Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures

Nabor das Chagas Mendonça
and
Ricardo de Oliveira Anido

Relatório Técnico DCC–15/93

Junho de 1993

Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures

Nabor das Chagas Mendonça
and
Ricardo de Oliveira Anido

Abstract

In large distributed computing systems, where copies of the same logical data are stored at many different sites, the replica control protocol must reduce communication costs when forming the quorums required at each access to the replicated data, in order to improve the system response time. An interesting way to achieve this reduction is to organize the copies into some logical structure, like a grid or a tree, and then to use this structure to form smaller quorums. Another example of a structure used to form smaller quorums is a generic tree in which only the leaves correspond to copies of the replicated data.

This paper presents a new replica control protocol that logically organizes the copies as leaves of a generic tree, but introduces the *blind-write* as another operation (besides the traditional read and write) defined over the replicated data. With this third operation, the proposed protocol turns out to be a generalization of the traditional voting scheme and others existing protocols that use a symmetrical logical structure (i.e., a structure in which the responsibility for controlling the replicated data is equally shared by all copies) to form the access quorums. The proposed protocol also makes possible to achieve better relations between quorums size and the total number of copies, even under high availability requirements.

1 Introduction

Data replication is a common technique used to improve reliability and performance in distributed systems. Keeping copies of the same logical data at different sites increases data availability (by making the data accessible regardless of failures in the system) and decreases system response time (by allowing a copy to be accessed locally or from a near site, and by sharing load among the sites that store a copy of the replicated data). The disadvantage, however, is the need for complex and expensive protocols to maintain the copies in a consistent state.

A large number of replica control protocols exists in the literature. Most of them follows the strategy of only allowing a read or write operation to be performed on replicated data if permissions from certain sets of copies are previously obtained. Such sets, called *quorums*, must be defined in such a way that every two conflicting operations (any pair of write/write or read/write operations) have at least one copy in common. Hence, it is guaranteed that a read operation will always get the most recent copy, and that no two write operations will occur independently thus excluding the possibility that copies diverge.

Voting [20, 12] is a well known mechanism to define quorums. In the simple voting protocol [20], a write quorum is formed by a majority of copies of the replicated data (with one vote per copy); in the weighted voting protocol [12], each copy is assigned a different number of votes, and a write quorum is formed by any group of copies whose votes represent a majority of the total votes assigned. In both protocols, read and write quorums must be set in a way that the sum of their votes is greater than the total votes in the system. It is easy to see that these restrictions guarantee the intersection between quorums of conflicting operations. Many variants of the voting mechanism have been proposed, for instance: dynamic voting [17, 8, 13, 19], voting with witnesses [16], voting based on virtual partitions [1] and voting with fragmentation [3].

Since a site needing to perform an operation on the replicated data has to communicate with all the copies of some quorum, system performance can be considerably penalized. To cope with this problem,

quorums size must be small, compared to the total number of copies. In large distributed systems, where replicated data may have a large number of copies[1], traditional voting protocols are not adequate since the number of copies required to form their quorums increases linearly with the total number of copies. Theoretically, smaller quorums can be obtained by assigning a different number of votes to different copies [7]; this solution, however, is not suitable to a system with a large number of copies because it would impose an undesirable centralized control.

A better way to reduce quorum size is to organize the copies into some logical structure and then to derive the quorums from this structure [2, 10, 14, 15, 4]. Quorums formed in this way in general have sublinear size and do not have any equivalent vote assignment in the weighted voting protocol [12]. Cheung, Ammar and Ahamad [10], for example, propose to organize the N copies of the replicated data into a rectangular grid; a read quorum is formed by at least one copy from all columns of the grid, and a write quorum is formed by all the copies from at least one column, plus the copies of a read quorum. When the number of rows equals the number of columns, read and write quorums size are $\sqrt{N}$ and $2\sqrt{N} - 1$, respectively. Agrawal and El Abbadi [4] use a tree structure to organize the copies and obtain quorums size varying from $\log N$ to $N/2 + 1$ copies. Kumar [14] proposes a generalization of the voting mechanism, called *hierarchical quorum consensus* (HQC), where a multilevel hierarchy of vertices organized as a tree is used. Copies of the replicated data correspond to the leaf vertices at the lowest level of the hierarchy, and vertices at higher levels are logical groups of vertices at the level immediately below. Access permissions from the copies are propagated to the top of the hierarchy as follows: a vertex at level i will give access permission if it receives access permission from at least a determined number (quorum of level i) of its children (vertices at level $i - 1$). In this way, an operation will be performed on the replicated data only if access permission is obtained from the root vertex of the

[1] Although it is not conceivable nowadays the synchronization of more than a few copies (more than 10, for example), this number is likely to grow in the future due to the evolution of distributed systems.

hierarchy. Read and write quorums of each level are defined under the same restrictions present in the voting mechanism. It is shown in [14] that the best case occurs when the hierarchy is a complete ternary tree, producing quorums of size $2^{\log_3 N}$ ($\approx N^{0.63}$).

In this work, we extend Kumar's HQC protocol by presenting another way to define quorums at each level of the hierarchy and allowing a third operation (besides the traditional read and write) to be performed on the replicated data: the *blind-write* operation. Read and blind-write quorums are defined for each level following only the restriction that they must have at least one vertex of the level below in common. Permission to perform a read or blind-write operation on the replicated data is obtained similarly to the HQC protocol; permission to perform a write operation, however, is obtained by combining read and blind-write access permissions. This combination assures that quorums of conflicting operations will still intersect, and makes possible to achieve, compared to the original HQC protocol, better relations between quorums size and the total number of copies. Moreover, the new resulting protocol, called *extended hierarchical quorum consensus* (HQC+) protocol, is also a generalization of others protocols that use a grid logical structure to form quorums [10, 2, 15], increasing flexibility to define quorums and, in many situations, data availability.

In the next section we define the system model adopted. Section 3 describes the HQC+ protocol and shows that it is correct. An analysis of quorums size and data availability is presented in section 4. Section 5 compares the HQC+ protocol to other related work and section 6 shows some practical examples of its superiority. We conclude the paper in section 7 summarizing our main results.

2 Model

A distributed system consists of a collection of independent processors which communicate only by exchanging messages through a communication system. Each processor has its own local memory, and is also referred as a *node*. A node may become inaccessible due to its own fai-

lure, to other nodes failures, or to failures in the communication system. We consider that both nodes and communication system components are *fail-stop*, i.e., they do not behave maliciously, but rather become inoperative in the presence of failures. Failures may lead to network partitioning, where the system is divided in groups of nodes called partitions [11]. Nodes within the same partition can communicate, whereas nodes in different partitions become inaccessible to each other.

Data is replicated by storing copies of the same logical data item at different nodes. Three operations, read, write and blind-write[2], are allowed on replicated data. Before performing an operation a node must obtain permission from a number of copies (quorum) according to a control protocol. Each copy is assigned a version number which is incremented every time the copy is updated, thus allowing a node to identify the most recent copy as the one with the highest version number.

In relation to protocol correctness we will adopt the *one-copy serializability* criterion [9], which states that, in order to maintain the consistency between copies of a replicated object, quorums of conflicting operations must have at least one copy in common. In the HQC+ protocol, two operations conflict if one of them is a write, or one is a read and the other is a blind-write. Notice that, under these rules, two read operations (or two blind-write operations) can proceed independently from each other, without affecting consistency.

3 The HQC+ Protocol

In this section, we describe the logical structure used by the HQC+ protocol and show how this structure is imposed on the set of copies of the replicated data — which is done in a similar way to the original HQC protocol. Then we show how the new HQC+ protocol derives quorums from this structure.

[2]During a blind-write operation, copies are modified regardless of their previous values; such situation occurs, for instance, in initializations. This distinction is made because the number of copies required to form a blind-write quorum is less than that required to form a traditional write quorum.

3.1 The Hierarchical Structure

The HQC+ protocol uses a logical m-level hierarchy of vertices organized as a tree to form the access quorums to the replicated data. Vertices at level 0 (or leaves of the tree) correspond to physical copies, and vertices at higher levels represent logical groups of vertices at the level immediately below. In general, a vertex at level i ($1 \leq i \leq m$) of the hierarchy represents a logical group of l_i vertices at level $i-1$. Hence, the vertex at the highest level (or the root of the tree) represents a logical group of l_m vertices at level $m-1$, and each vertex at level $m-1$, in turn, represents a logical group of l_{m-1} vertices at level $m-2$, and so on. The total number of copies is given by $\prod_{i=1}^{m} l_i$. Figure 1 shows an example of a 2-level hierarchy ($l_1 = l_2 = 3$) with 9 copies at level 0.

A read (blind-write) quorum is defined for each level i of the hierarchy. These quorums indicate the minimum number of children at level $i-1$ from which read (blind-write) access permissions must be obtained by a vertex at level i before it can give permission to its parent at level i+1. Read and blind-write quorums at level i of the hierarchy are denoted by r_i e bw_i, respectively. To guarantee intersection between read and blind-write quorums, r_i and bw_i must be chosen under the restriction $r_i + bw_i > l_i$. In order to minimize r_i and bw_i, this restriction is rewritten as $r_i + bw_i = l_i + 1$, thus eliminating the need that either r_i and bw_i be explicitly defined — it suffices the definition of r_i to infer bw_i ($bw_i = l_i - r_i + 1$). Notice that there is no need for the restriction $2bw_i > l_i$ since two blind-write operations do not conflict; this fact makes possible to eliminate restriction $r_i < l_i/2 + 1$, present in the original HQC protocol. To make notation clearer, values l_i and r_i ($1 \leq i \leq m$) will be expressed as vectors $l = (l_1, l_2, \ldots, l_m)$ and $r = (r_1, r_2, \ldots, r_m)$.

Each physical copy of the replicated data is assigned an m-subscript identifier of the form $o_{i1\ i2\ \ldots\ im}$, where $1 \leq i_j \leq l_j$ and $1 \leq j \leq m$. For example, in the 2-level hierarchy shown in figure 1, physical copies are identified as o_{11}, o_{12}, o_{13}, o_{21}, o_{22}, o_{23}, o_{31}, o_{32} and o_{33}. In the same way, in a 3-level hierarchy with 27 copies at level 0, physical copies would be identified as o_{ijk}, where $1 \leq i, j, k \leq 3$.

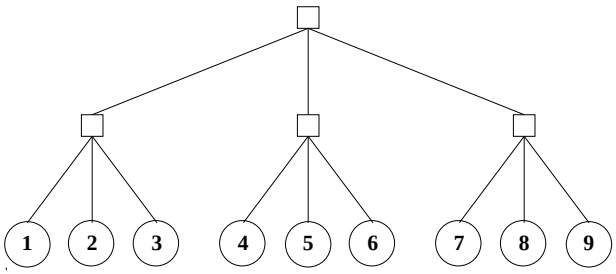

Figure 1: 9 copies organized into a 2-level hierarchy.

3.2 Quorum Formation

A read, blind-write or write operation can be performed on the replicated data, according to the HQC+ protocol, if the respective permission is obtained from the vertex at the highest level of the hierarchy (the root of the tree). The root will give read access permission if it receives these same permissions from at least r_m of its children at level $m - 1$. Each root's child at level $i - 1$, in turn, must obtain read access permissions from at least r_{m-1} of its children at level $m - 2$, and so on, until permissions be requested directly from physical copies at level 0. Blind-write access permissions are obtained in an analogous way. If we make $r = (2, 2)$ in the 2-level hierarchy shown in figure 1, the following sets of copies are examples of either read and blind-write quorums: {1, 2, 4, 5}, {2, 3, 7, 8} and {5, 6, 7, 9}. Similarly, if we make $r = (1, 2)$, examples of read quorums are {1, 4}, {6, 7} and {2, 8}, and examples of blind-write quorums are $\{1, 2, 3, 4, 5, 6\}$, $\{1, 2, 3, 7, 8, 9\}$ and $\{4, 5, 6, 7, 8, 9\}$. The algorithms to form read and blind-write quorums are shown in figure 2 (the only difference between them is the recursive calling to level $i - 1$). They are called by passing m and r_m as input parameters, and will return SUCCESS, if a quorum is successfully formed, or FAILURE, if a quorum could not be obtained.

Write access permission in the HQC+ protocol is obtained by com-

```
FormQuorum_r(i,size)
{
    num_examined = num_locked = 0;
    if (i > 0)
        while (num_locked < size) && (num_examined < l_i){
            num_locked += FormQuorum_r(i - 1,r_{i-1});
            num_examined++;
        }
    else
        while (num_locked < size) && (num_examined < l_i){
            if (copy o[i[1],i[2],...,i[m]] is unlocked){
                lock copy o[i[1],i[2],...,i[m]];
                num_locked++;
            }
            num_examined++;
        }
    if (num_locked < size)
        return(FAILURE);
    else
        return(SUCCESS);
}
```

```
FormQuorum_bw(i,size)
{
            .
            .
            .
            num_locked += FormQuorum_bw(i - 1,l_{i-1} - r_{i-1} + 1);
            .
            .
            .
}
```

Figure 2: Algorithms to form read and blind-write quorums.

bining read and blind-write access permissions; more precisely, a write quorum is formed from the union of a read and a blind-write quorum. This union guarantees the exclusive access to the replicated data required by write operations[3] (see correctness proof in next subsection). Combination of read and blind-write quorums is done in the following way: a vertex at level i of the hierarchy will give write access permission to its parent at level $i+1$ if, from its children at level $i-1$, $min(r_i, bw_i)$ give write access permissions and $|bw_i - r_i|$ give access permissions to the operation with the greatest quorum at level i (which will be a read, if $r_i < bw_i$, or a blind-write, on the contrary). For instance, let us assume that $l_m = 6$, $r_m = 1$ and $bw_m = 6$ for some hierarchy. In this case, a write operation can be performed if, among the vertices at level $m-1$, one gives write access permission and the remaining five give blind-write access permissions. Similarly, if we make $r_m = 5$ e $bw_m = 2$, it would be necessary write access permissions from two vertices at level $m-1$, plus read access permissions from other three vertices at this same level. Notice that the total number of vertices from which access permission must be obtained is always $max(r_i, bw_i)$; when $r_i = bw_i$, only write access permissions are requested. If we make $r = (1,3)$ in the hierarchy shown in figure 1, examples of write quorums are {1, 2, 3, 4, 7}, {1, 4, 5, 6, 9} e {2, 5, 7, 8, 9}. With this configuration, read and blind-write quorums have size three and write quorums have size five — the 2-level hierarchy, in this case, generates the same quorums generated in a 3×3 grid structure in the grid protocol [10][4]. The algorithm to form write quorums is shown in figure 3. It is called in a way similar to the two algorithms previously described.

[3] The idea of forming a write quorum by uniting a read quorum and its complement (or a blind-write quorum) was previously used in [6] and [15].

[4] The case in which the HQC+ protocol corresponds to the grid and others protocols is analyzed in section 5.

```
FormQuorum_w(i,size)
{
    num_examined = num_locked = 0;
    num_w = min(size,l_i-size+1);
    num_tot = max(size,l_i-size+1);
    if (i > 0)
        while (num_locked < size) && (num_examined < l_i){
            num_locked += FormQuorum_w(i - 1,r_{i-1});
            num_examined++;
        }
        if (num_locked = num_w)
            while (num_locked < num_tot) && (num_examined < l_i){
                if (size <= l_i/2)
                    num_locked += FormQuorum_bw(i - 1,r_{i-1});
                else
                    num_locked += FormQuorum_r(i - 1,r_{i-1});
                num_examined++;
            }
    else
        while (num_locked < num_tot) && (num_examined < l_i){
            if (copy o[i[1],i[2],...,i[m]] is unlocked){
                lock copy o[i[1],i[2],...,i[m]];
                num_locked++;
            }
            num_examined++;
        }
    if (num_locked < num_tot)
        return(FAILURE);
    else
        return(SUCCESS);
}
```

Figure 3: Algorithm to form write quorums.

3.3 Proof of Correctness

As mentioned in section 2, in order to prove that the HQC+ protocol is correct it is sufficient to show that it implements *one-copy serializability*. In other words, we must show that all quorums of conflicting operations formed in the m-level hierarchy have at least one copy of the replicated data in common. We do that by proving the following lemmas.

Lemma 1 *In a m-level hierarchy, the HQC+ protocol guarantees that there is at least one copy at level 0 common to any read and blind-write quorum.*

Proof. By induction on the levels of the hierarchy.
Basis. There is at least one vertex at level $m-1$ common to any read and blind-write quorum formed at level m. This statement is easily verified since read and blind-write quorums are formed at level m by r_m and bw_m vertices at level $m-1$, respectively, and $r_m + bw_m > l_m$.
Hypothesis. There is at least one vertex at level $i-1$ common to any read and blind-write quorum formed from the children of a vertex at level i ($1 \leq i \leq m-1$).
Inductive step. We must show that if there is a vertex at level $i-1$ (among the children of a vertex at level i) common to a read and a blind-write quorum, there is also a vertex at level $1-2$ (among the children of a vertex at level $i-1$) common to these two quorums. This step will guarantee that an intersection between read and blind-write quorums formed at level m, which is the basis of the proof, implies an intersection between them at level 0, where vertices correspond to physical copies.
If there is child of a vertex at level i common to a read and a blind-write quorum, this child is obviously a vertex at level $i-1$. This vertex corresponds to a logical group of l_{i-1} vertices at level $i-2$ in which a read and a blind-write quorum were also formed; as $r_{i-1} + bw_{i-1} > l_{i-1}$, it is guaranteed that there is at least one vertex at level $i-2$, which belongs to the logical group corresponding to the vertex at level $i-1$, common to the two quorums. Hence, the lemma follows. □

Lemma 2 *In a m-level hierarchy, the HQC+ protocol guarantees that there is at least one copy at level 0 common to any pair of quorums in which one of them is write quorum.*

Proof. As a write quorum formed from the children of a vertex at level i contains either a read quorum and a blind-write quorum (see section 3.2), and any read quorum intersects any blind-write quorum, it is guaranteed that there is a vertex at level $i-1$ (among the children of that vertex at level i) common to any pair of quorums formed at level i in which one of them is a write quorum. So, the proof of this lemma is analogous to the proof of the previous lemma. □

4 Protocol Analysis

In this section, we analyze quorums size and show how data availability can be calculated in the HQC+ protocol.

4.1 Quorums Size

The size of a read quorum, at a level i of the hierarchy, is expressed by the following recurrence:

$$S_r(i) = \begin{cases} 1 & (i = 0) \\ r_i \; S_r(i-1) & (i \geq 1) \end{cases}$$

The global size of a read quorum (i.e., related to level m of the hierarchy) is given by $S_r(m)$, and equals $\prod_{i=1}^{m} r_i$. Similarly, the global size of a blind-write quorum equals $\prod_{i=1}^{m} bw_i$ ($bw_i = l_i - r_i + 1$), and is expressed by:

$$S_{bw}(i) = \begin{cases} 1 & (i = 0) \\ bw_i \; S_{bw}(i-1) & (i \geq 1) \end{cases}$$

The size of a write quorum, related to a level i of the hierarchy, depends upon read and blind-write quorum size at that same level, and is given by:

$$S_w(i) = \begin{cases} 1 & (i = 0) \\ bw_i\ S_w(i-1) \ + \ (r_i - bw_i)\ S_r(i-1) & (i \geq 1\ \ and\ \ r_i \geq bw_i) \\ r_i\ S_w(i-1) \ + \ (bw_i - r_i)\ S_{bw}(i-1) & (i \geq 1\ \ and\ \ r_i < bw_i) \end{cases}$$

As a write quorum is formed by the union of a read and a blind-write quorum, $S_w(m)$ varies from $max(S_r(m), S_{bw}(m))$, when one quorum contains the other, to $S_r(m) + S_{bw}(m) - 1$, when the quorums have only one copy in common.

4.2 Data Availability

Availability of a replicated data is defined as the likelihood that a quorum of its copies is successfully formed considering that each node of the system is independently operational with probability ρ. This probability ρ is a measure of the nodes reliability.

In the voting scheme, the availability of a replicated data with N copies for an operation that requires a quorum of q copies is given by [5]:

$$AV(N, q, \rho) = \sum_{k=q}^{N} \left(\begin{array}{c} N \\ k \end{array} \right) \rho^k (1-\rho)^{N-k}.$$

Read and blind-write availability in the HQC+ protocol are calculated similarly, but one level at a time, by expressing the availability of one level in terms of the availability of the level immediately below — exactly as it is done in the original HQC protocol. The following recurrences express read (A_r) and blind-write (A_{bw}) availability in the HQC+ protocol.

$$A_r(i) = \begin{cases} \rho & (i = 0) \\ AV(l_i, r_i, A_r(i-1)) & (i \geq 1) \end{cases}$$

$$A_{bw}(i) = \begin{cases} \rho & (i = 0) \\ AV(l_i, bw_i, A_{bw}(i-1)) & (i \geq 1) \end{cases}$$

Availability for a write operation in the HQC+ protocol is also calculated one level at a time, but is expressed in a slightly different way. Let i be a level of the hierarchy. The write availability for that level is given by the difference between the probability of forming the largest quorum of that level ($max(r_i, bw_i)$) and the probability of forming *only* that quorum, i.e., the probability of not getting write access permissions from the minimum number ($min(r_i, bw_i)$) of vertices required at level $i-1$ (among the children of a vertex at level i). So, write availability (A_w) in the HQC+ protocol is expressed by:

$$A_w(i) = \begin{cases} \rho & (i = 0) \\ AV(l_i, r_i, A_r(i-1)) & \\ \quad - \ AV(l_i, r_i, A_r(i-1) - A_w(i-1)) & (i \geq 1 \ \ and \ \ r_i \geq bw_i) \\ AV(l_i, bw_i, A_{bw}(i-1)) & \\ \quad - \ AV(l_i, bw_i, A_{bw}(i-1) - A_w(i-1)) & (i \geq 1 \ \ and \ \ r_i < bw_i) \end{cases}$$

5 Related Work

Since the HQC+ protocol is an extension of Kumar's HQC protocol, with both using the same hierarchical structure, quorums generated by the latter can also be generated by the former. In particular, when $2bw_i > l_i$ (i.e., $r_i < l_i/2 + 1$) for every level i ($1 \leq i \leq m$) of the m-level hierarchy, any blind-write quorums is also a write quorum, and so the HQC+ protocol is equivalent to the original HQC protocol. In a single level hierarchy, either HQC and HQC+ protocols clearly correspond to the traditional voting scheme.

The grid protocol [10] organizes N copies of a replicated data into an rectangular grid structure composed of x rows and y columns ($N = x \times y$). A read quorum is formed by at least one copy from all columns of the grid, and a write quorum is formed by all copies from at least one column, plus a read quorum. The size of read and write quorums are y and $x + y - 1$, respectively; when $x \approx y$, both quorums size are of the order $O(\sqrt{N})$. Forming quorums using a grid structure is nothing but a 2-level voting processing (or 2-level quorum consensus) with specific read and blind-

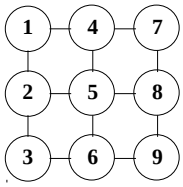

Figure 4: 9 copies organized into a 3×3 grid.

write quorums definitions for each level: at the highest level, a voting process happens among the columns of the grid (a read quorum needs all columns and a blind-write quorum needs just one); at the lowest level, a new voting process happens among copies inside each column (a read quorum needs one copy and blind-write quorum needs all of them) — then a write quorum is formed by the union of a read and a blind-write quorum. Figure 4 shows an example of 9 copies organized into a 3×3 grid structure. In this 3×3 grid, examples of read and write quorums are: {qr={1, 5, 9}, qw={1, 2, 3, 5, 9}} and {qr={3, 4, 8}, qw={3, 4, 5, 6, 8}}. In general, all quorums formed by the grid protocol using a $x \times y$ grid can be formed by the HQC+ protocol if we make $m = 2$, $l = (x, y)$ and $r = (1, y)$. Notice that this reduction is not possible in the HQC protocol because when we make $r_2 = y$ restriction $r_i < l_i/2 + 1$ is violated. As an example, if we make $r = (1, 3)$ in the hierarchy shown in figure 1, the HQC+ protocol generates exactly the same quorums generated by the grid protocol using the 3×3 grid shown in figure 4.

The hierarchical grid protocol [15] generalizes the grid protocol using a k-level hierarchical structure in which an element at a level i ($1 \leq i \leq k$) is defined as a $x_i \times y_i$ logical grid composed of level $i-1$ elements, and level 0 elements correspond to physical copies of a replicated data ($N = \prod_{i=1}^{k} x_i\ y_i$). Quorums are formed similarly to the original grid protocol, but one level at a time. The larger k the higher is data availability, and, when the number of rows equals the number of columns for all logical grids at all levels, quorums size remain of the order $O(\sqrt{N})$. Figure

5 shows an example of 16 copies organized into a 2-level hierarchical grid structure in which elements are 2×2 logical grids. In this 2-level hierarchical grid, examples of read and write quorums are: {qr={1, 6, 7, 8}, qw={1, 5, 6, 7, 8, 10, 14}} and {qr={1, 2, 11, 12}, qw={1, 2, 3, 7, 11, 12, 15}}. The HQC+ protocol is equivalent to the hierarchical grid protocol if we make $m = 2k$, $l = (x_1, y_1, x_2, y_2, \ldots, x_k, y_k)$ and $r = (1, y_1, 1, y_2, \ldots, 1, y_k)$. Figure 6 shows 16 copies organized into a 4-level hierarchy ($l = (2, 2, 2, 2)$) in which, if we make $r = (1, 2, 1, 2)$, the HQC+ protocol generates exactly the same quorums generated by the hierarchical grid protocol using the hierarchical grid structure shown in figure 5.

A grid set protocol is proposed in [2], where N copies of a replicated data are organized as a set of k grids of x rows and y columns ($N = k \times x \times y$). A read quorum is successfully formed if other read quorums are formed in at least k_r grids; similarly, a write quorum needs other write quorums in at least k_w grids. Values k_r e k_w are set according to the traditional voting scheme, following restrictions $k_r + k_w > k$ and $2k_w > k$. As the HQC+ protocol is a generalization of both the grid protocol and the simple voting protocol, it clearly corresponds to the grid set protocol if we make $m = 3$, $l = (x, y, k)$ and $r = (1, y, k_r)$.

The tree quorum protocol [4] uses a logical tree structure to organize N copies of a replicated data. In this structure, copies correspond to all vertices of the tree, instead of just their leaves (as it happens here and in the HQC protocol). In this tree structure, it is possible to form a write quorum of size varying from $logN$ to $N/2 + 1$ copies, and a read quorum of only one copy corresponding to the root of the tree. The major drawback of this protocol is that in its tree structure, as long as replicas near the root take place in more quorums than those near the leaves, copies share unequal responsibility in controlling the replicated data. This drawback is avoided by the HQC and HQC+ protocols since in their tree structures all copies are located at the same lowest level of the hierarchy, and thus play equal roles during synchronization.

All protocols that use a logical structure to form quorums are *static*, i.e., they do not adapt to changes in the system configuration caused by

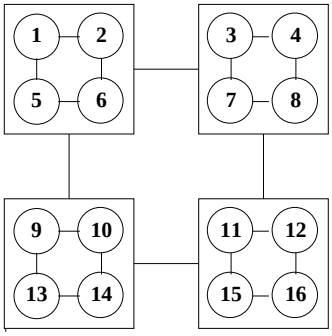

Figure 5: 16 copies organized into a 2-level hierarchical grid.

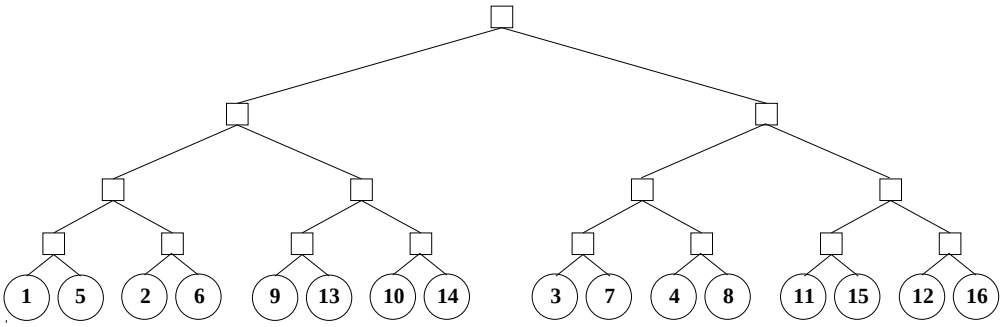

Figure 6: 16 copies organized into a 4-level hierarchy.

eventually failures — as do the so called dynamic protocols [17, 8, 13, 19, 1] [5]. It is interesting to note, however, that, as suggested in [18], through a rule that unambiguously imposes a desired logical structure on a set of copies, static protocols can be converted to dynamic ones. For conversion of the HQC and HQC+ protocols, for instance, we would need a rule that also defines read quorums for each level of the hierarchy.

Multidimensional voting [6] is another generalization of the traditional voting mechanism that can be used to implement, by a voting process among copies of the replicated data, any quorum-based protocol — in particular, the grid protocol, the HQC protocol and also the HQC+ protocol. In the multidimensional voting scheme, each copy is assigned a multidimensional vector in which each position has an independent number of votes. Votes from each dimension are collected similarly to the traditional voting scheme, and an operation is allowed to be performed on the replicated data if quorums are successfully formed in at least a stipulated number of dimensions. Since the multidimensional voting scheme is a generic concept, to produce a multidimensional voting assignment to all copies, it initially needs a logical structure as a reference, or requires that all possible quorums be explicitly enumerated — what, in the latter case, is not conceivable to a large number of copies. Hence, although all quorums generated by the HQC+ protocol may also be generated by multidimensional voting, the hierarchical logical structure and the way to define quorums are still required in order to calculate quorums size and data availability, or even to serve as reference for a multidimensional voting assignment. In [6], it is also proposed a way to implement a hierarchical structure in which copies are not located at the same level. This irregular structure (or incomplete tree) is not interesting because it relies different degree of responsibility in synchronization on the set of copies. Another observation is that with multidimensional voting copies are not aware of their positions in the logical structure and, therefore, can not take advantage of such knowledge to further reduce communication costs.

[5] Dynamic protocols provide higher availability, but rather require expensive reconfiguration procedures.

6 Practical Examples

Reducing the ratio *(quorum size)/(total number of copies)* is the best way to also reduce communication costs, make an effective load sharing and, as consequence, achieve better system response time. In [10], this ratio is investigated for the grid and simple voting protocols under the following conditions: a) data availability of $1 - 10^{-6}$ for read operations and of 0.9955 for write operations are required; b) node reliability is 0.95; and c) read operations arrival rate is about five times the corresponding arrival rate of write operations. In this investigation, it was verified that, in order to satisfy such high availability requirements, a minimum of 10 copies (qr=4 and qw=7) are needed in the voting scheme, while the grid protocol needs at least 30 copies (organized into a 6×5 grid with qr=5 and qw=10). Although more copies are required in order to the grid protocol to achieve the same level of availability as the voting scheme, the load sharing feature of the grid protocol provides a significant decrease in system response time. It was also shown in [10] that the voting scheme can not achieve this same level of load sharing even by increasing the number of copies.

With the HQC and HQC+ protocols, the minimum data availability required in [10], if we consider the same node reliability of 0.95, is already achieved starting from hierarchy configurations with 14 copies of the replicated data. Table 1 shows all cases, for the HQC, grid and HQC+ protocols, up to 30 copies, in which these minimum data availability requirements are satisfied with the best relations between quorums size and the total number of copies. Notice that these quorums relations, for all entries of table 1 and for each one of the three protocols, are smaller than the minimum possible quorums relations, with the same number of copies, in the voting scheme. The HQC+ protocol, however, having greater flexibility to define quorums, achieves quorums smaller than those achieved by the original HQC protocol. Notice also that all hierarchy configurations in table 1 used by the HQC+ protocol can not be used by the HQC protocol, since at least one of their levels violates restriction $r_i < l_i/2 + 1$. An additional observation is that with 30

Number of Copies	Protocol	l_i			r_i			qr	qw
		l_1	l_2	l_3	r_1	r_2	r_3		
14	HQC	7	2		4	1		4	8
	HQC+	7	2		2	2		4	8
16	HQC	4	4		2	2		4	9
	HQC+	4	4		3	1		3	9
18	HQC	3	3	2	2	2	1	4	8
	HQC+	2	3	3	1	3	1	3	8
20	HQC	5	4		2	2		4	12
	HQC+	4	5		3	1		3	11
22	HQC	11	2		4	1		4	16
	HQC+	2	11		2	2		4	12
24	HQC	4	3	2	2	2	1	4	12
	HQC+	3	8		3	1		3	10
25	HQC	5	5		3	2		6	12
	HQC+	5	5		4	1		4	12
26	HQC	13	2		5	1		5	18
	HQC+	13	2		3	2		6	14
27	HQC	3	3	3	2	2	1	4	12
	HQC+	3	9		3	1		3	11
28	HQC	7	4		4	1		4	16
	HQC+	2	7	2	2	1	2	4	10
30	HQC	5	3	2	3	2	1	6	12
	Grid	6	5		1	5		5	10
	HQC+	3	10		3	1		3	12

Table 1: Number of copies and hierarchy configurations that provides minimum availability requirements in the HQC, grid and HQC+ protocols.

copies, if we consider the frequency of read operations much higher than the frequency of write operations, the best quorums are obtained by the HQC+ protocol (qr=3 and qw=12) — in contrast with qr=5 and qw=10 of the grid protocol, and with qr=6 and qw=12 of the HQC protocol (quorums of exactly these size, 3 and 12, are obtained by the grid protocol if it uses a 10×3 grid; nevertheless, such structure does not permit that the grid protocol achieves the minimum data availability required for write operations).

7 Conclusions

This paper extended the hierarchical quorums consensus (HQC) protocol proposed in [14] adding the blind-write operation to the read and write operations normally performed on replicated data, and by presenting a new way to define quorums over the hierarchical logical structure. This extension, called extended hierarchical quorum consensus (HQC+), provided greater flexibility to define quorums even under high availability constrains, making possible that smaller quorums be reached and, as a consequence, better system response time be achieved.

Moreover, the HQC+ protocol turned out to be a generalization of all static protocols that use a logical grid structure to control replicated data, including the simple voting protocol and the HQC protocol itself. In fact, we argue that any protocol that forms quorums by organizing copies into a symmetrical logical structure, i.e., a structure in which copies have the same responsibility in controlling the replicated data (which is not the case of the tree quorum protocol proposed in [4]), can be seen as a special instance of the HQC+ protocol.

References

[1] A. El Abbadi and S. Toueg. Maintaining Availability in Partioned Replicated Databases. *ACM Transactions on Database Systems*, 14(2):264–290, June 1989.

[2] D. Agrawal and A. El Abbadi. Exploiting Logical Structures of Replicated Databases. *Information Processing Letters*, 33(5):255–260, January 1990.

[3] D. Agrawal and A. El Abbadi. Storage Efficient Replicated Databases. Technical Report TRCS90-5, Department of Computer Science, University of California, Santa Barbara, 1990.

[4] D. Agrawal and A. El Abbadi. The Generalized Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data. *ACM Transactions on Database Systems*, 17(4):689–717, December 1992.

[5] M. Ahamad and M. H. Ammar. Performance Characterization of Quorum-Consensus Algorithms for Replicated Data. *IEEE Transactions on Software Engineering*, 15(4):492–495, April 1989.

[6] Mustaque Ahamad, Mostafa H. Ammar, and Shun Yan Cheung. Multidimensional Voting. *ACM Transactions on Computer Systems*, 9(4):399–431, November 1991.

[7] Daniel Barbara and Hector Garcia-Molina. The Reliability of Voting Mechanisms. *IEEE Transactions on Computers*, C-36(10):1197–1208, October 1987.

[8] Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster. Increasing Availability Under Mutual Exclusion Constraints with Dynamic Vote R reassignment. *ACM Transactions on Computer Systems*, 7(4):394–426, November 1989.

[9] Philip A. Bernstein and Nathan Goodman. The Failure and Recovery Problem for Replicated Databases. In *Proceedings of the* 2^{nd} *Symposium on Principles of Distributed Computing*, pages 114–122. ACM, August 1983.

[10] S. Y. Cheung, M. Ammar, and M. Ahamad. The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data. In *Proceedings of the* 6^{th} *International Conference on Data Engineering*, pages 438–445. IEEE, 1990.

[11] Susan B. Davidson, Hector Garcia-Molina, and Dale Skeen. Consistency in Partioned Networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.

[12] D. K. Gifford. Weighted Voting for Replicated Data. In *Proceedings of the* 7^{th} *Symposium on Operating Systems Principles*, pages 150–162. ACM, December 1979.

[13] Sushil Jajodia and David Mutchler. Dynamic Voting Algorithms for Maintaining the Consistency of a Replicated Database. *ACM Transactions on Database Systems*, 15(2):230–280, June 1990.

[14] Akhil Kumar. Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data. *IEEE Transactions on Computers*, 40(9):996–1004, September 1991.

[15] Akhil Kumar and Shun Yan Cheung. A High Availability $\sqrt{N}$ Hierarchical Grid Algorithm for Replicated Data. *Information Processing Letters*, 40(6):311–316, December 1991.

[16] Jehan-François Pâris. Voting with Witnesses: A Consistency Scheme for Replicated Files. In *Proceedings of the* 6^{th} *Conference on Distributed Computing Systems*, pages 606–612. IEEE, June 1986.

[17] Jehan-François Pâris and Darrel D. E. Long. Efficient Dynamic Voting Algorithms. In *Proceedings of the* 4^{th} *International Conference on Data Engineering*, pages 268–275. IEEE, February 1988.

[18] Michael Rabinovich and Edward D. Lazowska. Improving Fault Tolerance and Supporting Partial Writes in Structured Coteries Protocols for Replicated Objects. In *SIGMOD*, pages 226–235. ACM, June 1992.

[19] Jin Tang and N. Natarajan. A Scheme for Maintaining Consistency of Replicated Files in Partioned Distributed Systems. In *Proceedings of the* 5^{th} *International Conference on Data Engineering*, pages 530–537. IEEE, 1989.

[20] Robert H. Thomas. Majority Concensus Approach to Concurrency Control for Multiple Copy Databases. *ACM Transactions on Database Systems*, 4(2):180–209, June 1979.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

04/93 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

05/93 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

06/93 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

07/93 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

08/93 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

09/93 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

10/93 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

11/93 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br