Quorum Systems[*]

Dahlia Malkhi
AT&T Labs-Research

March 23, 1999

Quorum systems are tools for increasing the availability and efficiency of replicated services. A *quorum system* for a universe of servers is a collection of subsets of servers, each pair of which intersect. Intuitively, each quorum can operate on behalf of the system, thus increasing its availability and performance, while the intersection property guarantees that operations done on distinct quorums preserve consistency.

The motivation for quorum systems stems from the need to make critical missions performed by machines reliable. The only way to increase the reliability of a service, aside from using intrinsically more robust hardware, is via replication. To make a service robust, it can be installed on multiple identical servers, each one of which holds a copy of the service state and performs read/write operations on it. This allows the system to provide information and perform operations even if some machines fail or communication links go down. Unfortunately, replication incurs a cost in the need to maintain the servers consistent. To enhance the availability and performance of a replicated service, Gifford and Thomas introduced in 1979 [3,10] the usage of *votes* assigned to each server, such that a majority of the sum of votes is sufficient to perform operations. More generally, quorum systems are defined formally as follows:

Quorum system: Assume a *universe* U of servers, $|U| = n$, and an arbitrary number of clients. A *quorum system* $\mathcal{Q} \subseteq 2^U$ is a set of subsets of U, every pair of which intersect. Each $Q \in \mathcal{Q}$ is called a *quorum*.

Access Protocol

To demonstrate the usability of quorum systems in constructing replicated services, quorums are used here to implement a multi-writer multi-reader atomic shared variable. Quorums have also been used in various *mutual exclusion* protocols, to achieve Consensus, and in commit protocols.

In our application, clients perform read and write operations on a variable x that is replicated at each server in the universe U. A copy of the variable x is stored at each server, along with a timestamp value t. Timestamps are assigned by a client to each replica of the variable when the client writes the replica. Different clients choose distinct timestamps, e.g., by choosing integers appended with the name of c in the low-order bits. The read and write operations are implemented as follows.

Write: For a client c to write the value v, it queries each server in some quorum Q to obtain a set of value/timestamp pairs $A = \{\langle v_u, t_u \rangle\}_{u \in Q}$; chooses a timestamp $t \in T_c$ greater than the highest

[*]Chapter in *The Encyclopedia of Distributed Computing*, Joseph Urban and Partha Dasgupta, editors, Kluwer Academic Publishers. To be published.

timestamp value in A; and updates x and the associated timestamp at each server in Q to v and t, respectively.

Read: For a client to read x, it queries each server in some quorum Q to obtain a set of value/timestamp pairs $A = \{\langle v_u, t_u \rangle\}_{u \in Q}$. The client then chooses the pair $\langle v, t \rangle$ with the highest timestamp in A to obtain the result of the read operation. It writes back $\langle v, t \rangle$ to each server in some quorum Q'.

In both read and write operations, each server updates its local variable and timestamp to the received values $\langle v, t \rangle$ only if t is greater than the timestamp currently associated with the variable. The above protocol correctly implements the semantics of a multi-writer multi-reader atomic variable (see Linearizability, Sequential Consistency).

Quorum Constructions

Perhaps the two most obvious quorum systems are the singleton, and the set of majorities, or more generally, weighted majorities suggested by Gifford [3].

Singleton: The set system $\mathcal{Q} = \{\{u\}\}$ for some $u \in U$ is the singleton quorum system.

Weighted Majorities: Assume that every server s in the universe U is assigned a number of votes w_s. Then, the set system $\mathcal{Q} = \{Q \subseteq U : \sum_{q \in Q} w_q > \frac{\sum_{q \in U} w_q}{2}\}$ is a quorum system called Weighted Majorities. When all the weights are the same, simply call this the system of Majorities.

An example of a quorum system that cannot be defined by voting is the following Grid construction:

Grid: Suppose that the universe of servers is of size $n = k^2$ for some integer k. Arrange the universe into a $\sqrt{n} \times \sqrt{n}$ grid, as shown in Figure 1. A quorum is the union of a full row and one element from each row below the full row. This yields the Grid quorum system, whose quorums are of size $O(\sqrt{n})$.

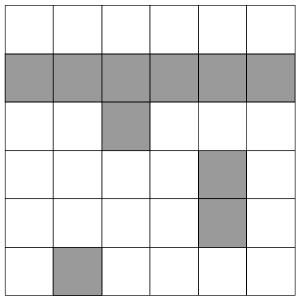

Figure 1: The Grid quorum system of 6×6, with one quorum shaded

Maekawa suggests in [4] a quorum system that has several desirable symmetry properties, and in particular, that every pair of quorums intersect in exactly one element:

FPP: Suppose that the universe of servers is of size $n = q^2 + q + 1$, where $q = p^r$ for a prime p. It is known that a finite projective plane exists for n, with $q + 1$ pairwise intersecting subsets, each subset of size $q+1$, and where each element is contained in $q+1$ subsets. Then the set of finite projective plane subsets forms a quorum system.

Voting and Related notions

Since generally it would be senseless to access a large quorum if a subset of it is a quorum, we want to focus on quorum systems that do not contain such anomalies. Garcia-Molina and Barbara [2] call such well formed systems *coteries*, defined as follows:

Coterie: A *coterie* $\mathcal{Q} \subseteq 2^U$ is a quorum system such that for any $Q, Q' \in \mathcal{Q} : Q \not\subseteq Q'$.

Of special interest are quorum systems that cannot be reduced in size (i.e., that no quorum in the system can be reduced in size). Garcia-Molina and Barbara [2] use the term "dominates" to mean that one quorum system is always superior to another, as follows:

Domination: Suppose that $\mathcal{Q}, \mathcal{Q}'$ are two coteries, $\mathcal{Q} \neq \mathcal{Q}'$, such that for every $Q' \in \mathcal{Q}'$, there exists a $Q \in \mathcal{Q}$ such that $Q \subseteq Q'$. Then $\mathcal{Q}$ *dominates* $\mathcal{Q}'$. $\mathcal{Q}'$ is *dominated* if there exists a coterie $\mathcal{Q}$ that dominates it, and is *non-dominated* if no such coterie exists.

Voting was mentioned above as an intuitive way of thinking about quorum techniques. As it turns out, vote assignments and quorums are not equivalent. Garcia-Molina and Barbara [2] show that quorum systems are strictly more general than voting, i.e. each vote assignment has some corresponding quorum system but not the other way around. In fact, for a system with n servers, there is a double-exponential ($2^{2^{cn}}$) number of non-dominated coteries, and only $O(2^{n^2})$ different vote assignments, though for $n \leq 5$, voting and non-dominated coteries are identical.

Measures

Several measures of quality have been identified to address the question of which quorum system works best for a given set of servers; among these, we elaborate on *load* and availability.

Load

A measure of the inherent performance of a quorum system is its *load*. Naor and Wool define in [7] the load of a quorum system as the probability of accessing the busiest server in the *best* case. More precisely, given a quorum system $\mathcal{Q}$, an *access strategy* w is a probability distribution on the elements of $\mathcal{Q}$; i.e., $\sum_{Q \in \mathcal{Q}} w(Q) = 1$. $w(Q)$ is the probability that quorum Q will be chosen when the service is accessed. Load is then defined as follows:

Load: Let a strategy w be given for a quorum system $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ over a universe U. For an element $u \in U$, the load induced by w on u is $l_w(u) = \sum_{Q_i \ni u} w(Q_i)$. The load induced by a strategy w on a quorum system $\mathcal{Q}$ is

$$L_w(\mathcal{Q}) = \max_{u \in U} \{l_w(u)\}.$$

The *system load* (or just *load*) on a quorum system $\mathcal{Q}$ is

$$L(\mathcal{Q}) = \min_{w} \{L_w(\mathcal{Q})\},$$

where the minimum is taken over all strategies.

The load is a best case definition, and will be achieved only if an optimal access strategy is used, and only in the case that no failures occur. A strength of this definition is that load is a property of a quorum system, and not of the protocol using it.

The following theorem was proved in [7] for all quorum systems.

Theorem: Let $\mathcal{Q}$ be a quorum system over a universe of n elements. Denote by $c(\mathcal{Q})$ the size of the smallest quorum of $\mathcal{Q}$. Then $L(\mathcal{Q}) \geq \max\{\frac{1}{c(\mathcal{Q})}, \frac{c(\mathcal{Q})}{n}\}$. Consequently, $L(\mathcal{Q}) \geq \frac{1}{\sqrt{n}}$.

Availability

The resilience f of a quorum system provides one measure of how many crash failures a quorum system is *guaranteed* to survive.

Resilience: The *resilience* f of a quorum system $\mathcal{Q}$ is the largest k such that for every set $K \subseteq U$, $|K| = k$, there exists $Q \in \mathcal{Q}$ such that $K \cap Q = \emptyset$.

Note that, the resilience f is at most $c(\mathcal{Q}) - 1$, since by disabling the members of the smallest quorum every quorum is hit. It is possible, however, that an f-resilient quorum system, though vulnerable to a few failure configurations of $f+1$ failures, can survive many configurations of more than f failures. One way to measure this property of a quorum system is to assume that each server crashes independently with probability p and then to determine the probability F_p that no quorum remains completely alive. This is known as *failure probability* and is formally defined as follows:

Failure probability: Assume that each server in the system crashes independently with probability p. For every quorum $Q \in \mathcal{Q}$ let $\mathcal{E}_Q$ be the event that Q is *hit*, i.e., at least one element $i \in Q$ has crashed. Let $crash(\mathcal{Q})$ be the event that all the quorums $Q \in \mathcal{Q}$ were hit, i.e., $crash(\mathcal{Q}) = \bigwedge_{Q \in \mathcal{Q}} \mathcal{E}_Q$. Then the system failure probability is $F_p(\mathcal{Q}) = \Pr(crash(\mathcal{Q}))$.

Peleg and Wool study the availability of quorum systems in [8]. A good failure probability $F_p(\mathcal{Q})$ for a quorum system $\mathcal{Q}$ has $\lim_{n\to\infty} F_p(\mathcal{Q}) = 0$ when $p < \frac{1}{2}$. Note that, the failure probability of any quorum system whose resilience is f is at least $e^{-\Omega(f)}$. Majorities has the best availability when $p < \frac{1}{2}$; for $p = \frac{1}{2}$, there exist quorum constructions with $F_p(\mathcal{Q}) = \frac{1}{2}$; for $p > \frac{1}{2}$, the singleton has the best failure probability $F_p(\mathcal{Q}) = p$, but for most quorum systems, $F_p(\mathcal{Q})$ tends to 1.

The load and availability of quorum systems

Quorum constructions can be compared by analyzing their behavior according to the above measures. The singleton has a load of 1, resilience 0, and failure probability $F_p = p$. This system has the best failure probability when $p > \frac{1}{2}$, but otherwise performs poorly in both availability and load.

The system of Majorities has a load of $\lceil\frac{n+1}{2n}\rceil \approx \frac{1}{2}$. It is resilient to $\lfloor\frac{n-1}{2}\rfloor$ failures, and its failure probability is $e^{-\Omega(n)}$. This system has the highest possible resilience and asymptotically optimal failure probability, but poor load.

Grid's load is $O(\frac{1}{\sqrt{n}})$, which is within a constant factor from optimal. However, its resilience is only $\sqrt{n} - 1$ and it has poor failure probability which tends to 1 as n grows.

The resilience of a FPP quorum system is $q \approx \sqrt{n}$. The load of FPP was analyzed in [7] and shown to be $L(\mathrm{FPP}) = \frac{q+1}{n} \approx 1/\sqrt{n}$, which is optimal. However, its failure probability tends to 1 as n grows.

As demonstrated by these systems, there is a tradeoff between load and fault tolerance in quorum systems, where the resilience f of a quorum system $\mathcal{Q}$ satisfies $f \leq nL(\mathcal{Q})$. Thus, improving one must come at the expense of the other, and it is in fact impossible to simultaneously achieve both optimally. One might conclude that good load conflicts with low failure probability, which is not necessarily the case. In fact, there exist quorum systems such as the Paths system of Naor and Wool [7] and the Triangle Lattice of Bazzi [1] that achieve asymptotically optimal load of $O(1/\sqrt{n})$ and have close to optimal failure probability for their quorum sizes. Another construction is the CWlog system of Peleg & Wool [9], which has unusually small quorum sizes of $\log n - \log\log n$, and for systems with quorums of this size, has optimal load, $L(\mathrm{CWlog}) = O(1/\log n)$, and optimal failure probability.

Byzantine quorum systems

For the most part, quorum systems were studied in environments where failures may simply cause servers to become unavailable (benign failures). But what if a server may exhibit arbitrary, possibly malicious behavior? Malkhi and Reiter [5] initiate the study of quorum systems in environments prone to arbitrary (Byzantine) behavior of servers. Intuitively, a quorum system tolerant of Byzantine failures is a collection of subsets of servers, each pair of which intersect in a set containing sufficiently many *correct* servers to mask out the behavior of faulty servers. More precisely, Byzantine quorum systems are defined as follows:

Masking quorum system: A quorum system $\mathcal{Q}$ is a *b-masking quorum system* if it has resilience $f \geq b$, and each pair of quorums intersect in at least $2b+1$ elements.

The masking quorum system requirements enable a client to obtain the correct answer from the service despite up to b Byzantine server failures. More precisely, a write operation remains as before; to obtain the correct value of x from a read operation, the client reads a set of value/timestamp pairs from a quorum Q and sorts them into clusters of identical pairs. It then chooses a value/timestamp pair that is returned from at least $b+1$ servers, and therefore must contain at least one correct server. The properties of masking quorum systems guarantee that at least one such cluster exists. If more than one such cluster exists, the client chooses the one with the highest timestamp. It is easy to see that any value so obtained was written before, and moreover, that the most recently written value is obtained. Thus, the semantics of a multi-writer multi-reader safe variable are obtained (see Linearizability, Sequential Consistency) in a Byzantine environment.

For a b-masking quorum system, the following lower bound on the load holds:

Theorem: Let $\mathcal{Q}$ be a b-masking quorum system. Then $L(\mathcal{Q}) \geq max\{\frac{2b+1}{c(\mathcal{Q})}, \frac{c(\mathcal{Q})}{n}\}$, and consequently $L(\mathcal{Q}) \geq \sqrt{\frac{2b+1}{n}}$.

This bound is tight, and masking quorum constructions meeting it were shown.

Malkhi and Reiter explore in [5] two variations of masking quorum systems. The first, called *dissemination quorum systems*, is suited for services that receive and distribute *self-verifying* information from correct clients (e.g., digitally signed values) that faulty servers can fail to redistribute but cannot undetectably alter. The second variation, called *opaque masking quorum systems*, is similar to regular masking quorums in that it makes no assumption of self-verifying data, but it differs in that clients do not need to know the failure scenarios for which the service was designed. This somewhat simplifies the client protocol and, in the case that the failures are maliciously induced, reveals less information to clients that could guide an attack attempting to compromise the system. It is also shown in [5] how to deal with faulty clients in addition to faulty servers.

Probabilistic quorum systems

The resilience of any quorum system is bounded by half of the number of servers. Moreover, as mentioned above, there is an inherent tradeoff between low load and good resilience, so that it is in fact impossible to simultaneously achieve both optimally. In particular, quorum systems over n servers that achieve the optimal load of $\frac{1}{\sqrt{n}}$ can tolerate at most $\sqrt{n}$ faults.

To break these limitations, Malkhi et al. propose in [6] to relax the intersection property of a quorum system so that "quorums" chosen according to a specified strategy intersect only with very high probability. They accordingly name these *probabilistic quorum systems*. These systems admit the possibility, albeit small, that two operations will be performed at non-intersecting quorums, in which case consistency of the system may suffer. However, even a small relaxation of consistency can yield dramatic improvements in the resilience and failure probability of the system, while the load remains essentially unchanged. Probabilistic quorum systems are thus most suitable for use when availability of operations despite the presence of faults is more important than certain consistency. This might be the case if the cost of inconsistent operations is high but not irrecoverable, or if obtaining the most up-to-date information is desirable but not critical, while having no information may have heavier penalties.

The family of constructions suggested in [6] is as follows:

W(n, ℓ)**:** Let U be a universe of size n. $W(n, \ell)$, $\ell \geq 1$, is the system $\langle \mathcal{Q}, w \rangle$ where $\mathcal{Q}$ is the set system $\mathcal{Q} = \{Q \subseteq U : |Q| = \ell\sqrt{n}\}$; w is an access strategy w defined by $\forall Q \in \mathcal{Q}, w(Q) = \frac{1}{|\mathcal{Q}|}$.

The probability of choosing according to w two quorums that do not intersect is less than $e^{-\ell^2}$, and can be made sufficiently small by appropriate choice of ℓ. Since every element is in $\binom{n-1}{\ell\sqrt{n}-1}$ quorums, the load $L(W(n, \ell))$ is $\frac{\ell}{\sqrt{n}} = O(\frac{1}{\sqrt{n}})$. Because only $\ell\sqrt{n}$ servers need be available in order for some quorum to be available, $W(n, \ell)$ is resilient to $n - \ell\sqrt{n}$ crashes. The failure probability of $W(n, \ell)$ is less than $e^{-\Omega(n)}$ for all $p \leq 1 - \frac{\ell}{\sqrt{n}}$, which is asymptotically optimal. Moreover, if $\frac{1}{2} \leq p \leq 1 - \frac{\ell}{\sqrt{n}}$, this probability is provably better than any (non-probabilistic) quorum system.

Relaxing consistency can also provide dramatic improvements in environments that may experience Byzantine failures. More details can be found in [6].

References

1. R. Bazzi. Planar quorums. In *Proc. 10'th Inter. Workshop on Dist. Algorithms (WDAG)*, Bologna, Italy, pages 251–268, October 1996.

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

3. D. K. Gifford. Weighted voting for replicated data. In *Proceedings of the 7th ACM Symposium on Operating Systems Principles*, pages 150–162, 1979.

4. M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. of Computer Systems*, 3(2):145–159, 1985.

5. D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing* 11(4):203–213, 1998.

6. D. Malkhi, M. Reiter, and R. Wright. Probabilistic quorum systems. In *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 267–273, August 1997.

7. M. Naor and A. Wool. The load, capacity and availability of quorum systems. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 214–225, 1994. To appear in SIAM Journal of Computing, 1998.

8. D. Peleg and A. Wool. The availability of quorum systems. *Information and Computation*, 123(2):210–223, 1995.

9. D. Peleg and A. Wool. Crumbling walls: A class of practical and efficient quorum systems. *Distributed Computing*, 10(2):87–98, 1997.

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