Multi-dimensional Quorum Sets for Read-Few Write-Many Replica Control Protocols

Bujor Silaghi
bujor@cs.umd.edu

Pete Keleher
keleher@cs.umd.edu

Bobby Bhattacharjee
bobby@cs.umd.edu

ABSTRACT

We describe d-spaces, a replica control protocol defined in terms of quorum sets on multidimensional logical structures. Our study is performed in the context of transactional replica control protocols that support the strongest form of replica consistency guarantees, one-copy serializability. This work is primarily motivated by asymmetrical access patterns, where the number of read accesses to data are dominant relative to update accesses, i.e. where the consistency protocols should be *read-few write-many.*

We show that quorums on d-spaces are optimal with respect to quorum group sizes (message complexity). We present a detailed availability analysis for read and write operations. For highly asymmetrical access patterns, our approach approximates the read-one write-all protocol with respect to read efficiency, while maintaining configurable levels of availability for write operations. Specifically, we show that implementing a read-few write-many replica protocol using d-spaces yields both superior operation availability, as well as message complexity, to the hierarchical quorum consensus method.

1. INTRODUCTION

We describe a method for defining replica quorum groups on multi-dimensional logical structures. Our contribution is twofold. First, we argue that implementing read-few write-many replica control protocols using our approach is superior to existing methods from the efficiency as well as availability standpoint. We provide analytic bounds that are independent of whether the method is implemented using local connectivity or global views. Second, unlike other structured quorum approaches, we propose using local connectivity instead of global views to arrange the sites into a logical structure. This addresses the inherent rigidity of logical structures based on global views, and allows membership changes without requiring global reconfiguration phases. A local reconfiguration protocol is run to eliminate holes in the structure created by failing sites.

Data replication is used in distributed systems to increase data availability, and to improve the access efficiency to a data item. Storing copies of data at multiple sites increases the likelihood of data remaining accessible to users despite host and communication failures. Having multiple up-to-date copies of data offers a choice at retrieval time, and allows the system to efficiently fetch a copy that is locally present or at a nearby site. The benefits of replication come at the price of complex and expensive synchronization mechanisms needed to maintain the consistency and integrity of data. The overhead induced by replica control protocols is in direct relation with the consistency guarantees offered. The stronger these guarantees are, the less efficient and less available are operations performed on replicated data items.

Ideally, the set of all copies of an object should appear as a single item to clients. Further, efficient schemes are desirable in the sense that only a small fraction of the set of all copies should be accessed per operation. This is especially important for read operations, which tend to represent the dominant access mode to data for most application classes. Finally, we would like to achieve both strict consistency and operational efficiency without sacrificing the availability of data, as this is the main reason for replicating data.

The consistency condition referred to is one-copy equivalence [7]. Coupled with serializable executions that many databases provide, the correctness criteria for replica control protocols in transactional environments is one-copy serializability [6]. Among the replica control protocols that can achieve one-copy serializability — when used in conjunction with appropriate mechanisms for concurrency control (e.g. two-phase locking), and atomic commitment (e.g. two-phase commit) — we identify the read-one write-all approach (ROWA) [4], the primary-copy approach [24], the quorum consensus scheme [11], and the available copies method [12].

The read-one write-all approach is one of the simplest protocols for managing replicated data. Read operations are allowed to read any copy, and write operations are required to write all copies. The read-one write-all protocol is imbalanced: reads are executed at a very low cost and are highly available. Write operations, on the other hand, are inefficient and have very low availability even compared to the case when data is not replicated: all copies need to be operational for a write to proceed.

The quorum consensus protocol generalizes the read-one write-all approach by enforcing a relaxed quorum intersection condition: read and write operations need only intersect in a pairwise fashion on at least one site. The low write availability of the read-one write-all protocol is addressed at the expense of more costly read operations. One class of quorum protocols (e.g. the grid protocol [9], the tree protocol [2], and the hierarchical quorum consensus protocol [16]) achieve efficiency by imposing a logical structure on the set of copies, and using structural information to create intersecting quorums. Operations have low message complexity, but quorums on logical structures are vulnerable to specific site failures and result in degraded operation availability.

There is a clear trade-off between the efficiency of a quorum protocol and its operational availability. The break-even point of this trade-off depends primarily on the logical structure (or the lack thereof) that arranges participating sites. In this paper we define a new quorum protocol based on logical structures, *d-space quorums*, and argue that its efficiency as well as its availability are superior to existing protocols. In particular, we show that implementing a read-few write-many replica protocol using d-spaces yields superior operation availability as well as message complexity to the hierarchical quorum consensus method.

Sites participating in d-space quorums are arranged in a d-dimensional space, and quorum groups are formed through subspace projection. Defining quorums on d-spaces is shown to be optimal with respect to quorum group sizes (message complexity). For highly asymmetrical access patterns, our approach approximates the read-one write-all protocol with respect to read efficiency, while maintaining acceptable levels of availability for write operations. The method subsumes some of the existing replica control protocols such as read-one write-all, and the Grid protocols. Further, d-space replica quorums are highly flexible in the sense that arbitrary read/write access ratios can easily be modeled. Finally, read accesses can be executed orders of magnitude more efficiently than updates without compromising the availability of either.

The remainder of this paper is structured as follows. In Section 2 we briefly describe existing quorum protocols with emphasis on structured quorums. Among the structured quorum approaches, the hierarchical quorum consensus scheme receives special attention. Section 3 defines quorum sets on multi-dimensional spaces, argues about the optimality of the approach with respect to communication complexity, and shows how the read-few write-many protocol can be implemented using d-spaces. In Section 4 we carry a detailed availability analysis and compare the performance of our approach with the hierarchical quorum consensus method. A reconfiguration mechanism that combines local connectivity and global views is discussed in Section 5. We summarize our findings and conclude the paper in Section 6.

2. RELATED WORK

Quorum protocols have received a lot of interest both in the database and the distributed systems communities. Replica control protocols based on quorum schemes can guarantee one of the strongest consistency conditions for replicated data, namely one-copy equivalence. If coupled with concurrency control mechanisms that interleave the execution of transactions in a serializable fashion, one-copy serializability results.

Synchronization takes place by defining groups of sites that need to agree before launching an activity, and requiring the intersection of groups defined for conflicting activities. A read operation on a copy conflicts with all write operations on any copy of the object. A write operation on a copy conflicts with all read and write operations. We will refer to the group of sites needed to perform a read (write) operation as the *quorum group* for that operation. The collection of read (write) quorum groups is called a read (write) *quorum set*. Thus, any element of a read quorum set must intersect all elements of a write quorum set, which in turn must intersect among themselves in a pairwise fashion.

2.1 Voting

Gifford defines quorum sets in terms of weighted voting [11] for synchronizing concurrent accesses to shared files. If the total number of votes is v , v_r votes are needed to read a file, and v_w votes are needed to write a file, such that (i) $v_r + v_w > v$ and (ii) $2v_w > v$. A site can be assigned more than one vote, and the distribution of votes to sites need not be uniform. Version vectors are used to identify the latest update, and locking is used to guarantee serial consistency.

Weighted voting does not assume a logical structure of copies when realizing quorums. Further, for symmetrical quorum configurations ($v_r \approx v_w$) the approach can tolerate failures of up to half of the participating sites. In this case however the communication costs are high as each transaction involves accessing half the copies. Weighted voting can model scenarios where read operations dominate accesses to data ($v_r \ll v_w$). In such cases, weighted voting approximates the read-one write-all approach both with respect to its poor write availability, as well as its high costs for write operations. In fact, the read-one write-all protocol is a special case of voting assignment for which $v_r = 1$, and $v_w = v$.

The notion of quorum sets is closely related to *coteries* as defined by Garcia-Molina and Barbara [10]. In coteries, an added minimality condition is imposed upon quorums groups, and no distinction is made between different types of quorum group conflicts. For efficiency reasons, quorum sets exploit this distinction between read and write operations. The authors also show that quorum based schemes are more expressive than voting. More exactly, there are quorum sets that cannot be modeled through voting assignments such that mutual exclusion is still guaranteed.

Thomas defines majority quorums as quorum sets for which each quorum group contains a majority of copies [26]. Again, this is a special case of weighted voting for which $v_r = v_w = \lfloor v/2 \rfloor + 1$. As note above, this assignment provides the best symmetric availability for read and write operations. The replica control protocol proposed by Thomas uses timestamps [17] instead of version vectors to synchronize accesses to a replicated database. When timestamps are used, intersection of write quorum groups is not necessary [25], while read quorums still need to intersect write quorums. Different non-intersecting writes are registered with unique timestamps generated from a totally ordered domain, and a read operation will be able to identify the latest update as long as it intersects with all write quorums. A similar technique (logs with timestamped entries and non-intersecting writes) has been used by Herlihy [13] in defining a general consensus method for abstract data types that extends the read/write semantics of operations on data.

2.2 Structured Quorums

Quorum sets defined on logical structures use structural information to define intersecting quorum groups. We briefly present the Grid protocol, the tree protocol, and the hierarchical quorum consensus protocol.

Maekawa proposed using finite projective planes to obtain a distributed mutual exclusion protocol [18] where the number of sites contacted per transaction (a quorum group) is on the order of $O(\sqrt{N})$. As an alternative protocol having the same performance characteristics, Maekawa further suggested organizing the sites in a two dimensional grid. Assuming a $\sqrt{N} \times \sqrt{N}$ logical grid, a site requesting the lock would contact all the sites found in some arbitrary line and column of the grid. Every two pairs of lines and columns intersect each other, and correct arbitration is ensured for lock grant/release primitives. Starting with Maekawa's finite projective planes approach and given his conditions, it can easily be shown that the solution is optimal. More exactly, quorum sets such that (i) each quorum group has (roughly) the same number of elements, and (ii) each element appears in (roughly) the same number of groups, will consist of quorum groups having $O(\sqrt{N})$ elements.

Maekawa's mutual exclusion protocol has inspired Cheung el al. in developing the Grid replica control protocol [9]. Quorum groups for read operations consist of one line, and quorum groups for write operations consist of one line and one column. The authors observe that instead of a line, for both read and write quorums, a more relaxed configuration that requires one node in each column (a *column cover*) can be employed. Two-phase locking is used for concurrency control and timestamp ordering is mentioned as an alternative scheme. The Grid protocol has low communication costs ($O(\sqrt{N})$), and is best suited for scenarios where the frequency of read and write operations are on the same order.

The tree protocol proposed by Agrawal and El Abbadi organizes the set of copies in a binary tree with $\log N$ levels [2]. A quorum group is formed by including all the sites in some arbitrary path from the root to a leaf. If a site along a path is unavailable, the quorum group is formed by substituting the failed site with the two child nodes of that site, and continuing recursively along both paths to the leaf level of the tree. The tree protocol features the lowest message complexity ($O(\log N)$) among all structured quorum schemes, assuming no site failures. In the presence of failures, the algorithm degrades gracefully as progressively more sites are involved in a quorum group, for a maximum of $N/2$ (when all sites on all levels but the leaf level have failed).

Despite its low communication costs and good tolerance to site failures, the tree protocol is less appealing when considering the distribution of accesses over the set of copies. The root site is part of all quorum groups (assuming no failures), while a leaf site is part of $N/2$ times fewer quorum groups. The tree protocol is not truly distributed and employs a weak form of decentralization to ensure exclusion of accesses.

Kumar extends weighted voting to voting on multiple levels of a hierarchy comprising the set of all replicas [16]. In contrast to the tree protocol, physical copies of objects are stored only at the leaves of the tree, while all other levels serve a logical grouping purpose. In effect, the protocol performs a hierarchical partitioning of the replica set. Given a perfectly balanced tree with $m+1$ levels (with the root on level 0 and replicas on level m) such that a node on level i has l_{i+1} children, the overall number of replicas is $\prod_{i=1}^{m} l_i$. A node assembled on level i must in turn recursively assemble r_{i+1} (w_{i+1}) of its l_{i+1} children nodes on level $i+1$ for a read (write) quorum group. The root node is part of all read (write) quorum groups. The quorum intersection condition is satisfied if (i) $r_i + w_i > l_i$, and (ii) $2w_i > l_i$ for all levels $i = \overline{1,m}$.

A read quorum group defined by the hierarchical quorum consensus scheme consists of $\prod_{i=1}^{m} r_i$ copies, and a write quorum groups of $\prod_{i=1}^{m} w_i$ copies. Weighted voting can be regarded as the special case of a two level hierarchy ($m = 1$). Optimal quorum group sizes are obtained for the hierarchical consensus method when each group contains three subgroups, i.e. $l_i = 3$. In this case symmetrical quorum groups consist of $N^{0.63}$ sites, which is closer to the optimal complexity $O(\sqrt{N})$ than to the default weighted voting performance ($\lfloor N/2 \rfloor + 1$). Further, the hierarchical quorum consensus method allows for imbalanced quorum groups for read and write operations to be specified. For these reasons we have chosen it to contrast the performance and availability of the approach that we advocate in this paper.

Logically structured quorum sets cannot be modeled with voting assignments in the general case [10, 5], making the quorum consensus approach more appealing than weighted voting [11] in a number of instances. Multidimensional (MD-) voting [5] addresses this problem by generalizing weighted voting to voting in multiple dimensions. The d-space quorum consensus method is conceptually different than MD-voting. The latter defines an abstract voting space, and requires a quorum of votes to be gathered in every dimension of interest. However, MD-voting can emulate d-space quorums in a rather inefficient manner. For an m by n 2-space quorum set, the MD-voting mapping is achieved by defining voting assignments in an n-dimensional space, and requiring that a quorum of votes be gathered in all n dimensions.

3. QUORUMS FOR REPLICATED DATA

We define quorum groups on multi-dimensional spaces and show how read-few write-many replica control protocols can be implemented using our method.

3.1 D-Space Quorums

Assume we have N replicas of a data item arranged in a d-dimensional discrete space D, and that each dimension i in D is indexed from 1 to n_i, i.e. $N = \prod_{i=1}^{d} n_i$. D is an abstract space, and the N replicas may not necessarily correspond to physical copies. For this reason we will refer to hosts of replicas as *nodes* and not *sites*. In Section 5.2.2

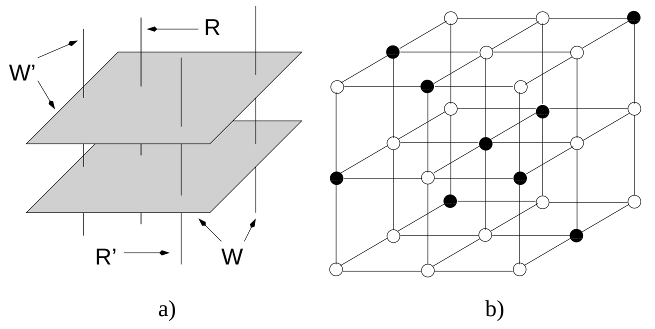

Figure 1: Example of 3-space read and write quorum groups. a) A read quorum (R and R') is a line and intersects all write quorums. A write quorum (W and W') is a plane and a line, and intersects all read quorums and all other write quorums. b) A cover of a plane (filled points) can be used instead of the plane to form a write quorum group.

we discuss how nodes are mapped to sites. Until then we assume that nodes are sites.

We choose k of the d dimensions in D, and let $u_{i_1}, u_{i_2}, \ldots, u_{i_k}$ be k arbitrary coordinates on selected dimensions $i_1, i_2, \ldots i_k$. Similarly, let $v_{j_1}, v_{j_2}, \ldots, v_{j_{d-k}}$ be arbitrary coordinates on the rest of the $d-k$ dimensions $(j_1, j_2, \ldots, j_{d-k})$.

We define subspace U to be:

$$\begin{aligned} U = &\{(x_1, x_2, \ldots, x_d) | \\ &(x_1, x_2, \ldots, x_d) \in D \wedge (x_{i_t} = u_{i_t})_{t=\overline{1,k}}\} \end{aligned} \quad (1)$$

and subspace V to be:

$$\begin{aligned} V = &\{(x_1, x_2, \ldots, x_d) | \\ &(x_1, x_2, \ldots, x_d) \in D \wedge (x_{j_t} = v_{j_t})_{t=\overline{1,d-k}}\} \end{aligned} \quad (2)$$

Subspaces U and V overlap and their intersection is minimal; there exists a single point (node) common to U and V. The intersection point is given by coordinates $u_{i_1}, \ldots, u_{i_k}$, $v_{j_1}, \ldots, v_{j_{d-k}}$ written in canonical order. In a 2-dimensional space, U and V represent intersecting lines parallel to the two axes. Note that U and V factorize space D in the sense that $|D|=|U||V|$. Thus each of the two subspaces contains considerably fewer sites that the original space. In particular, the sum $|U| + |V|$ is minimized when $|U| = |V|$.[1]

Let a read quorum group be V, and a write quorum group be $V \cup U$. Any two write quorum groups intersect, and any read quorum group intersects any other write quorum group. The quorum intersection property is satisfied and reads and writes are serializable. In particular, one-copy serializability [6] is guaranteed. On the left side of Figure 1 we illustrate read and write quorum groups for a 3-space quorum set.

For clarity of exposition we will assume hereafter that the extension of the replica space is the same along all dimensions, i.e. $n_i = N^{\frac{1}{d}}$ for all $i = \overline{1,d}$. All the results presented can easily be extended to account for the general case formally defined above. Given the new constraints, the replica space resembles a hyper-cube in a d-dimensional space, with space V containing $N^{\frac{k}{d}}$ points and space U containing $N^{\frac{d-k}{d}}$ points. A read quorum group will thus consist of $N^{\frac{k}{d}}$ nodes while a write quorum group will consist of $N^{\frac{k}{d}} + N^{\frac{d-k}{d}} - 1$ nodes.

[1]The condition holds only when d is even and $k = d/2$. If d is odd the sum $|U| + |V|$ is minimized when $k = \lfloor d/2 \rfloor$ or $k = \lceil d/2 \rceil$.

3.2 The Read-Few Write-Many Approach

More interesting for distributed systems are quorums sets that are highly asymmetrical, i.e. for which d is relatively high compared to k. One such case is given by read quorum groups consisting of $N^{\frac{1}{d}}$ nodes (a line), and write quorum groups consisting of $N^{\frac{1}{d}} + N^{\frac{d-1}{d}} - 1$ nodes (a line and a hyper-plane). We call this instantiation of d-space quorum sets the *read-few write-many* approach (RFWM).

Read-few write-many replica control protocols are amenable to scenarios where the frequency of read operations is orders of magnitude higher than the frequency of write operations. We call the ratio of frequencies for read and write operations the read/write access ratio. The communication complexity of a replica control protocol is the expected number of replicas to be contacted per operation. Our goal is to minimize the protocol's communication complexity, irrespective of operation type (i.e. read or write). Therefore, the ratio of quorum group sizes for read and write operations should be inverse to the read/write access ratio. By ρ_{wr} we denote the ratio of the write quorum size to the read quorum size. When ρ_{wr} equals the read/write access ratio, the communication complexity of a read-few write many replica control protocol is minimized. Any proportion of read and write operations can be modeled by choosing appropriate values for k and d (or more generally, for d and dimension extensions n_i).

Fault tolerance is reflected in the availability of the last update (data availability), and the availability of read and write operations. Data availability in RFWM is comparable to that offered by the read-one write-all protocol: a very high fraction of nodes need to fail for the last update to be lost. Read operations are robust: any line of $N^{\frac{1}{d}}$ nodes needs to be operational for a read to succeed. There are $N^{\frac{d-1}{d}}$ candidate lines to choose from. The read-one write-all protocol is deemed unsatisfactory due to its stringent requirement that all copies be available whenever an update occurs. The read-few write-many approach approximates ROWA from an efficiency standpoint, and dramatically improves over its write availability. A line of $N^{\frac{1}{d}}$ and an additional set of $N^{\frac{d-1}{d}} - 1$ nodes need to be operational for a write to successfully commit.

Even though we defined read and write quorum groups in terms of projective subspaces, the set of $N^{\frac{d-1}{d}}$ nodes in a write quorum group need not conform strictly to the definition. We allow any cover of a $N^{\frac{d-1}{d}}$ hyper-plane of nodes to act as the second component of a write quorum group. A cover of a hyper-plane is any set of points such that their projection covers the whole hyper-plane. For our discrete space we are interested in the minimal cover of a hyper-plane, i.e. a cover having exactly $N^{\frac{d-1}{d}}$ points. Relaxing a write quorum group to a line combined with the cover of a hyper-plane greatly improves the availability of write operations. On the right side of Figure 1 we show a (minimal) cover for a plane that can be part of a 3-space write quorum group. Note that any of the three planes parallel to the base of the cube is covered by the cover shown.

The choice of strict subspaces and covers that we advocate is not unique. We could have relaxed the definition of a write quorum to be a hyper-plane and the cover of a line. In this case a read quorum group would be itself the cover of a line and would have excellent availability. However, the availability of write operations would be adversely affected to the point of making it not substantially better than in the read-one write-all protocol. In Figure 1, write availability would translate to having one of the three planes parallel to the base of the cube fully operational.

3.3 Optimality

As noted above we expect read and write quorum group sizes to be inversely proportional to the access frequency for the corresponding operations. We argue that replica control protocol using d-spaces feature optimal communication complexity, i.e. read and write quorum sizes are minimal for a given access ratio. We require the following constraints on any quorum set defined on d-spaces:

QS1 Each read (write) quorum group in the set has r (w) nodes. The condition ensures that the message complexity of an operation is independent of the quorum group chosen.

QS2 Each node appears in at least one quorum group. The condition ensures that all copies are used effectively.

QS3 Each node is contained in the same number of quorum groups. The condition ensures uniform load sharing over the set of all copies (assuming quorum groups are selected uniformly at random when performing operations).

Given conditions QS1-3, Theorem 1 states the optimality of the approach. The following Lemma will help us prove the theorem.

LEMMA 1. *A set $\mathcal{W}$ that intersects all elements of a read quorum set satisfying conditions QS1-3 contains at least $w = N/r$ elements.*

PROOF. Assume each node is contained in k distinct quorum groups (by QS3). We also have that $k > 0$ (by QS2). The total number of nodes, considering all duplicates as distinct elements, is kN. Since there are r nodes in each quorum group (by QS1), there are kN/r groups in the quorum set.

We construct $\mathcal{W}$ starting with the empty set, such that $\mathcal{W}$ intersects all kN/r quorum groups. Every node added to $\mathcal{W}$ is contained in exactly k of the groups, and will ensure the intersection of $\mathcal{W}$ with the corresponding groups. Thus, with the addition of one node we can cover the intersection of at most k groups in the quorum set. Since there are kN/r quorum groups, at least N/r nodes need be added to $\mathcal{W}$ to cover the intersection of all groups in the quorum set. $\square$

THEOREM 1. *Read quorum sets defined using d-spaces are optimal with respect to quorum group size for any read/write access ratio ρ_{wr}. Write quorum sets are optimal within a factor of 2.*

PROOF. We assume that k and d exist such that $N^{\frac{d-k}{d}} \approx N^{\frac{k}{d}}\rho_{wr}$, and define d-space read quorum groups of size $N^{\frac{k}{d}}$ and write quorum groups of size $N^{\frac{k}{d}} + N^{\frac{d-k}{d}} - 1$ in the usual manner. We have that $(N^{\frac{k}{d}} + N^{\frac{d-k}{d}} - 1)/N^{\frac{k}{d}} = O(w/r)$, and the quorums satisfy the read/write access ratio.

A read quorum group contains $N^{\frac{k}{d}}$ nodes. Every node appears in exactly one of the read quorum groups. In fact, the read quorum set defines a partition on the set of all copies, where each member of the partition has the same number of nodes. Conditions QS1-3 are thus satisfied and by Lemma 1 we have that write quorum groups must contain at least $N^{\frac{d-k}{d}}$ elements. Since $N^{\frac{k}{d}} + N^{\frac{d-k}{d}} - 1 \leq 2N^{\frac{d-k}{d}}$ (assuming $k \leq d-k$), we have that write quorum groups are within a factor of 2 from the optimal size.

Read quorum groups cannot contain less than $N^{\frac{k}{d}}$ nodes since that would proportionately increase the size of write quorums (as given by Lemma 1). This would break the read/write access ratio. Thus, read quorum groups are optimal with respect to size. $\square$

Note that write quorum groups are within a factor of 2 from optimality due to their V subspace component ($N^{\frac{k}{d}}$). For read-few write-many protocols we have that $N^{\frac{k}{d}} \ll N^{\frac{d-k}{d}}$, and write quorum sizes are themselves close to optimality.

We show how k and d can be chosen such that the access ratio condition is satisfied. Given N nodes and access ratio ρ_{wr} we are looking for quorum sizes for write and read operations, w and r, such that (i) $w = N/r$, and (ii) $\rho_{wr} = w/r$. Thus we have that $w = \sqrt{N\rho_{wr}}$ and $r = \sqrt{N/\rho_{wr}}$. N can be factorized in the list of its prime factors. Each prime factor in the list appears as many times as its power of factorization. The list can then be partitioned in two sublists such that the multiplication of prime factors in one list approximates w, and in the second list approximates r.

Given w and r, values for k and d can easily be identified such that $N^{\frac{k}{d}}$ approximates r, and $N^{\frac{k}{d}} + N^{\frac{d-k}{d}} - 1$ approximates w. For the general case, where the extension of the replica space does not have to be the same along all dimensions, we have more flexibility in choosing values for k and d. If N has few prime factors (e.g. N is a prime number itself), a neighboring number of N can be factorized instead of N. In this case, a few holes will be present in the structure, and quorum groups containing the holes will not be operational. In Section 5.2.2 we discuss how nodes can be mapped onto sites such that N is chosen at will and any number of physical copies is naturally accommodated.

3.4 Replica Control Protocol

We briefly describe a read-few write-many replica control protocol that uses quorum consensus on d-spaces. We present the classical approach of using version numbers to identify the latest update, and locking to enforce mutual exclusion.

3.4.1 Access Semantics and Locking

Each copy has associated a version number. A read operation will read all the values of nodes in a read quorum group, together with their version numbers. The highest ranking version is considered to be the latest update of the item, and its associated value is the result of the read. A write operation will read the values of a subset of the nodes in the write quorum group, identify the highest ranking version, and write its associated value to another subset of the nodes in the group. For RFWM, a write operation will read from a line and write to the cover of a hyper-plane.

For both reads and writes, all the nodes in the corresponding quorum group are locked. If locking fails for a read group, a different group is chosen randomly or deterministically. Similarly, if locking of a write group fails on the line, the whole line needs to be replaced. If locking of a write group fails on the cover, the cover is changed by replacing only the unsuccessfully locked nodes. Locking failures is detected through timeouts. A timeout can be caused by failed nodes or by deadlocks. Locking lines cannot cause deadlocks since they form a partition of the replica space. Locking covers, on the other side, can easily deadlock two or more concurrently executing write operations. If no quorum group can be locked it means the system is unavailable due to massive node failures. We discuss fault tolerance in Section 5. Note that performing operations on quorum groups can benefit from multicast services provided at the network or application layer.

The locking procedure described above provides transactional semantics for accesses, and ensures one-copy serializability. If single-access semantics is desired, read quorums need not be locked. A read operation can be performed by reading a read quorum group and identifying the value with the highest ranking version. Similarly, the read component of a write quorum (a line for RFWM) need not be locked for write operations. However, the write component of a write quorum group needs to be locked to guarantee (single) update consistency.

3.4.2 Locking and Replication Granularity

The performance of a replica control protocol also depends on the update semantics, and the granularity of locks and of the replicated objects.

Update atomicity can be defined irrespective of the size of a data item. The cost of performing an update grows in proportion to both the size of the data item (locking granularity), as well as the number of items updated (write quorum size). Partial writes allow updates to occur on granularity smaller than the locking granularity. Distributed file-systems benefit from partial writes semantics (e.g. Coda [22], Echo [14]) since they usually offer locking granularity on a file, or even larger data units. Performing partial updates in version-based replica control protocols comes at the price of more complex algorithms. This is because only the most recent version of a data item can safely be replaced through a partial write in such protocols. We assumed total update semantics in the description above.

Replication granularity is orthogonal to locking granularity, and affects the state maintained per data item by the replica control protocol. At one extreme, replication granularity is the same as locking granularity. In this case, a d-space needs to be maintained individually for each object. The flexibility of selectively replicating objects at nodes (and thus allowing fine-grained control over the amount of replicated state), is achieved through high protocol state storage costs. At the other extreme, the granularity of replication is given by the set of all data items that the system manages. A node will replicate either all objects or none. Only one d-space needs to be maintained in this case for the whole database, file-system, etc., keeping the protocol's state overhead to a minimum. In between the two extremes, various trade-offs between protocol state and replication flexibility can be attained by replicating data items in group.

Note that for the case when the granularity of replication covers all data items, objects need not be effectively replicated at all sites. Instead, some of the nodes can maintain pointers to other nodes that actually store the replica. When a request arrives at a node holding a pointer, the request is forwarded to the node storing the replica. The cost of storing a replica is replaced by the cost of storing a pointer to it. However, node failures are correlated. When a node fails, all nodes that hold pointers to the failed node will logically fail as well. This also implies that node failure semantics changes. A node can be considered as failed for some data items (for which the node holds pointers to failed nodes), but operational for other data items (for which the nodes stores actual replicas).

4. AVAILABILITY ANALYSIS

We establish the availability of read and write operations as well as the availability of the last update. We assume that the network is reliable, node failures are both independent and fail-stop [23], and all nodes are identical. Let p be the probability of a node being operational, i.e. the node's availability. Note that since all nodes have the same availability, the optimal vote assignment for the voting protocol is one where all nodes are assigned one vote each [27]. The probability of finding m operational nodes among the N nodes is given by the binomial distribution:

$$b(N,m,p) = \begin{pmatrix} N \\ m \end{pmatrix} p^m (1-p)^{N-m} \qquad (3)$$

4.1 Data Availability

Data availability is expressed in terms of the probability that the last update of an item is accessible on at least one of the nodes that stores a copy. Since the last update is given by the last successfully executed write operation, at the time of the update $N^{\frac{d-1}{d}}$ sites have committed the new value. The probability that at least one of these sites will survive until the next update occurs is given by:

$$\alpha^D = 1 - (1-p)^{N^{\frac{d-1}{d}}} \qquad (4)$$

Note that data availability depends on the size of a write quorum, and thus implicitly captures the logical structure of a d-space. However, it does not depend directly on the particular organization of sites that is imposed by d-spaces. The same formula applies to any scheme that updates quorums of the given size (in this case $N^{\frac{d-1}{d}}$), including the voting approach, other logically structured quorums, etc.

4.2 Read Availability

The availability of read operations is the probability that the system allows the operation to perform successfully assuming no state changes while it is in progress. A read is successful if at least one line of $N^{\frac{1}{d}}$ sites can be accessed. A line is available with probability $p^{N^{\frac{1}{d}}}$, while m *selected* lines are available with probability:

$$\alpha_{line}(m) = p^{mN^{\frac{1}{d}}} \tag{5}$$

Knowing that there are $N^{\frac{d-1}{d}}$ potential lines to choose from, the availability of read operations is given by:

$$\begin{aligned} \alpha^R_{RFWM} &= 1 - (1 - \alpha_{line}(1))^{N^{\frac{d-1}{d}}} \\ &= 1 - (1 - p^{N^{\frac{1}{d}}})^{N^{\frac{d-1}{d}}} \end{aligned} \tag{6}$$

The availability of read operations in the Grid protocol is different than in RFWM even when considering a 2 dimensional replica space for which $N^{\frac{1}{d}} = N^{\frac{d-1}{d}}$ (since $d = 2$). The Grid protocol targets symmetrical scenarios for which the expected frequency of read and write operations is approximately the same ($\rho_{wr} \approx 1$). A write in Grid is performed similar to 2-spaces, by locking a column and a cover of all columns. However, a read is performed by locking the cover of all columns, as opposed to locking a column. The read availability is thus greatly improved in the $\sqrt{N} \times \sqrt{N}$ 2-dimensional space case and is given by [9]:

$$\alpha^R_{GRID} = (1 - (1-p)^{N^{\frac{1}{2}}})^{N^{\frac{1}{2}}} \tag{7}$$

Note that the replica control protocol advocated in Grid works well only for symmetrical spaces. The Grid analog for the RFWM approach would include in a read quorum a cover of a line, and in a write quorum a hyper-plane together with the cover of a line. This would improve the read availability over d-spaces, but would adversely affect write availability to the point of making the protocol unusable. To summarize, if the difference between read and write quorum sizes is substantial, quorum groups should be defined using the d-space approach, otherwise they should be defined using the Grid approach. For purposes of this study we are primarily interested in asymmetrical scenarios, and thus we will not establish the break-even point.

The read availability of the hierarchical quorum consensus is expressed recursively at each level. A logical node on level i is considered available if at least r_{i+1} of its l_{i+1} children are available. The availability of read operations, α^R_{HQC}, corresponds to the availability of the root node, while each leaf node is operational with independent probability p [16]:

$$\begin{aligned} \alpha^R_{HQC} &= \alpha^R_0 \\ \alpha^R_i &= \sum_{j=r_{i+1}}^{l_{i+1}} \binom{l_{i+1}}{j} (\alpha^R_{i+1})^j (1 - \alpha^R_{i+1})^{l_{i+1}-j} \\ \alpha^R_m &= p \end{aligned} \tag{8}$$

For reference we also present the read availability of the read-one write-all approach, and the weighted voting approach. [2]

4.3 Write Availability

The availability of write operations is the probability that the system allows the operation to succeed assuming no state changes while it is in progress. A write is successful if one line of $N^{\frac{1}{d}}$ nodes together with a cover of a hyper-plane of $N^{\frac{d-1}{d}}$ nodes can be accessed. Note that since the line already covers one node in the plane, only $N^{\frac{d-1}{d}} - 1$ nodes need to be covered. More generally, a write is successful if m lines and a cover of $N^{\frac{d-1}{d}} - m$ nodes are operational.

The cover of a hyper-plane of m nodes is available if there is at least one available node in each of the m corresponding lines. We further require that each such line not be fully available. At least one site, but not all of them, is available in a line with probability:

$$\begin{aligned} \alpha_{pointC} &= \sum_{m=1}^{N^{\frac{1}{d}}-1} b(N^{\frac{1}{d}}, m, p) \\ &= 1 - p^{N^{\frac{1}{d}}} - (1-p)^{N^{\frac{1}{d}}} \end{aligned} \tag{9}$$

The availability of a cover of a hyper-plane of m *selected* nodes is:

$$\begin{aligned} \alpha_{planeC}(m) &= (\alpha_{pointC})^m \\ &= (1 - p^{N^{\frac{1}{d}}} - (1-p)^{N^{\frac{1}{d}}})^m \end{aligned} \tag{10}$$

To compute the availability of write operations we add the probabilities for all combinations of m available lines (5) and $N^{\frac{d-1}{d}} - m$ additional nodes available in a cover (10):

$$\begin{aligned} \alpha^W_{RFWM} &= \sum_{m-1}^{N^{\frac{d-1}{d}}} \binom{N^{\frac{d-1}{d}}}{m} \alpha_{line}(m) \cdot \alpha_{planeC}(N^{\frac{d-1}{d}} - m) \\ &= (\alpha_{line}(1) + \alpha_{planeC}(1))^{N^{\frac{d-1}{d}}} - \alpha_{planeC}(N^{\frac{d-1}{d}}) \\ &= (1 - (1-p)^{N^{\frac{1}{d}}})^{N^{\frac{d-1}{d}}} \\ &\quad - (1 - p^{N^{\frac{1}{d}}} - (1-p)^{N^{\frac{1}{d}}})^{N^{\frac{d-1}{d}}} \end{aligned} \tag{11}$$

The write availability of the hierarchical quorum consensus approach, α^W_{HQC}, is expressed recursively and is similar to the read availability of the same method (8), except that the series $r_1, r_2 \ldots r_m$ is used instead of $w_1, w_2, \ldots w_m$ when defining α^W_i. For reference we also present the write availability of the read-one write-all approach, and the weighted voting approach. [3]

[2]Read availability for ROWA: $\alpha^R_{ROWA} = 1 - p^N$. Read availability for weighted voting assuming one vote per node: $\alpha^R_{VOT} = \sum_{m=v_r}^{N} b(N, m, p)$.

[3]Write availability for ROWA: $\alpha^W_{ROWA} = p^N$. Write availability for weighted voting assuming one vote per node: $\alpha^W_{VOT} = \sum_{m=v_w}^{N} b(N, m, p)$.

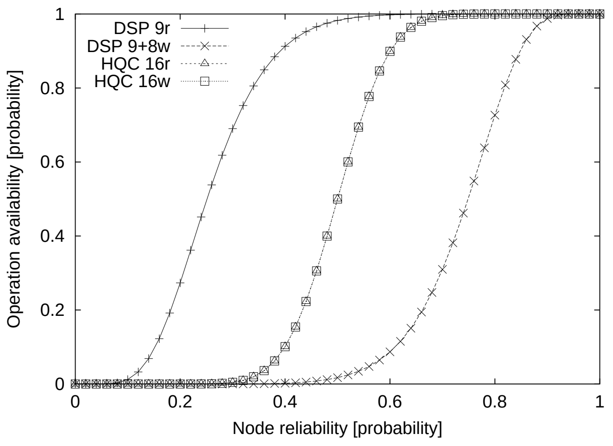

Figure 2: Operation availability for DSP and HQC with 81 nodes and targeted access ratio $\rho_{wr} = 1$.

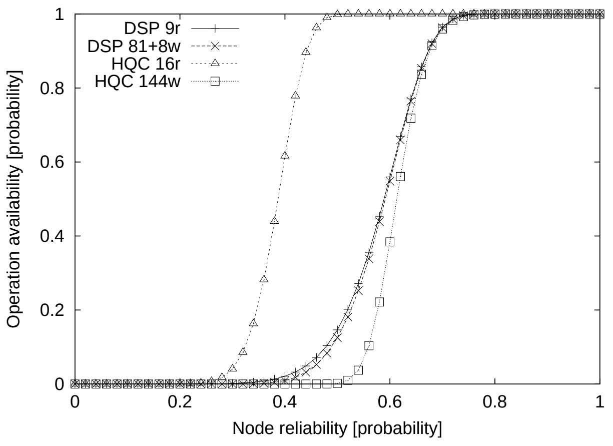

Figure 3: Operation availability for DSP and HQC with 729 nodes and targeted $\rho_{wr} = 9$.

4.4 D-Space vs. Hierarchical Quorums

We compare the communication complexity and operation availability of d-spaces (referred to as DSP) and the hierarchical quorum consensus methods (referred to as HQC). We will examine both symmetrical scenarios, where the access ratio is close to 1, and skewed scenarios, where read operations dominate accesses to data. The results shown correspond to formulas derived in the previous sections.

Hierarchical quorum consensus is optimal when each logical group is decomposed in three subgroups, and the resulting hierarchy is perfectly balanced. We choose the number of nodes with respect to such criteria, i.e. $N = 3^m$ ($l_i = 3$). The quorum intersection condition is satisfied if (i) $r_i + w_i > 3$, and (ii) $2w_i > 3$. Possible combinations for establishing appropriate quorums at each node are ($r_i = 1, w_i = 3$) and ($r_i = 2, w_i = 2$). Thus, a read quorum will contain $1^t 2^{m-t}$ sites, and a write quorum will contain $3^t 2^{m-t}$ sites, for some t such that $0 \le t \le m$ and $3^t \approx \rho_{wr}$.

We distribute the quorums at each level such that operation availability is maximized. More exactly, we strive for optimal write availability since they are the critical component with respect to failure (i.e. consistently having lower availability than read operations). It is beyond the scope of this paper, but it can be shown that availability for write operations, given the constraints above, is achieved when the top levels have $w_i = 3$ ($1 \le i \le t$), and lower levels have $w_j = 2$ ($t < j \le m$), and vice-versa.

4.4.1 Operational Availability

Figure 2 shows read and write availability as a function of site reliability for d-spaces and hierarchical quorum consensus. 81 nodes are considered, arranged in a 3^4 hierarchy for HQC, and a 9×9 2-dimensional space for DSP. This arrangement captures balanced read and write access frequencies ($\rho_{wr} = 1$). In this case, the read availability for d-spaces is derived using the Grid quorum definitions and given by (7). Read and write availability curves for HQC overlap due to the symmetry of read and write quorum definitions for $\rho_{wr} = 1$. Operation availability in d-spaces for a 9×9 space is similar to the Grid protocol and is highly

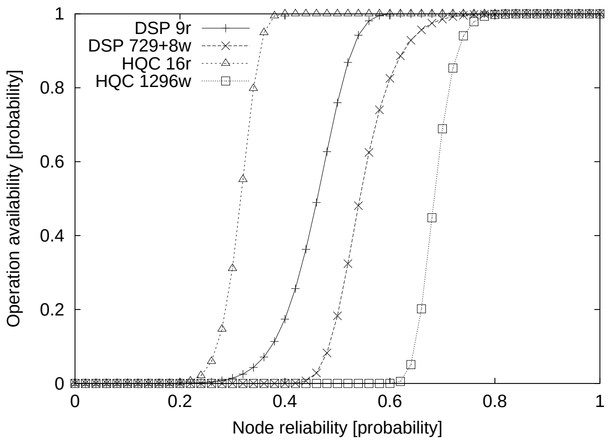

Figure 4: Operation availability for DSP and HQC with 6,561 nodes and targeted $\rho_{wr} = 81$.

asymmetrical. This in fact is considered to be the main argument against using the Grid protocol for symmetrical access patterns.

In Figures 3–5 we present operation availability in HQC and DSP for increasing access ratios, $\rho_{wr} = 9$, $\rho_{wr} = 81$, and $\rho_{wr} = 729$. In Figure 3, 729 nodes are considered arranged in a 3^6 hierarchy for HQC, and a $9 \times 9 \times 9$ 3-dimensional space for DSP. Similarly, 6,561 (59,049) nodes are considered in Figure 4 (Figure 5) arranged in a 3^8 (3^{10}) hierarchy for HQC, and a 4 (5)-dimensional space for DSP.

Starting with $\rho_{wr} = 9$, HQC and DSP change places as far as availability symmetry for read and write operations is concerned. For $\rho_{wr} = 9$, DSP is almost perfectly balanced, for $\rho_{wr} = 81$ is well balanced, and less balanced for $\rho_{wr} = 729$. HQC is consistently less balanced than DSP. Note however that with the exception of Figure 3, in all three other figures the two approaches are equivalent if we abstract over the degree of asymmetry, i.e. if one gains for read availability it loses the same amount for write availability.

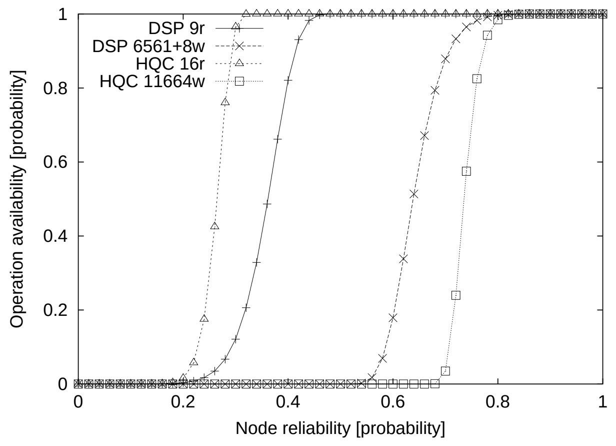

Figure 5: Operation availability for DSP and HQC with $59,049$ nodes and targeted $\rho_{wr} = 729$.

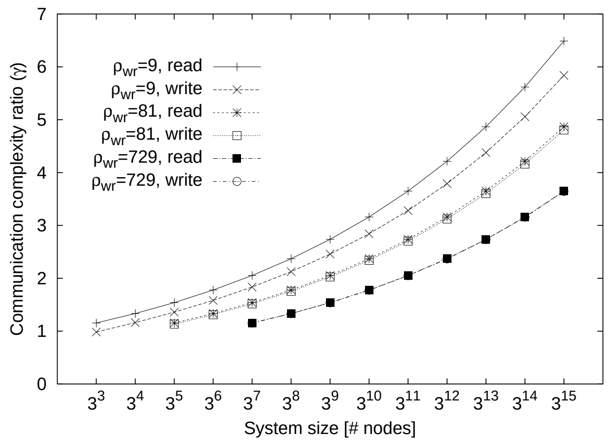

Figure 6: Communication complexity ratio (HQC/DSP) for read and write operations as function of system size and access ratio.

If the overall availability is not affected, the more symmetrical read and write availabilities are the, more desirable a method is deemed. In particular, despite its read efficiency and good availability, the read-one write-all protocol is considered unsatisfactory due to its poor write availability. Implementing the read-few write-many ($\rho_{wr} \gg 1$) approach using d-spaces will result in more balanced, and thus better, operation availability than using the hierarchical quorum consensus method.

4.4.2 *Communication Complexity*

We showed that d-space quorum sets have optimal message complexity for any read/write access ratio. We now show how communication costs for the hierarchical quorum consensus relate to d-spaces. A read in HQC has communication cost $1^t 2^{m-t}$, where $3^t = \rho_{wr}$ and $3^m = N$. The same ρ_{wr} is obtained in DSP by defining read quorum groups of size $\sqrt{N/\rho_{wr}} = \sqrt{3^{m-t}}$. The cost ratio for read operations in HQC versus DSP is given by:

$$\begin{aligned}\gamma^R = \frac{2^{m-t}}{\sqrt{3^{m-t}}} &= \left(\frac{N}{\rho_{wr}}\right)^{\log_9\left(\frac{4}{3}\right)} \\ &\approx \left(\frac{N}{\rho_{wr}}\right)^{0.13}\end{aligned} \tag{12}$$

Note that for $\rho_{wr} = 1$, the ratio becomes $N^{0.63}/\sqrt{N}$ which is exactly the ratio of quorum sizes for HQC and DSP in the symmetric case. Starting with write quorum sizes in HQC ($3^t 2^{m-t}$), and DSP ($\sqrt{N/\rho_{wr}} + \sqrt{N\rho_{wr}} - 1 \simeq \sqrt{3^{m-t}} + \sqrt{3^{m+t}}$), the corresponding cost ratio for write operations is:

$$\begin{aligned}\gamma^W &= \frac{3^t 2^{m-t}}{\sqrt{3^{m-t}} + \sqrt{3^{m+t}}} = \frac{3^t}{3^t + 1}\left(\frac{N}{\rho_{wr}}\right)^{\log_9\left(\frac{4}{3}\right)} \\ &= \frac{\rho_{wr}}{\rho_{wr} + 1}\gamma^R\end{aligned} \tag{13}$$

For all four figures shown, $N/\rho_{wr} = 81$ and thus $\gamma^R = 1.78$, while γ^W takes values 0.94, 1.62, 1.76 and 1.78, depending on the value of ρ_{wr}. Note that for a given ρ_{wr}, the ratio of communication costs for both read and write operation grows with N. In Figure 6 we show how γ^R and γ^W vary with system size for some of the used access ratios (9, 81, and 729).

Implementing the read-few write-many approach using d-spaces will result in a message complexity 2–3, times lower than implementing it using the hierarchical quorum consensus method. Beside saving network resources this also materializes in better load sharing at sites holding copies, and increased system throughput. The expected increase in system throughput is proportional to γ^R for read operations, and γ^W for write operations.

5. RECONFIGURATION AND FAULT TOLERANCE

The availability study carried out previously establishes the likelihood that executing read and write operations will commit by assembling all the sites in a quorum group. A client executing an operation must assemble one quorum group from the operation's quorum set. If no quorum group can be assembled, the corresponding operation is blocked and the client that issued the operation will not be able to make further progress, nor will any other client submitting subsequent requests (assuming no recoveries occur).

Structured quorum protocols rely on pre-defined identity for participating sites (given by the assignment of sites to quorum groups), and are thus less tolerant to failures than voting protocols. Voting protocols are more adaptable through reconfiguration because they define quorums based on the number of votes, regardless of the identity of the votes. In this section we discuss how reconfiguration can be performed on logical structures, and address related issues such as replica recovery, sites leaving or joining the network, and adapting quorums to profiled behavior.

5.1 Global Reconfiguration

Assuming that failed copies never recover, one may be tempted to allow operations to make progress as long as all available copies (as opposed to all copies) in a quorum group are properly locked. However, one-copy serializability follows from ordinary serializability only if two conflicting transactions that have conflicting accesses to a data item, have conflicting accesses to some copy of the data item [8]. If copies that could expose the conflict have failed, and are not included in the quorum in which they would otherwise be included, illegal executions (with respect to one-copy serializability) can occur.

5.1.1 Deadlock Prevention

Mechanisms are needed to cope with deadlock situations that arise when no quorum groups can be assembled. The necessary and sufficient condition for a deadlock to occur is that the set of failed sites include at least one quorum group. If a read quorum group has failed, no write quorum group can be assembled since all write groups have at least one site in common with the failed group. In this case all write accesses are blocked. Similarly, if a write quorum group has failed, no read or write quorum group can be assembled and all accesses are blocked.

Unfortunately there is no viable solution to address deadlocks caused by quorum group failures. Reconfiguring available sites after deadlock detection can break the protocol's correctness. For instance, if the failed quorum group recovers after reconfiguration, it may serve stale data to requesting clients unaware of the reconfiguration. In effect, this is equivalent to a partitioning of the initial replica set in two or more functional subsets, which is clearly undesirable.

However, deadlocks can be prevented by dynamically and transparently adjusting quorum groups to reflect failures and repairs in the system. In dynamic voting [15] quorum readjustment is performed within the protocol for write operations. Other approaches such as the virtual partition algorithm [1], and the epoch protocol [19, 20] check for changes in system topology or even reconfigure it as a separate transaction, asynchronously with regard to read and write operations. We briefly describe the epoch protocol as it can readily be applied to reconfigure d-space quorums.

5.1.2 Reconfiguring in Epochs

The epoch protocol [19, 20] infers a structure for the network, based on a rule and a total ordering of participating sites. This means that given any set of sites we can construct a logical structure (a d-space in this case) without recurring to a static mapping based on site identities. An *epoch* is a set of operational sites such that each site in the epoch knows about the availability of others. The current epoch describes the current state of the system. Initially, all replicas of the data item form the current epoch. Epoch checking is run periodically to poll all replicas in the system. If members of the current epoch are not accessible, or any replicas outside the current epoch have been successfully contacted, an attempt is made to form a new epoch.

The new epoch must contain a write quorum of the previous (current) epoch, and the set of new epoch members along with the new epoch number is recorded on every member of the new epoch. If network partitions occur, the intersection property of quorums guarantees that the attempt to form a new epoch will be successful in at most one partition. Further, a read or write operation must contact at least one member of the current epoch and therefore obtain the current epoch set. The transition from one epoch to another (quorum re-adjustment) need not be performed with every change in the topology, i.e. with every site failing. This gives failing sites the opportunity to recover and will reduce the number of adjustments performed over time. However, the checking and reconfiguring procedures should be aggressive enough that deadlock conditions entailed by quorum group failures are avoided.

5.2 Local Reconfiguration

We present an alternative to global reconfiguration protocols. The method defines the replica space independent of the set of sites holding the physical copies. Local connectivity is combined with a best-effort form of global views to achieve seemingly contradicting goals: local reconfiguration policies and good global latencies.

5.2.1 Global View vs. Local Connectivity

The fact that logically-structured quorum approaches, are not accommodating to mutations (reconfiguring the space, adding or removing nodes individually or in group) is inherent to its global view of a logical structure. Flexibility is sacrificed for the sake of efficiency. At the other extreme, similar structures are maintained by the CAN protocol to route among participants in peer-to-peer networks [21]. CAN lacks a global view and uses local connectivity (each sites knows its $2d$ immediate neighbors) to arrange the sites in d-dimensional tori.

Reconfiguration is localized in CAN, making the approach more flexible to respond to mutations in network topology. However latency is hurt as an isolated remote contact requires $O(dN^{\frac{1}{d}})$ incremental hops toward the destination. The latency to access a quorum group with local connectivity is given by the spatial diameter of the space defining the group, i.e. $N^{\frac{1}{d}}$ for reads and $N^{\frac{d-1}{d}}$ for writes. Thus, even though message complexity is the same irrespective of implementation choice (global view or local connectivity), the high latency of the latter makes it prohibitive as a support mechanism for implementing structured quorums.

We propose a solution that combines the advantages of global views and local connectivity. Local connectivity is used in the same spirit as in CAN. However, we do not advocate implementing d-spaces on top of CAN as the latter is too heavyweight for purposes of a replica control protocol, and does not integrate in its failure detection protocol locking revocation mechanisms.

5.2.2 Mapping the Replica Space

First, we make a distinction between the replica (node) space, and the set of sites hosting physical copies. The replica space is a very large space such that the number of sites will never match the number of nodes, and will feature high dimensionality. For instance $N = 2^{32}$, fixed ($d = 32$). Note that being able to choose arbitrary values for N (in particular 2^{32}), allows us to easily account for any access ratio ρ_{wr}, within a factor of 2.

The node space is equitably divided and mapped onto the sites such that each site holds a contiguous subspace of it (range mapping). This is no different than CAN, and we advocate a similar protocol for performing zone-reassignment in the background to prevent uneven fragmentation of the node space. Note however that the coordinate space in CAN is continuous, whereas our replica space is discrete and contains exactly N points (nodes) arranged in a multi-dimensional grid.

All nodes mapped to the same site correspond to one physical copy. In particular, their values and version numbers are kept consistent at all times: if a node is updated, the new image is reflected instantaneously at all other nodes mapped to the same site. Unfortunately, despite the fact that site failures may be independent, node failures are not due to the range-mapping of nodes to sites. Either all nodes mapped to a site are operational, or none of them. However, if the number of nodes per site is approximately the same, and the zone of nodes mapped to any site forms a regular subspace, then the availability relations deducted in Section 4 and the communication complexity of the protocol will hold.

5.2.3 Joining and Leaving the Structure

Second, we achieve flexibility to mutations through local connectivity. Each node has $2d$ neighbors in the replica space. Each site has as many neighboring sites as given by the mapping of nodes to sites. Given an even and regular (along axes) mapping, each site needs to maintain approximately $2d$ links to neighboring sites. When sites join or leave the structure, we use a protocol similar to CAN's for routing through the grid, and for splitting and merging the zones (subspaces) assigned to affected sites.

Upon joining no data transfer is necessary. The newly joined site will simply initialize the version numbers for its assigned data items to null values. Upon leaving data transfer is necessary only if the recipient's versions are smaller than the leaving site's versions. Thus, even though the replica space is static, the mapping of nodes to sites and the local connectivity allows us to support arbitrary mutations of the network through local reconfigurations.

5.2.4 Caching Quorum Groups

Third, we achieve the good latency of global views by employing a relaxed form of globality through caching. Every site caches a write quorum. Since a write quorum includes a read quorum (the line for RFWM), sites implicitly cache a read quorum as well. Caching a quorum translates to caching a list of mappings from subspace ranges to site identities (a site's identity can be the tuple of its network address and port number). When a read or write request arrives at a site, the corresponding quorum cached at the site will be used. Some of the mappings cached for a quorum group may be stale due to zone reassignments. Stale mappings are detected by timing out, or by confronting the cached mapping with the actual mapping.

If a stale mapping is detected, the correct mapping for the corresponding subspace range will be identified by performing CAN-like routing. This procedure can be optimized. Routing to the zone of interest can be started at the closest zone in the cached quorum whose mapping has just been validated.

Stale cached quorums do not affect correctness. The protocol's performance would however degrade due to repeated routings to refresh mappings, most notably when the rate of site joining/leaving is high relative to the rate of requests. Note that global reconfiguration protocols (e.g. the epoch protocol) also incur considerable overhead when the topology of the network changes rapidly. Furthermore, a write quorum must be available for global reconfiguration protocols to successfully reconfigure the structure.

A protocol is needed for combining quorum groups cached at different sites. Read and write locking may timeout on a faulty site before the site recovers, or is declared failed and evicted. User requests should not be delayed by transient failures. Thus, the site that experiences a timeout when using its cached quorum group may want to replace the offending mappings from the group. If the timeout occurred on a read, then the whole quorum group needs to be replaced since read quorums form a partition of the replica space. If the timeout occurred on the cover component of a write, only mappings that timed-out must be replaced. This property is highly convenient since replacing the whole cover component of a cached write quorum may prove to be expensive. Additionally, read quorums can be replaced, or the cover components of write quorums can be shuffled, at any time to ensure a more even distribution of mappings in caches. A site can use its neighbors or live sites in its cached quorum groups to perform quorum shuffling with.

5.2.5 Fault Tolerance

Local connectivity enables sites to join and leave the structure without invalidating existing quorums (by creating holes in the structure), or requiring a global reconfiguration mechanism. Faulty sites that do not recover (or recover too late) can also be eliminated through local reconfiguration. We informally describe a failure detection protocol that ensures the proper transfer of zones and locks from the site declared failed to one of its operational neighbors.

They key of the protocol lies in enforcing that a site declares itself failed (upon recovery) before any of its neighbors do so. We assume the following timeouts are in effect. τ_{access} is the timeout of an individual read or write access. A lock grab/release request will timeout after τ_{access}, and the requester will try to assemble other quorum groups. τ_{oper} is the timeout of a read or write operation. If a site has granted its lock to a requester and the lock hasn't been released in τ_{oper}, it is by default considered to be released.

Each site exchanges periodic heartbeat messages with all its neighbors. We assume that incommunicado states are symmetric: sites can either contact each other both ways or none. A site will declare itself failed after not hearing from any of its neighbors for at least τ_{self}. The period for heartbeat messages should be much smaller than τ_{self} to allow for a few exchanges to occur within τ_{self}. A site will tentatively declare its neighbor failed after not hearing from it for at least τ_{fail}. When a site declares one of its neighbors failed, it will contact other neighbors of the failed site to concur on the decision, and evict the failed site from the structure if necessary.

We have the following timing constraints: $\tau_{access} \ll \tau_{oper} \ll$

$\tau_{self} < \tau_{fail}$. We have that $\tau_{access} \ll \tau_{oper}$ to give enough time the initiator to complete an operation while allowing it to timeout on individual accesses. Naturally, $\tau_{oper} \ll \tau_{self}$ since we want to avoid blocking on operations for the whole length of time that a transient failure is allowed. Finally, we have that $\tau_{self} < \tau_{fail}$ because we want to ensure that a site will declare itself failed before its neighbors will. τ_{fail} can be chosen to be $2\tau_{self}$.

If a site fails and recovers before τ_{self} elapses, the failure is considered transient. A nice feature of quorum consensus schemes is that recovering copies require no special treatment. A recovering copy that missed writes, when included in a quorum group will have its version number smaller than some other copy in the same group that took part in the writes. Thus, its value will not be read until written. The only condition that breaks the invariant above is if a whole quorum group fails, misses some writes, and then recovers. This is an impossible condition since (i) either no writes could have taken place due to the failed quorum group, or (ii) if writes took place, the recovering quorum must have been evicted from the structure through reconfiguration.

If a site fails and recovers after τ_{self} elapses, or does not recover at all, the failure is considered permanent. Sites that fail permanently and recover subsequently must join the structure anew before further processing requests. Since $\tau_{oper} \ll \tau_{self}$, we are guaranteed that the lock of a permanently failing site has been released by the time the site declares itself failed. Since $\tau_{self} < \tau_{fail}$, we are guaranteed that the lock has been released by the time the site is evicted from the structure by its neighbors. Thus, the lock can safely be transferred to the neighbor that will take over the failing site's subspace. The data and associated versions hosted by the failing site are lost and cannot be recovered. To ensure consistency, the neighbor that takes over the subspace and the lock will need to perform a read operation for the reassigned data items.

5.2.6 Profiling the Access Ratio and Adaptation

Quorum groups in d-spaces are defined given a targeted read/write access ratio (ρ_{wr}). It is conceivable that many applications of interest exhibit identifiable data access patterns and further, that such patterns may change with time, based on user input or other conditions. Previous work addresses to some extent dynamically reconfigurable quorums by developing protocols that adapt to changes in the environment [15, 19, 3].

In our context, the reconfiguration mechanism can be extended to model the current access pattern. Each site locally profiles the observed access ratio and establishes an average with its neighbors and other sites it communicates with. When the profiled ratio is consistent and substantially different than the one in effect, a global reconfiguration transaction (e.g. the epoch protocol) is initiated to redistribute the read/write quorum sizes. At least a write quorum needs to be part of the reconfiguration process. Reconfiguring in this sense preserves the replica space and the mapping of nodes to sites. However, upon successful reconfiguration, all quorums cached at sites will be invalidated with a transient performance penalty.

6. CONCLUDING REMARKS

We have described a method for defining replica quorum groups by arranging participating sites in logically structured multi-dimensional spaces. The protocol is a generalization of the Grid protocol to multiple dimensions. It is also reciprocal to Grid in the sense that it defines inverted read and write quorum groups with respect to subspace covers. The central argument of our study is that d-space quorums offer a flexible way to build protocols with ideal balances of low communication complexity and high availability.

First, d-spaces allow the more frequent read operations to execute efficiently, at a limited and controllable expense of more rarely executed write operations. We have shown that the communication complexity of the protocol is optimal for a given read/write access ratio. Second, and more importantly, read operations can be performed efficiently without adversely affecting the availability of updates. To our knowledge, the quality of trade-off between read efficiency and update availability of our approach is not matched by existing quorum protocols. Surprisingly, for highly skewed read/write access ratios the availability of updates can approximate or even match the availability of read operations.

Existing structured quorum schemes are based on static global views. This achieves good communication latencies but hurts the protocol's adaptivity as it relies on global reconfiguration mechanisms. For d-spaces we propose implementing a combination of local connectivity on multi-dimensional spaces and a relaxed form of global views (cached quorums). Future work will establish whether the ability to locally adapt to membership changes (joins, leaves) and site failures can offset the performance penalty incurred by stale cached mappings.

Our work has been motivated by the commonly accepted observation that read accesses to data occur orders of magnitude more often than updates. We believe that read accesses will be a defining characteristic of usage in future distributed systems, including peer-to-peer architectures. Presently deployed file-sharing utilities fit this description, with the vast majority of data exported by participants being unaltered since creation. The read efficiency of the read-few write-many approach combined with its good write availability make it a good candidate for implementing replica control protocols for databases, file-systems, and peer-to-peer networks.

7. REFERENCES

[1] A. El Abbadi and S. Toueg. Maintaining availability in partitioned replicated databases. *ACM Transactions on Database Systems*, 14(2):264–290, June 1989.

[2] D. Agrawal and A. El Abbadi. An efficient solution to the distributed mutual exclusion problem. In *Proc. of the* 8^{th} *ACM Symposium on Principles of Distributed Computing*, pages 193–200, Edmonton, Alberta, Canada, June 1989. ACM Press.

[3] D. Agrawal and A. El Abbadi. Using reconfiguration for efficient management of replicated data. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):786–801, October 1996.

[4] M. Ahamad, M. Ammar, and S. Cheung. Replicated data management in distributed systems. In T. L. Casavant and M. Singhal, editors, *Readings in Distributed Computing Systems*, pages 572–591. IEEE Computer Society Press, Los Alamitos, CA, January 1994.

[5] M. Ahamad, M. H. Ammar, and S. Y. Cheung. Multidimensional voting. *ACM Transactions on Computer Systems*, 9(4):399–431, November 1991.

[6] P. A. Bernstein and N. Goodman. An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Transactions on Database Systems*, 9(4):596–615, December 1984.

[7] P. A. Bernstein and N. Goodman. A proof technique for concurrency control and recovery algorithms for replicated databases. *Distributed Computing*, 2(1):32–44, January 1987.

[8] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in database systems.* Addison-Wesley, Boston, MA, 1987.

[9] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The Grid protocol: A high performance scheme for maintaining replicated data. *IEEE Transactions on Knowledge and Data Engineering*, 4(6):582–592, December 1992.

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

[11] D. K. Gifford. Weighted voting for replicated data. In *Proc. of the 7th Symposium on Operating Systems Principles*, pages 150–162, Pacific Grove, CA, December 1979.

[12] N. Goodman, D. Skeen, A. Chan, U. Dayal, S. Fox, and D. Ries. A recovery algorithm for a distributed database system. In *Proc. of the 2nd ACM Symposium on Principles of Database Systems*, pages 8–15, Atlanta, GA, March 1983.

[13] M. Herlihy. A quorum-consensus replication method for abstract data types. *ACM Transactions on Computer Systems*, 4(1):32–53, February 1986.

[14] A. Hisgen, A. Birrell, C. Jerian, T. Mann, M. Schroeder, and G. Swart. Granularity and semantic level of replication in the Echo distributed file system. In *Proc. of the IEEE Workshop on Management of Replicated Data*, pages 2–4, Houston, TX, November 1990.

[15] S. 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.

[16] A. Kumar. Hierarchical quorum consensus: A new algorithm for managing replicated data. *IEEE Transactions on Computers*, 40(9):996–1004, September 1991.

[17] L. Lamport. Time clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[18] M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2):145–159, May 1985.

[19] M. Rabinovich and E. D. Lazowska. Improving fault-tolerance and supporting partial writes in structured coterie protocols for replicated objects. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 226–235, San Diego, CA, June 1992.

[20] M. Rabinovich and E. D. Lazowska. Asynchronous epoch management in replicated databases. In *Proc. of the 7th International Workshop on Distributed Algorithms*, pages 115–128. Springer-Verlag, 1993.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proc. of the ACM SIGCOMM*, San Diego, CA, August 2001.

[22] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D. C. Steere. Coda: A highly available file system for a distributed workstation environment. *IEEE Transactions on Computers*, 39(4):447–459, 1990.

[23] R. D. Schlichting and F. B. Schneider. Fail-stop processors: an approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems*, 1(3):222–238, August 1983.

[24] M. Stonebraker. Concurrency control and consistency of multiple copies of data in distributed INGRES. *IEEE Transactions on Software Engineering*, 5(3):188–194, May 1979.

[25] R. H. Thomas. A solution to the concurrency control problem for multiple copy databases. In *Proc. of the 16th IEEE Computer Society International Conference*, New York, NY, Spring 1978.

[26] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2):180–209, June 1979.

[27] Z. Tong and R. Y. Kain. Vote assignments in weighted voting mechanisms. In *Proc. of the 7th Symposium on Reliable Distributed Systems*, pages 138–143, Columbus, OH, October 1988.