

# Dynamo Paper Summary

Paper: [amazon-dynamo-sosp2007](#)

## Design Principles

**Decentralization:** Dynamo uses consistent hashing to distribute data across nodes evenly, avoiding hot spots and ensuring that each node operates independently, thus eliminating single points of failure.

**Eventual Consistency:** Prioritizes availability and partition tolerance over immediate consistency, ensuring that the system remains operational and clients can always read and write data, even during network partitions.

**Incremental Scalability:** Dynamo allows seamless scaling by adding new nodes, which automatically take over portions of the data range from existing nodes to balance the load and ensure continuous performance.

## System Architecture

**Partitioning Algorithm:** Consistent hashing maps both nodes and data items to a circular space, ensuring even distribution of data and making it easy to add or remove nodes.

**Replication:** Data is replicated across multiple nodes to ensure durability and availability, using hinted handoff and anti-entropy algorithms.

**Versioning:** Uses vector clocks to capture the causality between different versions of an object, allowing for effective conflict detection and resolution through application-specific logic.

**Quorum-Based Replication:** Ensures data consistency and availability by requiring read and write operations to overlap sufficiently. Uses sloppy quorum instead of a strict quorum policy.

## Techniques

**Hinted Handoff:** Temporarily stores data on alternate nodes if the preferred node is down, ensuring data is not lost and is eventually handed back to the correct node once it recovers (similar to when you are not at home and the delivery person gives your package to your neighbor).

**Merkle Trees:** Used for efficient data synchronization between replicas by comparing data between nodes, minimizing the amount of data transferred.

**Sloppy Quorum:** Allows writes to the nearest available nodes when replica nodes of the specific key are unavailable, maintaining availability and moving data to the preferred nodes once they become available again.

## Key Takeaways

**High Availability:** Ensures continuous service even during server failures and network partitions, prioritizing availability and fault tolerance.

**Scalability:** Easily scales by adding new nodes without significant reconfiguration, supporting Amazon's need for high throughput and low latency.

**Data Consistency:** Uses eventual consistency with mechanisms like vector clocks and quorum-based replication to ensure data reliability and resolve conflicts.

## Further Reading

- [Designing Data Intensive Applications](#): Chapter 5, Replication
- [Patterns of Distributed Systems](#): Lamport Clock, Version Vector, Majority Quorum, Gossip Dissemination