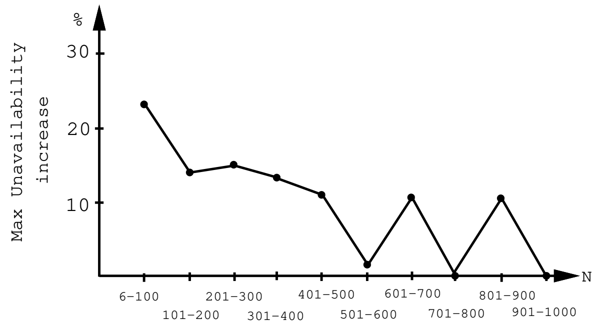

Figure 4: Increase of unavailability of almost fully distributed grids.

1−availability, rather than difference in availability. Moreover, in view of the above comment about sudden swings even for consecutive values of N, we decided to consider interval values of N, and for each interval, determine the *maximum* loss of availability. Figure 4 depicts the maximum increase of unavailability for the range of N from 6 to 1000 (for $N = 5$ the difference in unavailability is 125%). It shows that except for very small N, the increase in unavailability ranges from 0% to 17%. Hence, not only does a noticeable loss of availability occur when one requires that grids use all nodes, but this effect does not diminish even for large N.

Finally, we turn to consider weighted or combined availability. Assuming F is the fraction of read operations, the combined availability is computed as: $A = F \cdot RA + (1 - F) \cdot WA$, where RA and WA are read and write availabilities. Note that, although to compute the maximum write availability, only grids

N	10	20	30	500	1000
Dim. for $F = .8$	3×3	4×6	4×7	11×49	13×80
Dim. for $F = .99$	3×3	4×6	4×7	11×49	13×80
Dim. for $F = .999$	2×5	4×5	4×7	11×49	13×80

Table 7: Dimensions of grids with maximum combined availability and various read-write ratios.

where $m \leq n$ needed to be considered, imposing this condition does not always maximize read availability. Hence, all m, n combinations must be examined. Table 7 shows the grids providing the best combined availability for several values of N and F. It is interesting to note that the best grid configuration that maximizes combined availability is also the one which maximizes write availability alone, except for the cases where F is almost 1, i.e., nearly all operations are reads. This supports our assertion that the modified grid protocol produces a favorable change in the tradeoff between read and write availability; i.e., it is now possible to obtain a gain in write availability at the expense of a much smaller loss in read availability than with the original grid protocol.

6 Asymptotic vs. initial behavior of grids

In this section we discuss the write availability of grids. It was shown in [7] that in square grids, write availability goes to 0 when N goes to infinity. Here, in Section 6.1, we study the asympotic write availability for *non-square* grids. We show that asymptotically high WA is possible for non-square grids, and characterize precisely the grid dimensions under which this is achieved. While the theoretical guarantee of high WA for non-square grids is very useful, the asymptotic result from Section 6.1 is somewhat pessimistic in terms of quorum sizes, i.e., the quorum sizes become almost linear in N. Therefore, in Section 6.2, we also examine empirically the initial behavior of WA for grids with smaller relative quorum sizes. This analysis shows that while such grids have asymptotically low WA, for practical values of p, their WA initially grows towards 1, and starts to decline only for extremely large values of N ($N = 23000$, when $p = 0.99$). Thus, these grids can often be useful in practice despite their low asymptotic WA.

6.1 Asymptotic behavior

Consider a rectangular grid arrangement of N nodes into m rows and n columns where $m = f(N)$, $n = g(N)$ and $mn \geq N$ (the last condition says that all N nodes are used in the grid). We are interested in finding $f(N)$ and $g(N)$ such that WA increases asymptotically towards 1 in the limit when $N \to \infty$. Here we prove that the asymptotic value of WA is 1 in a small range near $n = \frac{N}{\log N}$.

Theorem 2. In a rectangular grid arrangement of N nodes (each assumed to be up with probability p and down with probability $q = 1 - p$) into m rows and n columns ($mn \geq N$), there are constants $c_1 = 2\log(1/p)$ and $c_2 = \log(1/q)$, such that:

(1) the write availability WA of this grid asymptotically approaches 1 if $c_1 \frac{N}{\log N} \leq n \leq c_2 \frac{N}{\log N}$.

(2) the write availability WA asymptotically approaches 0 if $n \leq \frac{c_1}{2} \frac{N}{\log N}$ or $n \geq 2c_2 \frac{N}{\log N}$ [1].

Proof We first define two events (see Section 2 for terminology) as follows: $E1 \equiv$ All columns are alive; $E2 \equiv$ At least one column is good. Recall, from Section 3, that the write availability WA $=$ Prob$(E1 \wedge E2)$.

The theorem follows from the following claims. (The claims will be proved shortly.)

1. if $n \leq c_2 \frac{N}{\log N}$, then Prob$(E1) \to 1$
2. if $n \geq c_1 \frac{N}{\log N}$, then Prob$(E2) \to 1$
3. if $n \geq 2c_2 \frac{N}{\log N}$, then Prob$(E1) \to 0$
4. if $n \leq \frac{c_1}{2} \frac{N}{\log N}$, then Prob$(E2) \to 0$

Given these claims, the two parts of the theorem are proved separately below.

(Part 1)

We derive a lower bound on WA in the following manner.

WA $=$ Prob$(E1 \wedge E2) =$ Prob$(E1)+$ Prob$(E2)-$Prob$(E1 \cup E2)$.

Since Prob$(E1 \cup E2) \leq 1$, we get WA $\geq$ Prob$(E1)+$ Prob$(E2) - 1$.

If $c_1 \frac{N}{\log N} \leq n \leq c_2 \frac{N}{\log N}$ then from Claims 1 and 2 we conclude that both Prob$(E1)$ and Prob$(E2)$ approach 1, which in turn implies that WA approaches 1.

(Part 2)

If $n \geq 2c_2 \frac{N}{\log N}$ then from Claim 3 we have Prob$(E1) \to 0$.

Similarly if $n \leq \frac{c_1}{2} \frac{N}{\log N}$ then from Claim 4 we have Prob$(E2) \to 0$.

Since WA $\leq$ minimum(Prob(E1), Prob(E2)), either of these implies that WA approaches 0. $\square$

---

[1] Note: c_1 and c_2 depend upon p. For $p = 0.99, c_1 = 0.0288$ and $c_2 = 6.64$.

Now we shall prove the four claims made above.

Proof (of Claims 1-4) In the introduction of algorithm OptimalWriteAvail, it was explained that we restrict our attention to *general* grids with at most one hole per column. To keep the exposition simple, we will use the expressions for computing various probabilities for solid grids, i.e., ones with no holes. It can be verified that this does not affect the asymptotic analysis.

As shown in Section 3, $\text{Prob}(E1) = (1 - q^{\frac{N}{n}})^n$ and $\text{Prob}(E2) = 1 - (1 - p^{\frac{N}{n}})^n$. Using the fact that $(1 - X)^Y = (1 - X)^{\frac{1}{X} \cdot XY} = [(1 - X)^{\frac{1}{X}}]^{XY}$ approaches $(\frac{1}{e})^{XY}$, we get that $\text{Prob}(E1) \rightarrow (\frac{1}{e})^{nq^{N/n}}$ and $\text{Prob}(E2) \rightarrow 1 - (\frac{1}{e})^{np^{N/n}}$. Below, we will prove claims 1 and 2. The proofs of claims 3 and 4 are exactly similar, and are therefore omitted.

(claim 1)

Suppose $n \leq c_2 \frac{N}{\log N}$. Substituting value of c_2 we get $n \leq \frac{N}{\log N} \log(1/q)$. So $(N/n) \log(1/q) \geq \log N$. $nq^{\frac{N}{n}} = \frac{n}{(1/q)^{\frac{N}{n}}} = \frac{n}{2^{\frac{N}{n}\log(1/q)}} \leq \frac{n}{2^{\log N}} = \frac{n}{N} \leq \frac{c_2}{\log N} \rightarrow 0$. $\text{Prob}(E1) \rightarrow (\frac{1}{e})^{nq^{\frac{N}{n}}} \rightarrow 1$.

(claim 2)

Now if $n \geq c_1 \frac{N}{\log N}$ then substituting value of c_1 we get $n \geq \frac{2N}{\log N} \log(1/p)$. So $(N/n) \log(1/p) \leq \frac{\log N}{2}$. $np^{\frac{N}{n}} = \frac{n}{(1/p)^{\frac{N}{n}}} = \frac{n}{2^{\frac{N}{n}\log(1/p)}} \geq \frac{n}{2^{\frac{\log N}{2}}} = \frac{n}{\sqrt{N}} \geq \frac{c_1\sqrt{N}}{\log N} \rightarrow \infty$. 1− $\text{Prob}(E2) \rightarrow (\frac{1}{e})^{np^{\frac{N}{n}}} \rightarrow 0$, which implies that $\text{Prob}(E2) \rightarrow 1$. □

In summary, we have exactly characterized the conditions under which the write availability WA of a rectangular grid will asymptotically approach 1. The dimensions of the grid must be $m = \Theta(\log N)$ and $n = \Theta(\frac{N}{\log N})$; the resulting write quorum size is $\Theta(\frac{N}{\log N})$, which is very close to being linear in N.

6.2 Initial behavior

The result stated above implies that for the write availability to asymptotically approach 1, the write quorum size must be close to $N/\log N$ which is almost linear in N. This is undesirable because it produces large quorum sizes and a low degree of load sharing. Therefore, we studied in more detail the empirical behavior of write availability as a function of N for larger values of m than $\log N$, say N^t for $0 < t \leq 0.5$. The objective was to find out how WA actually varies as N increases. For instance, does it drop quickly to 0 as N increases, or does it fall gradually, or is their some other behavior? Another purpose was to study the

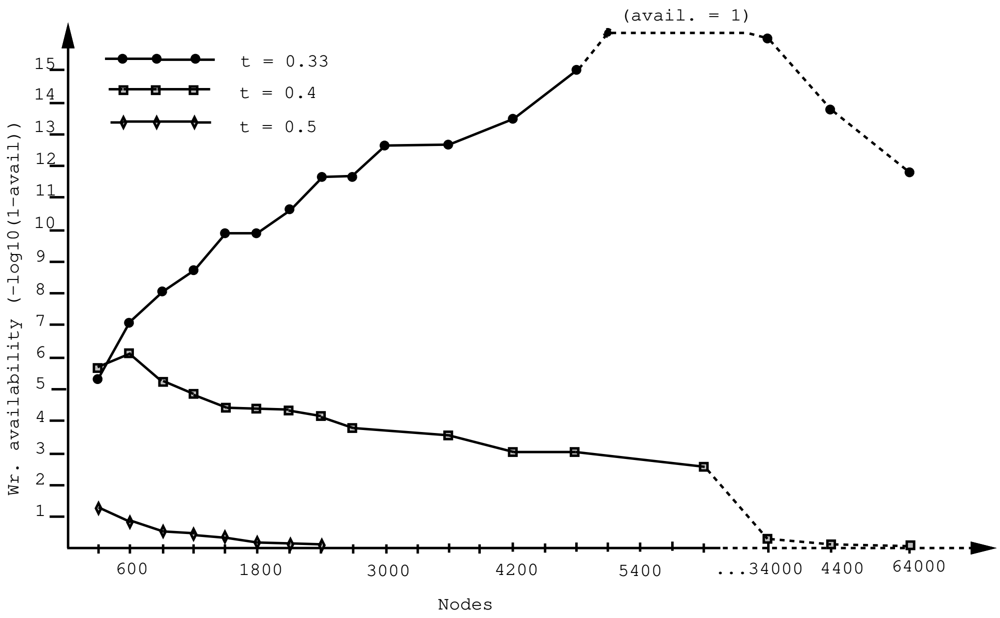

Figure 5: Write availability of $m \times n$ grids, for $m = \lfloor N^t \rfloor$ and $p = 0.9$.

impact of various values of t and p on this behavior.

Figures 5 and 6 plot the write availability against N for $m \times n$ grids such that $m = \lfloor N^t \rfloor$ and $n = \lceil N/m \rceil$, for several values of t and p. The write quorum size $Q_W = m + n - 1 = O(N^{1-t})$. Notice that when $p = 0.9$ and $t = 0.33$, $Q_W = O(N^{0.67})$, and WA continues to grow until $N = 23000$. In fact, for $p = 0.99$, even WA of a square grid ($t = 0.5$, $Q_W = O(N^{0.5})$) grows until $N = 23000$ nodes. These figures show that for sufficiently large p, the write availability of grids with small write quorum sizes grows with N, approaches 1, and begins to fall only when N becomes very large ($N > 23000$). This means that, in spite of the negative theoretical result from the previous subsection, rectangular grids are still very promising both from availability and load sharing points of view in most practical situations.

7 Availability and load sharing

In this section, we turn to the problem of satisfying both a load sharing objective and a data availability objective simultaneously. The algorithm for solving this problem is given in Figure 7. The goal is to find the smallest N and a corresponding $m \times n$ grid that satisfies both objectives. It begins with small N and calculates write availability and relative quorum size of all grids containing all N nodes. As before, only hollow grids with no more than one hole in any column are considered. For a given N, the algorithm examines grids in increasing order of write quorum size and stops as soon as it finds a grid satisfying both quorum

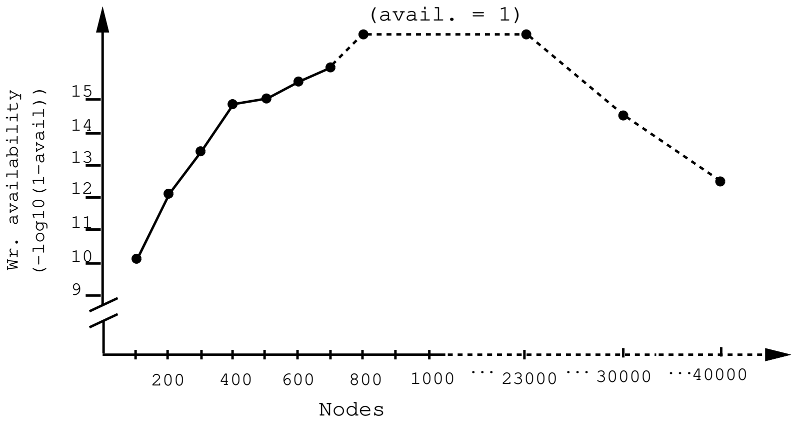

Figure 6: Write availability of square grids ($m = \lfloor N^{0.5} \rfloor$) and $p = 0.99$.

```
FindingGrid(rel-wr-quorum-size-cutoff, wr-avail-cutoff): (m, n)
/* Find a grid such that its relative write quorum size does
not exceed rel-wr-quorum-size-cutoff, its write availability
is not lower than wr-avail-cutoff, and N is as small as possible. */
   N = 4; /* (No load sharing is possible for N < 4) */
   DO-FOREVER
      /* Start with the grid that has the smallest quorum size
      for a given N */
      m = ⌊√N⌋;   n = ⌈√N⌉;
      IF (mn < N)
         m = m+1;
      ENDIF
      cur-wr-avail = wr-avail(m, n, N);
      WHILE ( m > 1 AND cur-wr-avail < wr-avail-cutoff)
         /* find next grid */
         n = n+1;
         WHILE (mn > N+n)
            m = m-1;
         ENDWHILE
         cur-wr-avail = wr-avail(m, n, N);
      ENDWHILE
      IF (cur-wr-avail >= wr-avail-cutoff AND
         rel-wr-quorum-size <= rel-wr-quorum-size-cutoff)
         RETURN(m, n);
      ELSE
         N = N + 1;
      ENDIF
   ENDDO
END
```

Figure 7: The algorithm for finding the grid with given degree of load sharing and availability.

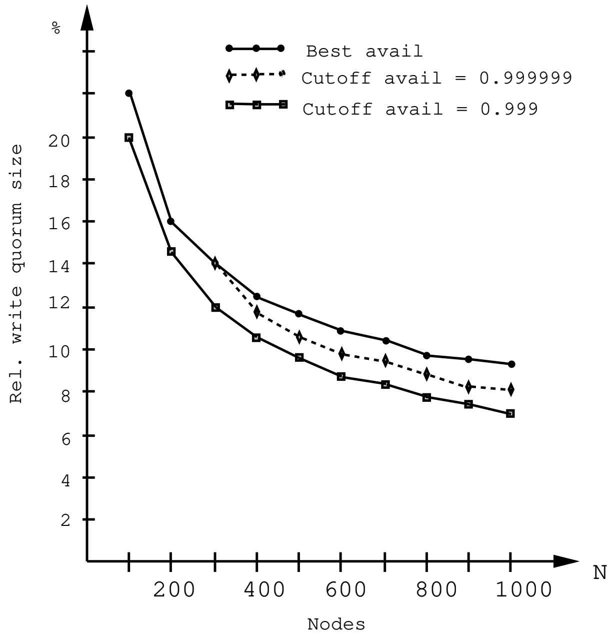

Figure 8: Relative write quorum size of grids with a given write availability ($p = 0.9$).

size and availability requirements. Since, as seen in Section 5.1, grids can provide asymptotically high write availability and low relative quorum size at the same time, this algorithm will always terminate.

In Figure 8, the minimal relative write quorum size satisfying a cutoff write availability is plotted against N for several cutoff values. Also shown for comparison purposes is the relative write quorum size of the grids giving the *best* achievable write availability for a given N (the cutoff does not apply here).

Although the asymptotic theoretical result suggests that the relative write quorum size would drop at a very slow, logarithmic rate, the actual decline is much steeper for values of N used in practice. However, in view of the discussion in section 5.2, this result is not surprising. Another point is that the spread of relative quorum sizes provided by grids with different availability requirements is rather narrow. For example, when $N = 500$, and $WA = 0.999$, the relative write quorum size is 9.6%. On the other hand, for the same N, and the maximum possible write availability (0.99999999 in this case) the relative write quorum size is 11.8%. Clearly, in this example, choosing the maximum availability grid produces a large improvement in availability at only a small loss in the degree of load sharing. However, the other point to consider is the absolute value of quorum sizes. For $N = 500$, the maximum availability grid has the dimensions 11×49 and a write quorum of 59 nodes, while the grid with 0.999 availability has the dimensions 16×33 and a write quorum of 48 nodes. Therefore, although the difference in the degree of load sharing between these two grids is small (only 2.2%), the difference in absolute quorum size is more than 20% and this will translate into a large performance impact. Moreover, Figure 8 shows that a write availability threshold of 0.999, and a load sharing threshold of 11.8%, can be achieved by a grid with only 300 nodes. Therefore, it is not a good idea to always select a grid with the highest availability for a given N, and, hence, the algorithm of this section is useful.

A very similar algorithm to the one shown here can be proposed to deal with the case where thresholds for weighted availability and weighted relative quorum size are stated, assuming that the read-ratio F is known. The only difference is that, as explained earlier, in this case all grids containing N nodes must be considered (not just grids where $m \leq n$). For a given N, the algorithm would identify the grids with availability greater than the threshold and choose the grid with the smallest quorum size among them. The algorithm would stop if this grid satisfies the quorum size requirement. Otherwise, the number of nodes N would be incremented and the process repeated.

8 Conclusion

In this paper, we first proposed a modified grid protocol which dominates the existing grid protocol, and studied various performance aspects of both solid and hollow grids. We showed that the modified protocol is optimal, and also compared its performance with the existing protocol in terms of availability. Algorithms for maximizing availability in an unconstrained manner, and also for designing grids subject to both availability and load sharing constraints were given. We have shown that general grids (which allow some empty positions or holes) are useful to consider because often general grids produce a higher availability than solid grids where all positions must correspond to nodes. We also studied the asymptotic behavior of grids and showed that for the write availability to increase asymptotically, the dimensions of a N-node grid must be approximately $\log N \times N/\log N$. On the other hand, an empirical study showed that the asymptotic behavior is significant only for very large N. Therefore, for most practical values of N, the grid dimensions can be $N^t \times N^{1-t}$, where $t > 0.33$. It was found that the exact value of t depends upon p, the probability that any node in the grid is up.

References

[1] D. Agrawal and A. El Abbadi. An efficient solution to the distributed mutual exclusion problem. In *Proc. of Symp. on Principles of Distributed Computing*, pp. 193-200, 1989.

[2] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The Grid protocol: a high performance scheme for maintaining replicated data. In *Proc. of the IEEE 6th Int. Conf. on Data Engineering*, pp. 438-445, 1990.

[3] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The Grid protocol: a high performance scheme for maintaining replicated data. *IEEE Trans. on Knowledge and Data Eng.*, 6(4) pp. 582-592, December 1992.

[4] D.K. Gifford. Weighted voting for replicated data. In *Proc. of the Seventh ACM Symposium on Operating Systems Principles*, pp. 150-159, December 1979.

[5] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *J. of the ACM,* 32(4), pp. 841-860, October 1985.

[6] A. Kumar and K. Malik. Generalizing and optimizing hierarchical quorum consensus algorithms for replicated data. Graduate School of Management, Cornell University, Tech. Report, October 1991.

[7] A. Kumar and S. Y. Cheung. A $\sqrt{N}$ high availability hierarchical grid algorithm for replicated data. *Information Processing Letters* 40(1991), pp. 311-316.

[8] A. Kumar, M. Rabinovich and R. Sinha. A Performance Study of a New Grid Protocol and General Grid Structures for Replicated Data, Tech. Report 93-03-02, Department of Computer Science and Engineering, University of Washington, March 1993.

[9] M. L. Neilsen and M. Mizuno. Decentralized consensus protocols. In *Proc. of the 10th Int. Phoenix Conf. on Comp. and Comm.*, pp. 257-262, 1991.

[10] M. L. Neilsen. *Quorum structures in distributed systems*, PhD thesis, Kansas State University, May 1992.

[11] M. Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. on Comp. Systems.* 3(2), May 1985.

[12] S. Rangarajan, S. Setia and S. Tripathi. Fault tolerant algorithms for replicated data management. In *Proceedings of 8th International Conference on Data Engineering*, IEEE, 1992.

[13] D. Skeen. A quorum-based commit protocol. In *Proc. of the 6th Berkeley Workshop on Distr. Data Managementt and Comp. Networks*, pp. 69-80, 1982.

[14] Thomas, R. H., "A majority consensus approach to concurrency control," *ACM Trans. on Database Systems* 4(2), pp. 180-209, June 1979.