Dynamo: Amazon's Highly Available Key Value Store

Vivek Kale

What we are Dealing With

- Amazon is an E-commerce platform with
  - millions of customers
  - thousands of servers distributed across the world.

- Need to support best-seller lists, shopping carts, customer preferences, session management, sales rank, product catalog.
- If one data center goes down, people should still be able to buy and sell products. If Amazon.com service is down even for a short time, the business loses large amounts of money.

Design Requirements

1. **Reliability**: Even in the face of malfunctioning systems, software should not malfunction (e.g. customers should not be charged more to their credit card than they paid).

2. **Scalability**: To support growth in business and to be agile in changing market conditions, scalability is extremely important

3. **Performance**: Latencies should be very low because of Service level agreements(SLA).

4. **Availability:** A small period of downtime can hurt a corporation like Amazon financially and also diminish trust of customers.

Design of Dynamo

Primary Considerations:

1. Lose Strong Consistency for the sake of High Availability
2. Conflict resolution is always executed during read, rather than during a write. No writes are lost.

Some other considerations

1. Incremental scalability
2. Symmetry
3. Decentralization
4. Heterogeneity

Related Work

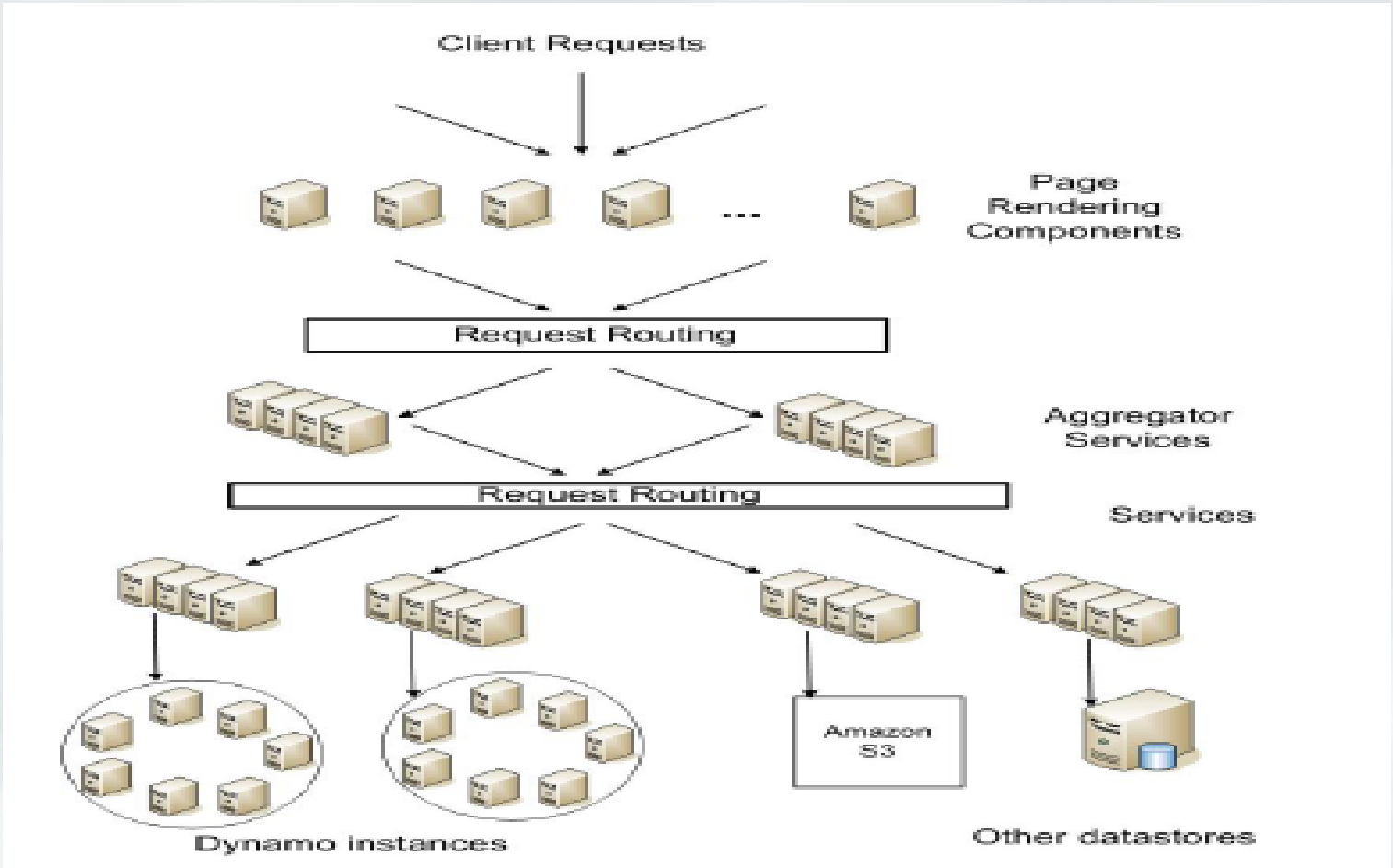
- **Peer to Peer Systems** : Chord ensures that queries can be answered in a bounded number of hops. OceanStore resolves conflicts through processing a series of updates and finding a total order among them.
- **Distributed File Systems:** Google File System, Coda

- **Relational Databases:** Not capable of handling network partitions, uses strong consistency, and very expensive.

- ***Dynamo differs in the following ways:***

1. Always writeable
2. Single Administrative domain
3. Support for hierarchical namespaces not required
4. Built for "latency sensititive" applications. Read and write operations should be performed within a 2-3 milliseconds ("zero-hop DHT")

Service-Level Agreements



A large number of dependencies mean that latencies of each component should be even lower.

Design Techniques and Advantages

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

Dynamo uses the right balance of fundamental techniques for a very large-scale distributed system

Partitioning Algorithm and Replication

1. Consistent hashing: the range of a hash function is treated as a ring.

2. ”Virtual Nodes”: Each node might be responsible for more than one virtual node.

3. Data is replicated at N hosts and contains a preference list. This is a list of nodes that is responsible for storing a particular key.

Execution of Get() and Put()

Two separate Approaches:

Approach 1: Each request is routed through a load balancer. A node is selected based on this load information.

Approach 2: Partition-aware library routes requests directly to the appropriate coordinator nodes.

Terminology

N: Top most healthy/preferred nodes in the system

R: Minimum number of nodes that participate in a successful read operation.

W: Minimum Number of ***nodes*** that participate in a successful write operation.

coordinator: a node designated to handle a read or write operation.

Consistency Protocol: Strict Quorum

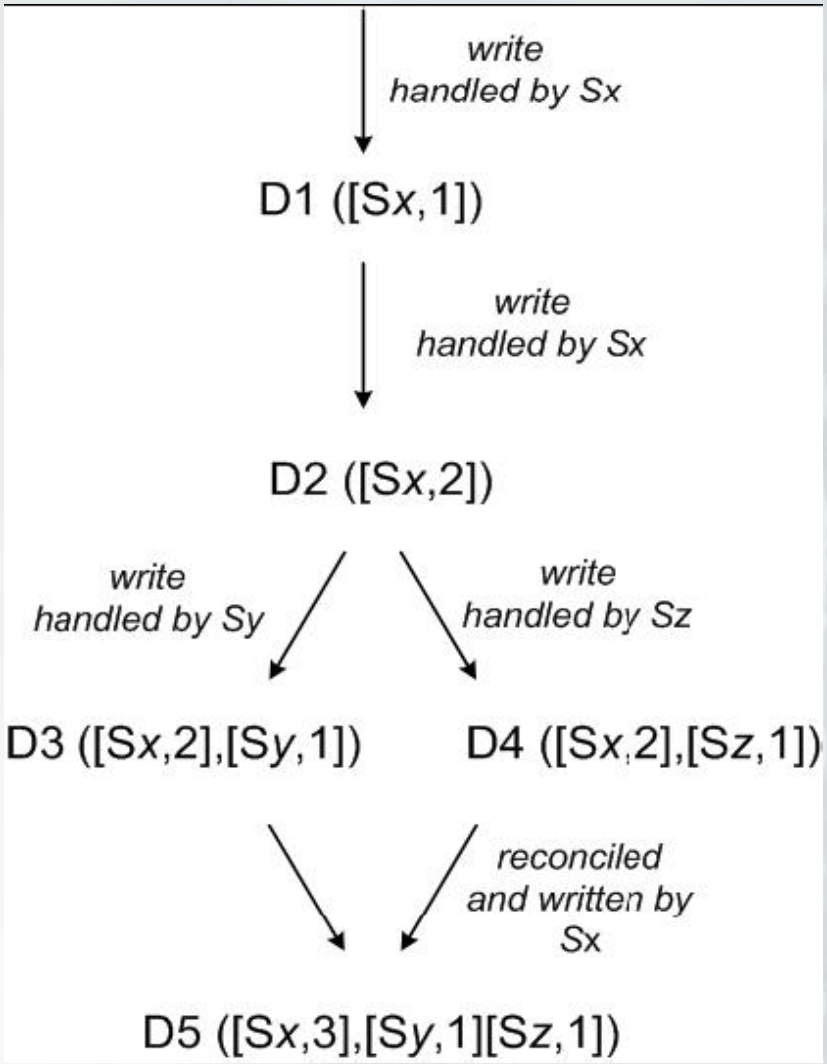
- Read and write operations involve the first N nodes in a preference list.
- The ratio of R to W is the minimum number of nodes that must participate in a successful read/write operation.
- Latency of a get()/put() operation is dependent on the slowest of the R or W replica nodes.
- R and W are usually configured to be below N, but R+W is set to be greater than N

An Improvement: Sloppy Quorum

- To ensure availability, Dynamo uses a sloppy quorum: all read and write operations are on the first N healthy nodes, skipping some nodes on the consistent hashing rings.
- Dynamo uses Hinted Handoff, where a node that is temporarily down, the data is handed off to another healthy node with a hint that the data should be redirected to the original recipient.
- Nodes stored these replicas in their local database which is scanned periodically.
- The preference list of a key is created so that the objects are replicated across multiple data centers.
- If one data center is down, the read and write operations can still succeed.

Versioning Using Vector Clocks

- Every version of every object is associated with one vector clock.
- A vector clock consist of a list of elements <node, counter>
- ***Rule:*** If the counters on the first object's clock less than all of the counters of the nodes in the second clock, then the first node is an ancestor of the second.
- Thus, the first node can be eliminated.



Average Latencies for Reads/Writes

1. There is a significant difference between daytime and nighttime request rates (which they refer to as “di-urnal”)

2. Write latencies are higher than read latencies because writes always result in disk access.

3. The 99th percentile latencies are much higher (about 100 times larger) than the average latencies.

4. Also, it is interesting to note the latency variation for reads is much higher than latency variation of writes

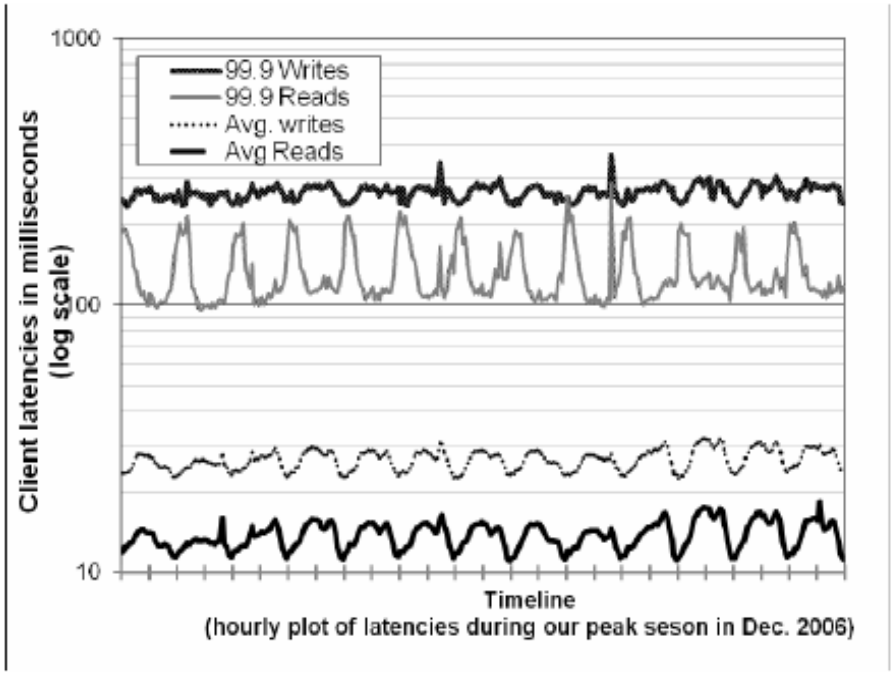


Figure 4: Average and 99.9 percentiles of latencies for read and write requests during our peak request season of December 2006. The intervals between consecutive ticks in the x-axis correspond to 12 hours. Latencies follow a diurnal pattern similar to the request rate and 99.9 percentile latencies are an order of magnitude higher than averages

Buffered and Non-Buffered Writes

1. Buffering writes for objects lowers latency by factor of 5, even for a small 1000-object buffer.

2. Buffering also reduces performance variations.

3. However, server crashes can result in missing writes queued in buffer.

4. To reduce this durability risk, the coordinator chooses one of N replicas to perform a durable write.

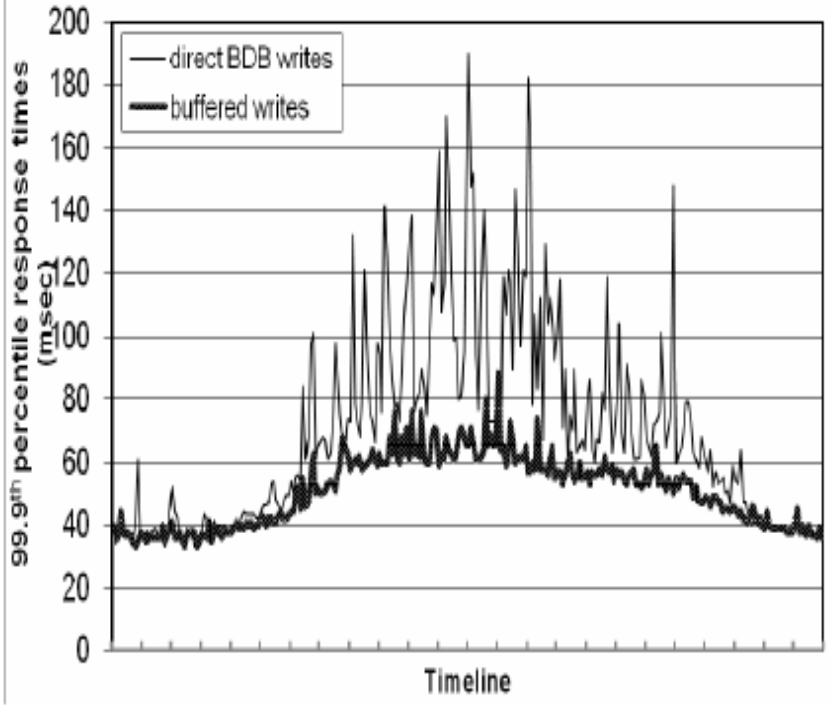


Figure 5: Comparison of performance of 99.9th percentile latencies for buffered vs. non-buffered writes over a period of 24 hours. The intervals between consecutive ticks in the x-axis correspond to one hour.

Fraction of Nodes out of Balance

1.The number of nodes out-of-balance(imbalance ratio) *decreases with increasing request load.*

2. Explanation for high loads: many popular keys are accessed. Due to uniform distribution of keys, load is evenly distributed.

3. Explanation for low loads: fewer popular keys are accessed, giving higher load imbalance.

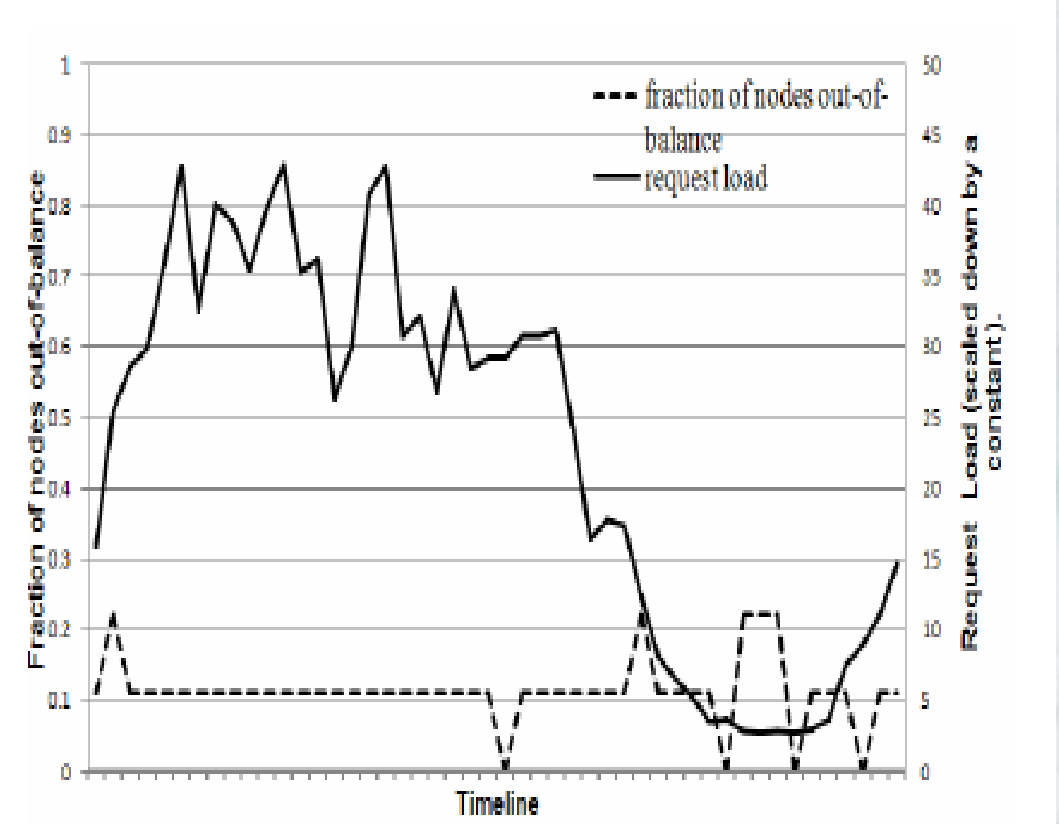


Figure 6: Fraction of nodes that are out-of-balance (i.e., nodes whose request load is above a certain threshold from the average system load) and their corresponding request load. The interval between ticks in x-axis corresponds to a time period of 30 minutes.

Conclusions

- Dynamo combines different fundamental techniques of distributed systems to achieve scalability and reliability.
- Its credibility is shown through its success in a challenging e-commerce application environments
- The "eventually-consistent" storage system can be a basis for many other highly available applications.
- Dynamo can be extended to provide for many further optimizations, particularly with the constantly changing demands in industry.

Discussion

1. Would this be applicable to other applications and other ccntexts? What would the values of N,R, W be for them?

2. Are there better methods than Merkle trees? Are the storage requirements adequate?

3. Is the synthesis of the many different techniques the contribution? Or is there one particular technique that stands out?

4. Why do you think Dynamo allows the number of read (R) and written (W) replicas to be configured? How can one tune R and W to be sure to "eventually" have consistency?