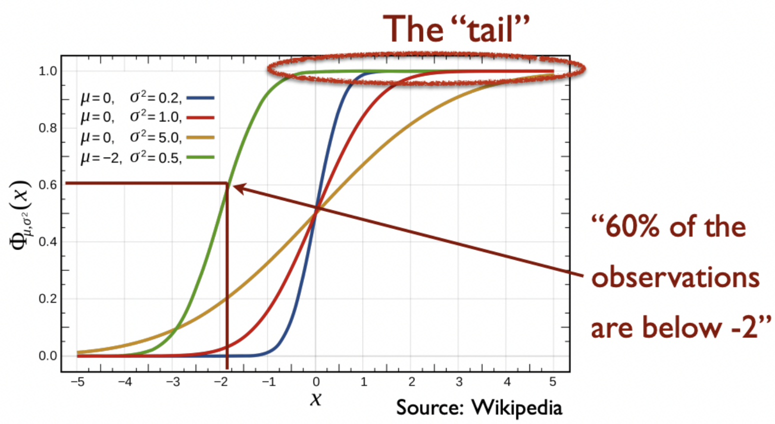
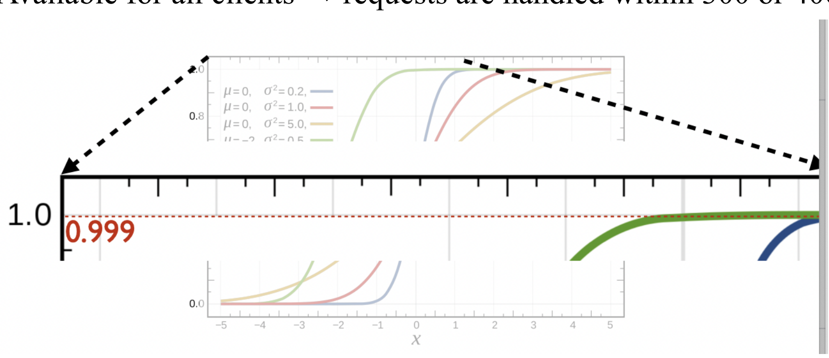
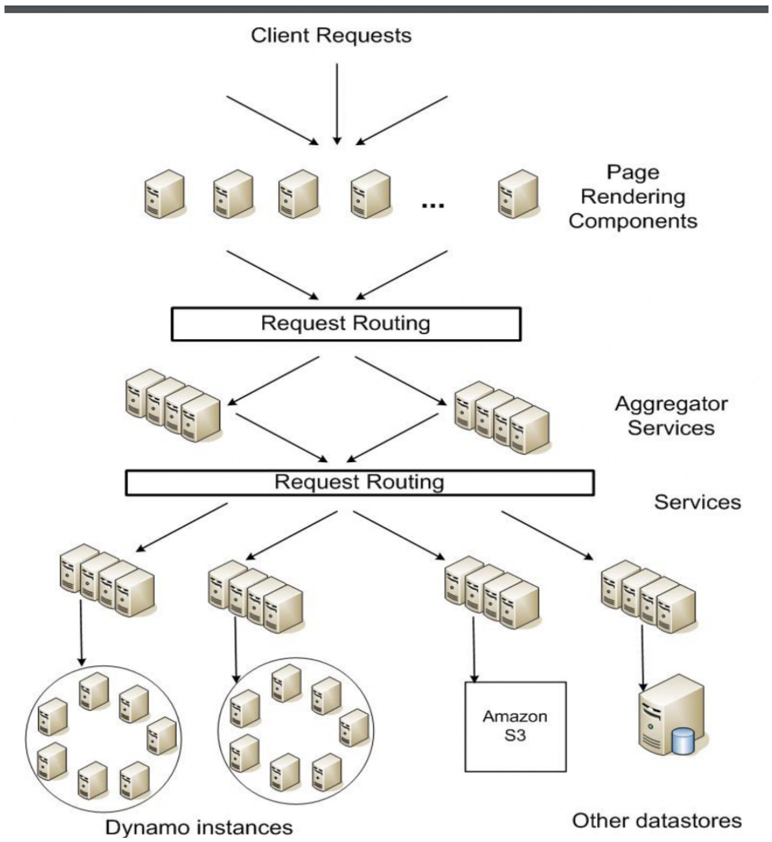
CAS CS 350
Lec 22

Dynamo

1. Why this paper
    a. Optimistic concurrency control
    b. Weak consistency model
    c. Cool mixture of ideas and techniques
2. Techniques used in Dynamo
    a. Consistent hashing
    b. Vector clocks
    c. Quorum-based distributed consensus - weaker than Raft
    d. Eventual consistency
    e. OCC(optimistic concurrency control) with conflict resolution
        i. Allow machines to do whatever they want and check whether there is conflict afterwards (if there is, we need to resolve it)
    f. Gossip-based replica synchronization
    g. Single-hop request routing
        i. Keep tail percentage
3. 99.9th percentile
    a. 

    b. Available for all clients → requests are handled within 300 or 400 milliseconds
    c. 

4. The Amazon context
    a. Culture - customer satisfaction and reliability is money (high response time means money lose for Amazon → make always available)
    b. Apps developers are experts who push hard to meet Amazon's goals
        i. Apps must work in the face of failures at all levels
    c. Systems must be able to be incrementally scaled
    d. Dynamo an always available storage service that is ALWAYS available
5. The Big picture
    a. 

    b. State is a collection of key value pairs and each key is replicated at few random nodes, by key hash (need replication due to fault tolerance, load balancing)
    c. Replicate keys on few sites
        i. The more replicas we have, more failures we can tolerate (can serve read/write requests faster)
        ii. The less replicas we have, the less resources and memory we use (low cost)

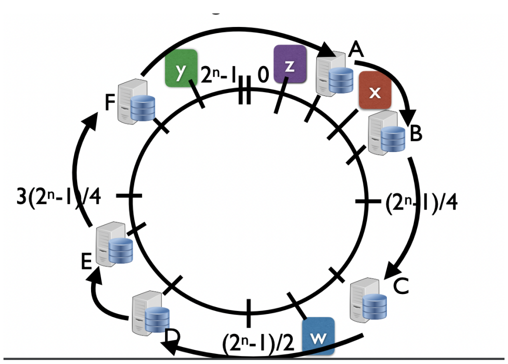
6. How it works: Interface

object,context = get(k)

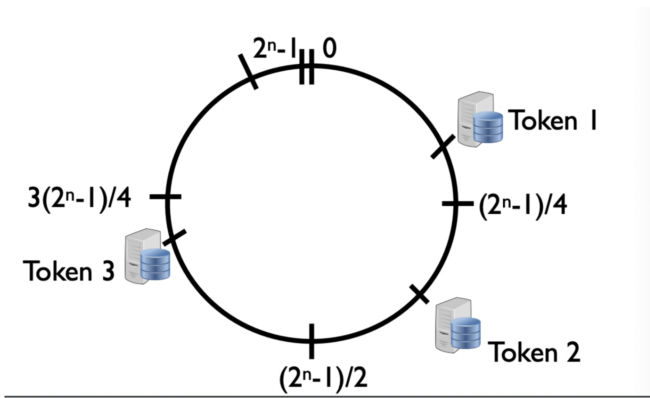
put(k,object,context)

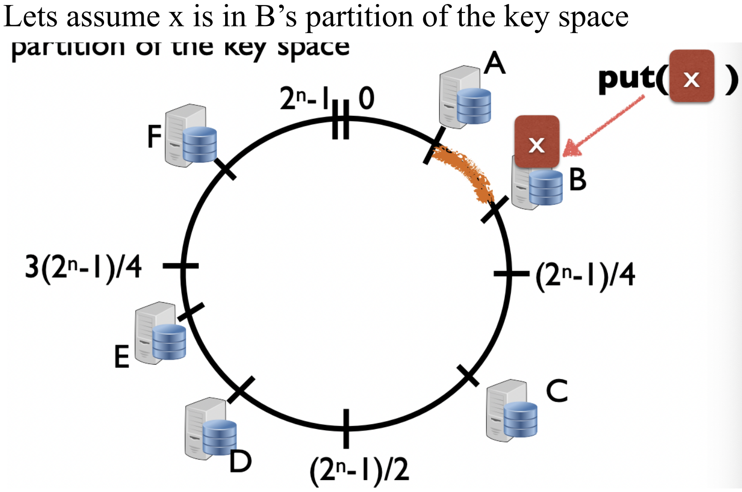
MD5 Hash of k used to generate 128 bit identifier

   a.
   b. Put is write, get is read
   c. Primary key data store: k to blob
   d. Keys are stored in dynamo nodes by using hash function
   e. Get locates objects' replicas and returns one or more versions if there are conflicts - context encodes things like vector clock of replicas
   f. Put locates where replicas should go and gets them to disk
7. Consistent hashing

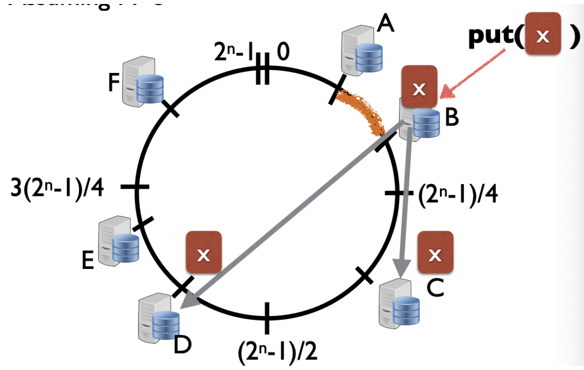


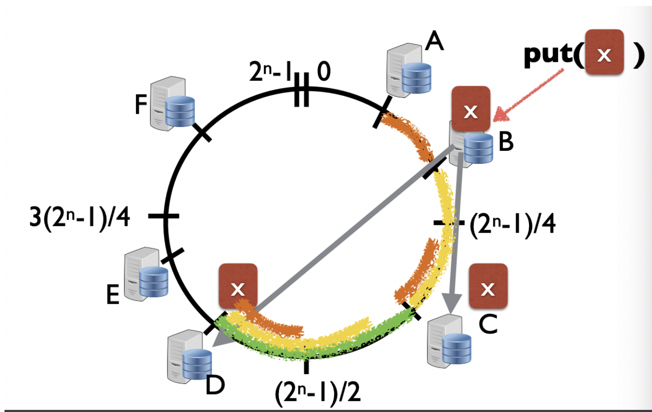
   a. Logical ring: ____________
   b. Big idea is that nodes can be inserted and removed by only talking to neighbors to exchange data rest of ring is untouched
   c. Insert the number (such as IP) to the range and map the position to the server
   d. Each ring has a master replica
   e. Hash the key → position on logical ring → the node that stores this key (master node) → goes clockwise and stores
   f. For example, if node x, it is stored in node B (first node in clockwise direction)
   g. Why not use simple hashing key such as Hash(key) % 5?
      i. If the number of servers become different (due to failing), we need to reshuffle all key-value pairs (all keys must be redistributed/repartitioned, which is expensive because the system becomes unavailable during this period)
   h. In consistent hashing, keys move to neighboring nodes on the logical ring

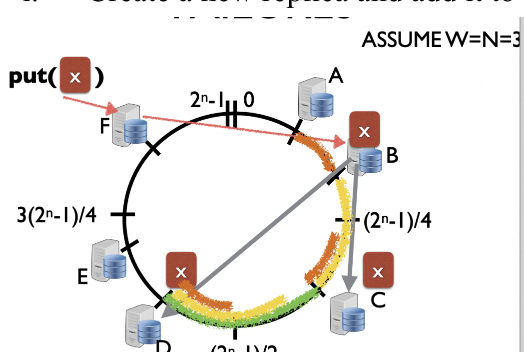
8. Why consistent hashing?
    a. Naturally decentralized
    b. Nodes to be added or removed dynamically without the need to repartition the keys
    c. Load balancing
9. Optimizations
    a. Dynamo does a bunch of optimizations
        i. Fast response time
        ii. Fast recovery
10. Use virtual nodes
    a. When adding a physical node to the right, the node is assigned many positions (virtual nodes) or "tokens"
    b. 

    c. Each token is responsible for small set of keys (to handle load balancing)
11. Replication
    a. Lets assume x is in B's partition of the key space
    b. 

    c. Master replica of key x is B
    d. The value of x will be stored in node B
    e. Dynamo will replicate the key-value pair

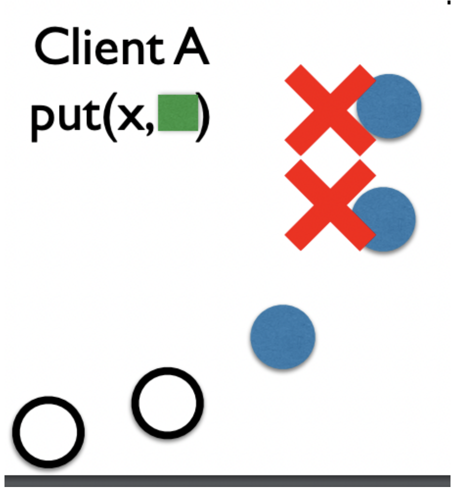


   f. Assuming N = 3,
   g. N is configurable → we can handle N-1 failures
   h. Thus a node is responsible for the region between its id and its Nth predecessor
   i. In this case, node D will replicate data from node B and C
   j. 

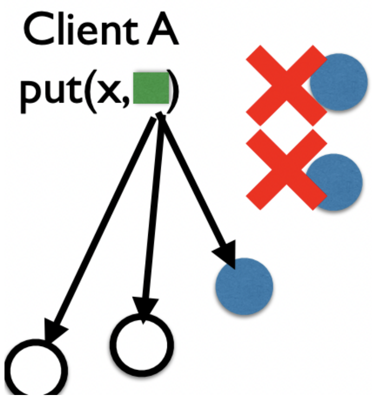
   k. Each key value pair has different N (depending on its importance)
12. What's the point of this?
   a. Fault tolerance and load balancing
   b. To achieve high availability and durability
13. Failures
   a. Temporary:
      i. Keep going around the ring until you find some who is alive
      ii. Always writable - different from PNUTS and TAO
   b. Permanent:
      i. Create a new replica and add it to the ring
   c. 

d. Assume client wants to put x and it falls between node A and B – node B is the master of x
e. Replication of x occurs to nodes C and D
f. What happens if node B fails during the request?
    i. PNUTS → request fails and system becomes unavailable until master is recovered (ensures that there is always one master replica for key-value pair)
    ii. Dynamo → still process the put/write request
        1. Request will go to successor of B → node C
        2. The object will be stored on node C
        3. It will replicate the same key-value pair on its two successor nodes, nodes D and E
        4. Node E saves the information even though it was not supposed to, due to the failure of node B
        5. Therefore, in node E, these keys are stored in separate data disk and moves it back to node B when B becomes available again (the system will send the requests regarding x from nodes B, C, and D)

14. Consequences of Always writable
    a. Consequences 1:
        i. No master! Look Mom "No hands"
        ii. Idea: "Sloppy Quorum"
    b. Consequence 2:
        i. Always writable + failures = conflicting versions can arise
        ii. Idea: "eventual consistency" techniques
15. Sloppy quorum
    a. Write should always succeed no matter what happens (write always returns success to client)
    b. Reads will show the latest write with high percentage
    c. Try to get consistency benefits of single master if no failures (reads will observe the latest write)
        i. But allows progress even if coordinator fails, which PNUTS does not
    d. When no failures, send reads/writes through single node, the coordinator
        i. Causes reads to see writes in the usual case
    e. But don't insist! Allow reads/writes to any replica if failures
16. N, R, W & Quorum
    a. Parameters configured by engineer
    b. R is the minimum number of nodes that must participate in successful read/get operation and W is the minimum number of nodes that must participate in a successful write/put operation. Setting R and W such that $R + W > N$ yields quorum-like system.

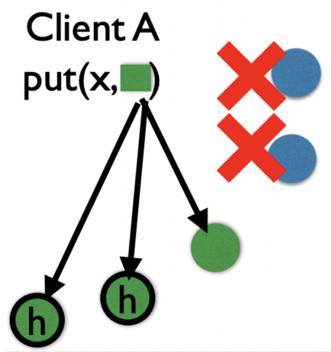
17. N, R, W
    a. N: Durability - survive lose of nodes, disks, data centers (how many copies, we can tolerate up to N-1 failures)
    b. If we don't hear back from R/W nodes then the operation is failed - loss of availability
        i. Highest availability when W = 1 for writes
    c. Lower values of R/W - increase risk of inconsistencies (return back to client before replicas are written)
        i. Reduce durability as persists only at < N locations
18. Quorum Like
    a. Interesting behavior quorum like when : R + W > N (very unlikely to have inconsistencies)
        i. Never wait for all N
        ii. But R and W will overlap
        iii. Cuts tail off delay distribution and tolerates some failures
    b. Dynamo sends concurrent requests to all N nodes when user sends write requests but will return back to the user only when W nodes have the write on its server
19. Sloppy Quorum
    a. N is first N "reachable" nodes in preference list
        i. Each node pings successors to keep rough estimate of up/down
        ii. "Sloppy" quorum, since nodes may disagree on reachable
        iii. Sloppy quorum means R/W overlap *not guaranteed*
    b. Want gets to read most recent writes
    c. R + W > N works because R and W will overlap But remember we still send it to N nodes, we just don't necessarily wait for all N, we wait for R on read and W on write
    d. Let N = 3, W = 2, R = 2
    e. 

    f. Key x is supposed to be store in three upper nodes
    g. Two nodes have failures

h. 

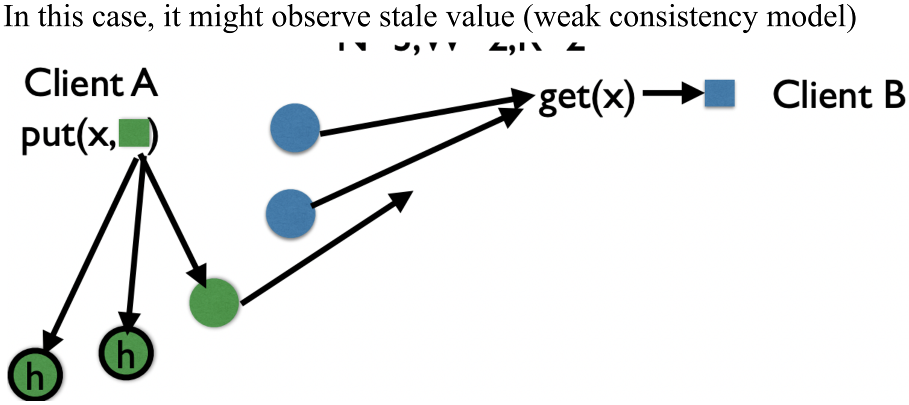
i. Therefore, it stores data in next 2 preceding servers

j. It will wait until only two nodes have the operation put(x) since W == 2

k. 

l. Another client B contact three rings and sends get(x)

m. In this case, it might observe stale value (weak consistency model)

n. 

o. Given that what nodes make up N can change based on our notion of “reachable” - N/2 + 1 might not overlap

p. This system will be fixed eventually (Dynamo will fix it some time in the future)

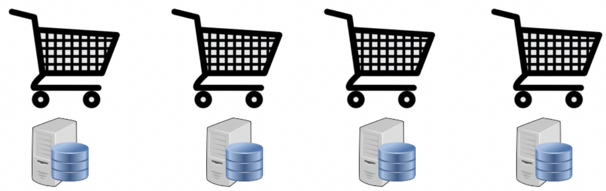
20. Eventual consistency and versioning
    a. On a put a coordinator does async writes to its replicas
    b. Put can return before hearing from all replicas (W<N)
        i. Thus replica’s can be stale - longer if failures
        ii. Can get divergent versions - gets and put to stale version can create forked version

c. Example

i. N = 3, R = 2, W = 2

ii. Carts start empty

iii.



iv. Client 1 wants to add item X (get() from n1, n2, yields)

v. N1 and n2 fail → put (“X”) goes to n3 and n4

vi.

vii. No node that participated in both quorums - lead to inconsistency

viii. Client B wants to add items

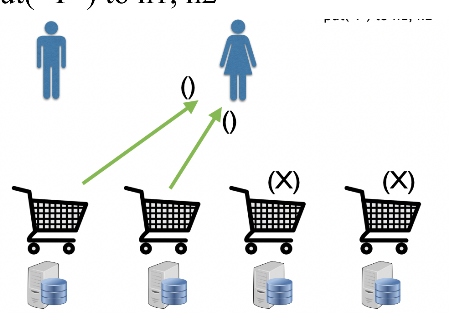
ix. N1, n2 revive

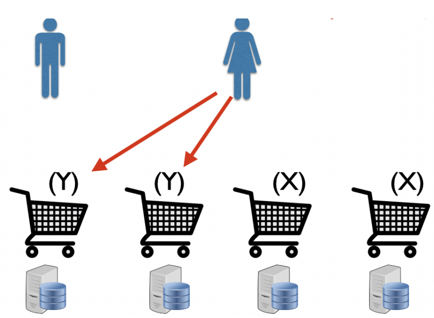
x. Client 2 wants to add Y

xi. get() from n1, n2 yields

xii. Put(“Y”) to n1, n2

xiii.



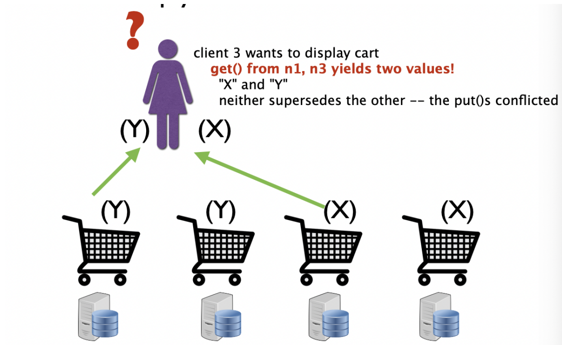


xiv. At the end, we have

xv. Client 3 wants to display cart

xvi. get() from n1, n3 yields two values - X and Y

xvii. Neither supersedes the other – the put()s conflicted

xviii. 

21. How to resolve
    a. Application dependent
    b. Shopping cart:
        i. Merge - union?