Quorum Systems

1

Replica Control

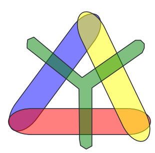
- Hides replication from transaction
- Knows location of all replicas
- Translates transaction's request to access an item into request to access a particular replica(s)
- Maintains some form of *mutual consistency*:
  - *Strong*: all replicas always have the same value
    - In every committed version of the database
  - *Weak*: all replicas eventually have the same value
  - *Quorum*: a quorum of replicas have the same value

Adapted from Chap. 24 "Implementing Distributed Transactions" by Kifer, Bernstein, and Lewis

3

Quorum Systems

- A quorum system is an intersecting family of sets over some universe.
- Formal Definition :
  - U – Universe
  - $\mathcal{F} \subseteq 2^U$
  - $\forall A, B \in \mathcal{F}\ ,\ A \cap B \neq \phi$
  - $\mathcal{F}$ is called a *quorum system* .**A**,**B** are called *quorum sets*.



- Examples
  - Majority – all sets that contain more than half of the elements .
  - Dictatorship – all sets that contain a specific element.
  - Many more…

Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

4

Quorum Systems - Applications

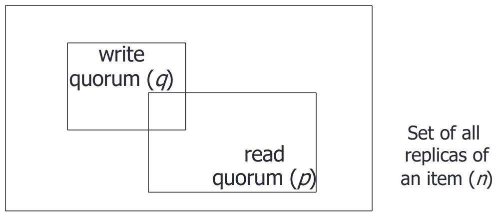
- The universe is associated with a set of servers/processors .
  - Mutual exclusion: In order to enter a critical section, a user must get permission from a quorum set.
    - Intersection property guaranties mutual exclusion .[GB85].
  - Data replication: Divide the quorum sets into reading sets and writing sets .
    - Intersection property guaranties effective search.

[GB85] Garcia-Molina H, Barbara D: How to assign votes in a distributed system. J Assoc Comput Mach 32(4):841–855 (1985)

Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

5

Quorum Consensus Replica Control

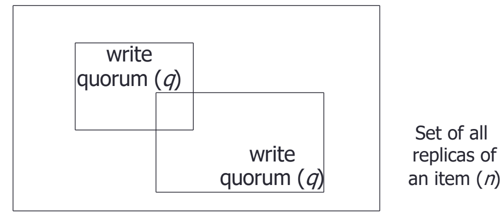


- $p+q > n$ (read/write conflict)
- An intersection between any read and any write quorum is guaranteed

Adapted from Chap. 24 "Implementing Distributed Transactions" by Kifer, Bernstein, and Lewis

6

Quorum Consensus Replica Control



- $q > n/2$ (write/write conflict)

An intersection between any two-write quorums is guaranteed

Adapted from Chap. 24 "Implementing Distributed Transactions" by Kifer, Bernstein, and Lewis

7

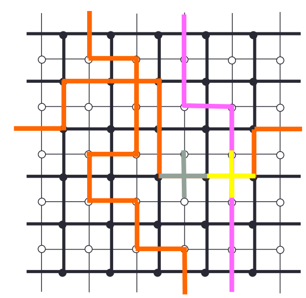
Mutual Consistency

- **Problem**: Algorithm does not maintain mutual consistency; thus reads of replicas in a read quorum might return different values
- **Solution**: Assign a timestamp to each transaction, T, when it commits; clocks are synchronized between sites so that timestamps correspond to commit order
  - T writes: replica control associates T's timestamp with all replicas in its write quorum
  - T reads: replica control returns value of replica in read quorum with largest timestamp. Since read and write quorums overlap, T gets most recent write
  - Schedules are serializable

Adapted from Chap. 24 "Implementing Distributed Transactions" by Kifer, Bernstein, and Lewis

8

Example – The *PATHS* Quorum System



- Suggested by [NW98] .
- Each element is a pair of of dual (crossing) edges.
- Quorum Set <-> Left – Right path + Top – Bottom path.
- Every top-bottom path in the dual grid intersects every left-right path in the grid.
- Data replication:
  - Top Bottom – Writing.
  - Right Left – Reading.

[NW98] Naor M,Wool A: The load, capacity, and availability of quorum systems. SIAM J Comput 27(2):423–447 (1998)

Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

9

Quorum Consensus Replica Control

- Allows a tradeoff among operations on availability and cost
  - A small quorum implies the corresponding operation is more available and can be performed more efficiently but ...
  - The smaller one quorum is, the larger the other

Measures of Quality - Load

- Load
  - Each quorum set is chosen by the user with some probability .
  - Imposes a probability of accessing a processor.
  - The load measures the access probability of the busiest processor.
    - Example: In the Majority system, if a random set of size *n/2 + 1* is chosen, the load is asymptotically half (and this is best possible).

Measures of Quality - Load

- Theorem [NW98] :
  - Let c be the size of the smallest quorum set.
  - The load of a quorum system is at least $\max\{\frac{1}{c}, \frac{c}{n}\}$
    - In other words, the load is at least $\frac{1}{\sqrt{n}}$ for every system and every access strategy.
- The *PATHS* system has load of. $\Theta(\frac{1}{\sqrt{n}})$

Failures

- Algorithm can continue to function even though some sites are inaccessible
- No special steps required to recover a site after a failure occurs
  - Replica will have an old timestamp and hence its value will not be used
  - Replica's value will be made current the next time the site is included in a write quorum

Measures of Quality - Availability

- Availability.
  - Assume each processor fails with some fixed probability *p* .What is the probability that there still exists a quorum set?
- Theorem [NW98]: [If $p > \frac{1}{2}$ then the singleton has the 'best' availability.
- The *PATHS* system has a live quorum set with probability , $1 - e^{-\Omega(\sqrt{n})}$,for $p < \frac{1}{2}$.

Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

14

Dynamic Quorum Systems.

- Motivation :Peer-to-Peer applications.
  - Related Work – dynamic probabilistic quorums [AM03]
- Processors may join and leave the quorum system.
- Objectives:
  - Low cost of Join/Leave
  - Maintain the intersection property) integrity(
    - New quorum sets
    - Adjust old quorum sets that are no longer valid.
  - Scalability
    - Load should reduce when processors join.
    - Availability should increase when processors join.
    - Probe complexity should not grow by much.

[AM03] Abraham I, Malkhi D: Probabilistic quorums for dynamic systems. In: Proceedings of the 17th International Symposium on Distributed Computing (DISC), 2003, pp 60–74

Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

17

Dynamic Quorums

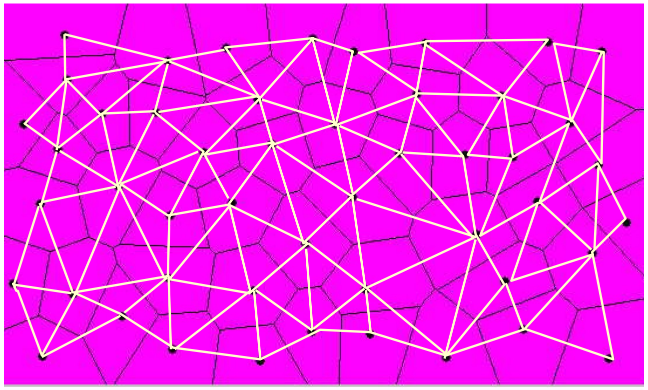
Main Idea :

- Assign each processor to a point in a continuous space.
- Divide the space into cells using a Voronoi diagram.
  - A cell consists of all the points that are closest to the processor.
  - Adding and removing a point is a *local* computation.

Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

18

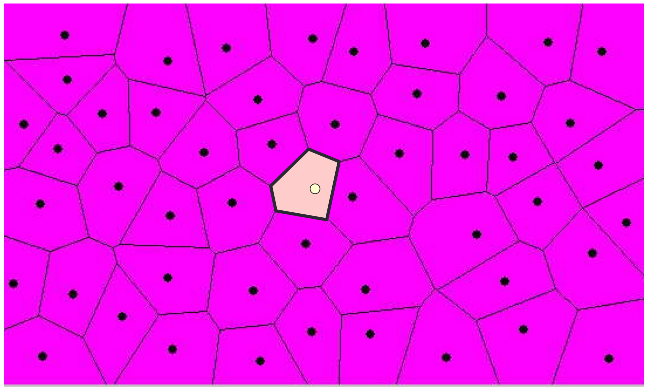
Voronoi Diagram – The Delaunay Graph



Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

19

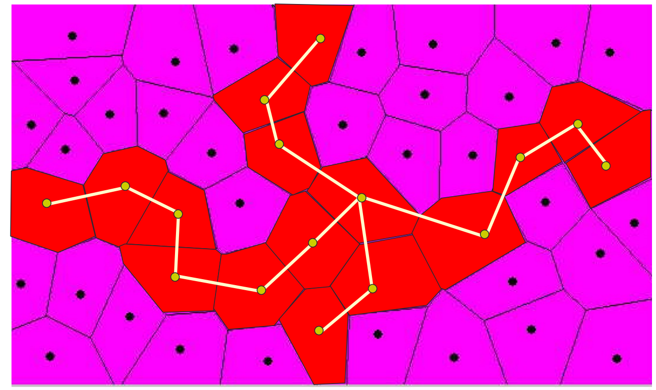
Voronoi Diagram – Join\Leave



Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

20

The Dynamic Paths Quorum System



Adapted from "Scalable and Dynamic Quorum Systems" by Moni Naor & Udi Wieder

22

46

Synchronous vs. Asynchrnonous Update

- **Problem**: Synchronous-update is slow since all replicas (or a quorum of replicas) must be updated before transaction commits
- **Solution**: With asynchronous-update only some (usually one) replica is updated as part of transaction. Updates propagate after transaction commits but…
  - only weak mutual consistency is maintained
  - serializability is not guaranteed

Adapted from Chap. 24 "Implementing Distribued Transactions" by Kifer, Bernstein, and Lewis

46

47

Summary of Distributed Transactions

- *The good news*: If transactions run at SERIALIZABLE, all sites use two-phase commit for termination and synchronous update replication, then distributed transactions are globally atomic and serializable.
- *The bad news*: To improve performance
  - applications often do not use SERIALIZABLE
  - DBMSs might not participate in two-phase commit
  - replication is generally asynchronous update
- Hence, consistent transactions might yield incorrect results

Adapted from Chap. 24 "Implementing Distribued Transactions" by Kifer, Bernstein, and Lewis

47