Dynamo: Amazon’s Highly Available Key-value Store

Jianan Wang
**Motivation**

- **Scale**
  - Built on myriad of servers and network components
  - Deal with loss server (e.g., outage) and extension (business enlarging)
- **Simple: key-value**
  - This simple query model is fine and more efficient than relational schema (shopping cart vs. order)
- **Highly available (sacrifice consistency)**
  - “Always on” service for business
- **Guarantee Service Level Agreements (SLA)**
SERVICE LEVEL AGREEMENTS (SLA)

- Application can deliver its functionality in **bounded** time
  - Every dependency in the platform needs to deliver its functionality with even tighter bounds.
- Example
  - service guaranteeing that it will provide a response within 300ms for **99.9%** of its requests for a peak client load of 500 requests per second.
SERVICE-ORIENTED ARCHITECTURE

![Diagram of a service-oriented architecture]
DESIGN CONSIDERATION

- **Sacrifice strong consistency for availability (SLA)**
  - ACID guarantees tend to have poor availability.

- “always writeable”
  - (data store where no updates are rejected due to failures or concurrent writes. Conflict resolution is executed during *read* instead of *write*.)

- Incremental scalability.
  - Symmetry
  - Decentralization
  - Heterogeneity
SYSTEM ARCHITECTURE

- Partitioning
- High Availability for writes
- Handling temporary failures
- Recovering from permanent failures
- Membership and failure detection
PARTITION ALGORITHM

- Consistent hashing: the output range of a hash function is treated as a fixed circular space or “ring”.
- ”Virtual Nodes”: Each node can be responsible for more than one virtual node.
ADVANTAGES OF USING VIRTUAL NODES:

- If a node becomes unavailable, his load is evenly dispersed across the remaining available nodes.
- The newly available node accepts a roughly equivalent amount of load from each of the other available nodes.
- The load can be decided based on its capacity, accounting for heterogeneity in the physical infrastructure.
REPLICATION

- Each data item is replicated at N hosts.
- "preference list": The list of nodes that is responsible for storing a particular key.
- eg N=3.
DATA VERSIONING

- A put() call may return to its caller before the update has been applied at all the replicas.
- A get() call may return many versions of the same object.
- Key Challenge: distinct version sub-histories - need to be reconciled.
- Solution: uses vector clocks in order to capture causality between different versions of the same object.
VECTOR CLOCK

- A vector clock is a list of (node, counter) pairs.
- Every version of every object is associated with one vector clock.
- *If the counters on the first object’s clock are less-than-or-equal to all of the nodes in the second clock, then the first is an ancestor of the second and can be forgotten.*
VECTOR CLOCK EXAMPLE

D1 ([Sx,1])

D2 ([Sx,2])

D3 ([Sx,2],[Sy,1])  D4 ([Sx,2],[Sz,1])

D5 ([Sx,3],[Sy,1],[Sz,1])
**VECTOR CLOCK**

- write requests may be handled by nodes that are not in the top N nodes in the preference list causing the size of vector clock to grow.
- stores a timestamp that indicates the last time the node updated the data item.
- When the number of (node, counter) pairs in the vector clock reaches a threshold (say 10), the oldest pair is removed from the clock.
EXECUTION OF GET() AND PUT() OPERATIONS

- Two strategies to select a node:
  - A generic load balancer
    - not have to link any code specific to Dynamo in its application
  - Directly to the appropriate coordinator nodes.
    - achieve lower latency
TEMPORARY FAILURES
SLOPPY QUORUM

- R/W is the minimum number of nodes that must participate in a successful read/write operation.

- Setting R + W > N yields a quorum-like system.

- In this model, the latency of a get (or put) operation is dictated by the slowest of the R (or W) replicas.
HINTED HANDOFF

- Assume $N = 3$. When $A$ is temporarily down or unreachable during a write, send replica to $D$.
- $D$ is hinted that the replica belongs to $A$ and it will deliver to $A$ when $A$ is recovered.
- Again: “always writeable”
REPLICA SYNCHRONIZATION

Merkle tree:
- A hash tree where leaves are hashes of the values of individual keys.
- Parent nodes higher in the tree are hashes of their respective children.
- Each branch of the tree can be checked independently without requiring nodes to download the entire tree.
- Help in reducing the amount of data that needs to be transferred while checking for inconsistencies among replicas.
### Summary of Techniques Used in Dynamo and Their Advantages

<table>
<thead>
<tr>
<th>Problem</th>
<th>Technique</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>Consistent Hashing</td>
<td>Incremental Scalability</td>
</tr>
<tr>
<td>High Availability for writes</td>
<td>Vector clocks with reconciliation during reads</td>
<td>Version size is decoupled from update rates.</td>
</tr>
<tr>
<td>Handling temporary failures</td>
<td>Sloppy Quorum and hinted handoff</td>
<td>Provides high availability and durability guarantee when some of the replicas are not available.</td>
</tr>
<tr>
<td>Recovering from permanent failures</td>
<td>Anti-entropy using Merkle trees</td>
<td>Synchronizes divergent replicas in the background.</td>
</tr>
<tr>
<td>Membership and failure detection</td>
<td>Gossip-based membership protocol and failure detection.</td>
<td>Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.</td>
</tr>
</tbody>
</table>
IMPLEMENTATION

- **Java**
- Local persistence component allows for **different storage engines** to be plugged in:
  - Berkeley Database (BDB) Transactional Data Store: object of tens of kilobytes
  - MySQL: object of > tens of kilobytes
  - BDB Java Edition, etc.
PERFORMANCE

- diurnal pattern
- write>read latency (disk access).
- 99.9>Avg latency (variability in request load, object sizes, and locality patterns)
IMPROVEMENT

- Each storage node maintains an object buffer in its main memory.
- Each write operation is stored in the buffer and gets periodically written to storage by a writer thread. (like spooling of printer 😊)
- Read operations first check if the requested key is present in the buffer
Improvement (Cont’d)

- lowering the 99.9th percentile latency during peak traffic with a buffer of thousand objects.
- write buffering smoothes out higher percentile latencies
IMPROVEMENT (CONT’D)

- A server crash can result in missing writes that were queued up in the buffer.
- Have the coordinator choose one out of the N replicas to perform a “durable write”
- The performance of the write operation is not affected by the performance of the durable write operation
**Balance**

- Imbalance: Deviates from the average load by a value more than a certain threshold (here 15%)
- Imbalance ratio decreases with increasing load
- under high loads, a large number of popular keys are accessed and the load is evenly distributed
PARTITIONING AND PLACEMENT OF KEY

- The space needed to maintain the membership at each node increases linearly with the number of nodes in the system
PARTITIONING AND PLACEMENT OF KEY (CONT’D)

- divides the hash space into \( Q \) equally sized partitions
- The primary advantages of this strategy are:
  1. **decoupling** of partitioning and partition placement,
  2. **enabling the possibility** of changing the placement scheme at runtime.
PARTITIONING AND PLACEMENT OF KEY (CONT’D)

- Divides the hash space into \( Q \) equally sized partitions
- Each node is assigned \( Q/S \) tokens where \( S \) is the number of nodes in the system.
- When a node leaves the system, its tokens are randomly distributed to the remaining nodes.
- When a node joins the system it "steals" tokens from nodes in the system.
LOAD BALANCING EFFICIENCY

- Strategy 3 achieves better efficiency
CONCLUSION

- Highly available and scalable data store for Amazon.com’s e-commerce platform.
- Successful in handling server failures, data center failures and network partitions.
- Allows service owners to customize their storage system by allowing them to tune the parameters N, R, and W.
Some of the slides in this presentation have been taken from the presentation created by paper authors Giuseppe DeCandia and so on.