Dynamo: Amazon's Highly Available Key-value Store

Summary notes by Qian Li

- What's cool about dynamo?
    - No traditional ACID.
    - Let specific apps to choose ways to reconcile conflicts
- Main design goals?
    - Highly available: "always on"
    - 99.9% tail latency (SLA)
- Why tail latency important?
    - Large number of machines, small fraction of slow machines can have large influence.
- What are they sacrificing?
    - Consistency: eventual consistency
    - Security - assume only internal process can access, one Dynamo instance per app.
    - Primary key only, no complicated relational operations. Don't support search on columns other than primary key.
- Dynamo API
    - get(key)
    - put(key, context, value)
- Techniques Used
    - Consistent hashing
    - sloppy quorums
    - version vectors
    - Merkle trees
    - gossip protocol
    - clients must declare their own conflict resolution
- Consistent hashing
    - Strategy 1: described in the system architecture but not actually used.
    - Strategy 2: split placement and partitioning.
    - Strategy 3: the final choice.
    - Problems in common:
        - It's hard to know which keys are popular -> hot spot issue.
        - When using consistent hashing for search index, there is no locality among the keys, cannot get the benefit of accessing sequential keys. Should we keep the key sorted?
- Sloppy quorum
    - R + W > N
    - N is 100 now (in 2018).
    - R and W ideally should overlap
    - But with Sloppy quorum, R and W may not overlap.

    - What happens if the put data gets written very far down the ring?
        - Store a local copy, and put back when the node in preference list becomes available.
- Eventual consistency with vector clocks
    - Why replicas can diverge?
        - Concurrent writes.
        - Shopping cart: the union of all items.
        - Session state service -> always the latest.
    - How to prevent version vectors to be big?
        - Keep a counter per node, and remove old history.
- Resynchronization
    - If a node goes down, the neighbor can cache reads/writes.
    - Merkle trees:
        - Anti-entropy = synchronizing two replicas
        - It could sometimes be slower than a deep check. If the tree is tall, it can save time.
        - The communication cost can be large.
- Choosing coordinators
    - Two ways:
        - Ask the load balancer -> system driven
        - Dynamo-specific library in client to choose coordinator. -> client driven, can be 2.25x faster than the first one.
- Performance
    - Fig 4: day and night vary. Can see the Christmas eve because this shows December :)
    - Why the average latencies is much lower than the tail?
        - Nodes at different data centers. It can affect a lot on the latency.
    - How to improve latency?
        - Sloppy quorums: geo replication need to be chosen carefully. FB doesn't serve the client exactly nearby.
        - Change R, W, N, fine tune parameters.
        - Gossip protocol -> may generate large traffic. But may not affect a lot on the latencies.
    - Read is doing conflict resolution or not? Included in the performance number or not?
        - It is not clear from the paper.
    - Server driven or client driven is better?
        - It depends on the system requirement.
- Can we share Dynamo instance between different services?
    - Security part. Authentication?
    - Malicious client?
    - Load balancing may be an issue.
- Questions? Thoughts?

    - What they described consistent hashing is not what they really used.
    - In the mid-2007, distributed, consistent hashing was a really hot topic.
    - Suppose we need to upgrade servers/changing APIs/bring up a new service in this system, decentralized system is better/worse?
        - It is harder in peer-to-peer system.
        - Paxos consensus in peer-to-peer?
        - AWS S3 crashed because of the wrong gossip.
        - But even centralized system can go wrong - Gmail went down in 2011.
- Session state?
    - cookie, who logged in with the cookie.
- Maintain a routing table introduces overhead.
    - Always allocate more space in hash table to ensure performance.
    - Communicate the entire routing table.
- How to test the system to decide the size of the hash table?
    - Make assumptions the distribution
    - Machine learning can solve the issue?
    - Just build one node, simulate how many writes and reads to do.
- Related work
    - Spanner by Google: uses atomic clock.
    - FaRM by Microsoft.
- How to make this paper easier to read?
    - Write examples: shopping cart
    - Description of the context: what we have in a “context”.
    - Linking concept in real world.
    - They never show what happened when they tried version1, 2, 3
    - It describes everything.
- Virtual nodes?
    - Hash the id and get multiple virtual nodes.
    - It is similar to micro sharding
- Dynamo is actually using existing DB.
    - Berkeley DB.
- They didn’t say how much the shopping card improved from Dynamo.
    - Some differences between industry paper and academia paper.