Dynamic Logical Structures: A Position Statement for Managing Replicated Data*

D. Agrawal A. El Abbadi
Department of Computer Science
University of California
Santa Barbara, CA 93106

1 Introduction

Recently, several researchers have proposed imposing a logical structure on the set of copies in a replicated database, and using structural information to create intersecting quorums. Protocols that use a logical structure, e.g., the grid protocol [CAA90] and the tree protocol [AE90], execute operations with low communication costs while providing fault-tolerance for both read and write operations. However, as with the original read-one write-all approach, the improved performance for read operations results in a degraded write availability when failures occur. In particular, structure-based protocols are vulnerable to the failure of specific sites. In this sense, these protocols are analogous to the read-one write-all protocol, i.e., low cost read operations but write operations are vulnerable to failures. In this position statement, we first discuss extensions to the grid protocol to improve the fault-tolerance of write operations by using the notions of structured read and write grid quorums. As is the case in the standard quorum protocols [Giff79, Tho79], the increased fault-tolerance for write operations is at the increased cost of executing read operations. In order to let users continue using the analogues of the read-one write-all protocol with respect to a logical structure, we develop reconfiguration protocols for dynamically adapting to failures and recovery. This results in the following dichotomy. Users accesses are through the simple analogues of the read-one write-all protocol with respect to a logical structure and therefore have low communication cost for read operations. On the other hand, the reconfiguration protocol uses the notion of quorums in the context of a logical structure to ensure high data availability. A similar approach can be applied to the tree protocol [AE92].

In a replicated database, copies of an object may be stored at several sites in the network. A read is executed by reading a read quorum of copies, and a write is executed by writing a write quorum of copies. The notion of intersecting quorums is closely related to coteries [GB85].

*This research is supported by the NSF under grant numbers IRI-9004998 and IRI-9117904.

The following restriction is placed on the choice of quorum assignments [Giff79, Tho79]:

- Quorum Intersection Property: For any two operations $o[z]$ and $o'[z]$ on an object $z$, where at least one of them is a write, the quorums must have a nonempty intersection.

The view-based protocol uses configuration information to achieve fault-tolerance of both read and write operations, while reducing the cost of executing read operations. Reconfiguration is initiated through a special transaction called a view-reconfiguration transaction. The reconfiguration transaction updates copies and configurations in an atomic step. The transaction succeeds in including an object in a new view only when it can determine the current value of that object. In order to ensure that the new view has the current value, the reconfiguration transaction must access a reconfiguration quorum which satisfies the following property:

- Reconfiguration Quorum Property: A reconfiguration quorum must have a nonempty intersection with any write quorum and two reconfiguration quorums must have a nonempty intersection.

The above rules in addition to a concurrency control protocol ensure one-copy serializability [ET89].

2 Reconfigurable Grids

We start by generalizing the notion of quorums in the context of the grid structure. In general, a grid quorum has a length $l$ and a width $w$, and is represented as a pair $(l, w)$. A read operation is executed by accessing a read grid quorum $(l, w)$, which is formed from $l$ copies in each of $w$ different columns. A write operation is executed by writing a write grid quorum, $(l, w)$, which is formed from (a) $l$ copies in each of $w$ columns as well as (b) any $\sqrt{w} - l + 1$ copies in each of $\sqrt{w} - w + 1$ columns. (The second component of the write quorum is to ensure the intersection between

---

*The latter part of the condition is relaxed in [ET89].
two write quorums, if timestamps are used instead of version numbers, this component is not needed.) In order to ensure the quorum intersection property between read and write quorums, if the read grid quorum is of size \((l, w)\) then the write grid quorum must be \((\sqrt{n} - l + 1, \sqrt{n} - w + 1)\).

Although the grid quorum protocol achieves high data availability, increase in fault-tolerance is obtained at the increased cost of read operations. The reconfiguration approach can be used to increase fault-tolerance while maintaining low communication costs. After reconfiguration the \(\sqrt{read/\bar{write}}\) assignment i.e., read quorums have dimensions \((1, \sqrt{n}/2 + 1)\) and write quorums have dimensions \((\sqrt{n}/2, \sqrt{n}/2 + 1)\), which is used for constructing read and write grid quorums. For reconfiguration we use the majority grid quorum assignment, i.e., grid quorums with dimension \((\sqrt{n}/2 + 1, \sqrt{n}/2 + 1)\), which provides the highest degree of fault-tolerance (in general the majority assignment provides the best data availability [AA89] when copies have equal weights).

We now summarize the rules. Assume a grid of dimensions \(\sqrt{n} \times \sqrt{n}\), and that there are \(l\) available columns and \(w\) available rows in the current view (a row or a column is available if it contains at least one copy in the current view):

- **Write Quorum Rule**: A write quorum is formed from all available copies in \(\sqrt{n}/2 + 1\) columns. Each available column included in the write quorum must have at least \(\sqrt{n}/2 + 1\) copies.
- **Read Quorum Rule**: A read quorum is formed from one copy in any \(l - \sqrt{n}/2\) of the \(l\) available columns.
- **Reconfiguration Rule**: A reconfiguration quorum is formed from any majority grid quorum of dimensions \((\sqrt{n}/2 + 1, \sqrt{n}/2 + 1)\).

Consider a scenario in which all copies except a quarter of the grid are unavailable as shown in Figure 1. A reconfiguration transaction can still form a new view by accessing a reconfiguration quorum consisting of the available copies \([13,14,15,18,19,20,23,24,25]\). Write operations in this view will write all copies whereas read operations access any single copy. This example is one of the many possible failure scenarios in which as many as 3/4th of the total number of copies become unavailable and yet read and write operations can be executed. No other variants of the static quorum protocol (where all copies are treated equal) can tolerate as many failures.

Figure 2(a) illustrates the availabilities in the majority quorum, read-one write-all, \(\sqrt{read/\bar{write}}\) [CAA90], and the grid reconfiguration protocols when there are nine copies of an object that are organized as a grid with three columns and three rows. Similarly, Figure 2(b) illustrates the availability with twenty-five copies that are organized as a grid with five columns and five rows. Figure 2 clearly shows the imbalance between read and write availability when the read-one write-all protocol is used. This imbalance is partially remedied by the \(\sqrt{read/\bar{write}}\) protocol due to the use of a logical structure. However, as the figure shows, there still remains a significant imbalance between read and write availabilities. For example with 25 copies and when an individual copy has availability 75%, read availability is 100% whereas write availability is approximately 74%. The quorum protocol and the proposed reconfiguration based protocol in this case have read and write availability of approximately 99%. This analysis illustrates that by using the reconfiguration approach data availability in the grid protocol can be made comparable to the quorum protocol while maintaining the low cost of read operations. In particular, read operations in the grid protocol access fewer than \(\sqrt{n}/2\) copies compared to \(n/2\) in the quorum protocol. Note that with 9 copies, the availability of both operations in the quorum and reconfiguration based protocol becomes almost 100% when the availability of an individual copy is greater than 85%. Similarly, with 25 copies, this availability level is attained beyond copy availability of 75%. In this sense, we are able to achieve the high degree of read availability of the read-one write-all protocol for both read and write operations using the reconfiguration based approach. This is achieved without the high cost of the read operations required by the quorum protocol.

### 3 Reconfigurable Trees

The development of a reconfiguration based protocol for grids can also be applied to tree structures [AE92]. For lack of space we only illustrate the availabilities in the context of the tree structures. Figure 3 illustrates the availabilities of read and write operations in the majority quorum, read-one write-all, ReadRoot [AE90] and the tree reconfiguration protocols. We again observe that there is an imbalance in availability between read and write operations that use the ReadRoot protocol without reconfiguration. For example with 13 copies and copy availability of 75%, read availability is almost 99% whereas write availability is approximately 50%. With reconfiguration, read and write availability in this case become approximately 92%. For copy availability greater than 85% the data availability becomes approximately 100%. Read and write operations, on the other hand, have costs comparable to the ReadRoot protocol. This analysis illustrates that by using the reconfiguration approach, data availability in the tree protocol can be made comparable to the quorum protocol while maintaining the low cost of read operations.

### 4 Discussion

The proposed approach provides comparable data avail-
Figure 1: A grid organization with 1/4 of the copies available

Figure 2: Comparison of the availabilities in various protocols
ability to the majority assignment, while significantly reducing the cost of executing read operations. Reconfiguration has the added cost of the view reconfiguration transaction, which is executed whenever reconfiguration is required. Logical structures, however, provide a limited degree of fault-tolerance for write operations. Hence, for temporary, short-term failures, no reconfiguration needs to be performed, and most read and write operations will be allowed to execute. Thus, reconfiguration will only need to be performed when failures persist and result in degraded availability for both read and write operations. In [AE92] we have applied the same general techniques to reconfigure tree-based protocols.

References


