Dynamo

• Highly available and scalable distributed data store
• Manages state of services that have high reliability and performance requirements
  – best-seller lists
  – shopping carts
  – customer preferences
  – session management
  – sales rank
  – product catalog
Background

• Amazon e-commerce platform
  – hundreds of services
    - from recommendations to fraud detection
  – tens of thousands of servers
    - across many data centers worldwide

• RDBMS?
  – no …
  – don’t need complex queries
  – RDBMS replication strategies are inefficient

• Simple queries
  – key/value pairs (blobs), < 1MB
Transactions at Amazon

• ACID properties
  – atomicity ✓
    - pretty important
  – consistency ❌
    - weak is good …
  – isolation ❌
    - forget it …
  – durability ✓
    - pretty important
Efficiency

• Strict latency requirements for services
  – measured at 99.9th percentile
• Typical request to put together a page requires responses from 150 services
• Services rely on dynamo for storage
Service-Level Agreements

• Contract between client and server
• Example
  – provide a response within 300 ms for 99.9% of its requests at a peak load of 500 requests/sec
• Requires *admission control*
  – not discussed
Amazon Architecture
Design Considerations

• Replication important for durability and availability

• Tradeoff between consistency and performance
  – writes should never be delayed
  – reads should return quickly, despite possible inconsistencies
Assigning Data to Nodes

• Consistent hashing (as in Chord)
  – no finger tables
  – each node knows the complete assignment

• Issues
  – replication
  – coping with nodes of varying performance
Consistent Hashing

coordinator(1) = 1
coordinator(2) = 3
coordinator(6) = 0
Virtual Nodes

coorinator(6) = 0
Adding and Deleting Nodes

• Without virtual nodes (e.g., Chord)
  – added nodes acquire objects only from successors
  – deleted nodes give objects only to successors

• With (randomly distributed) virtual nodes
  – added nodes acquire objects uniformly distributed
  – deleted nodes give objects to uniformly distributed others
Replication

• Each key assigned to coordinator node
  – via DHT
• Coordinator replicates data items at n-1 other nodes
  – n is an application parameter
  – next n-1 distinct real nodes on ring
• Set of n nodes for a data item is called its preference list
Versioning

• Eventual consistency
  – update request may return to caller before it is propagated to all replicas
  – delete request may reach replica that doesn’t have what is being deleted
    - delete requests treated as “put” requests
  – versions are immutable (but garbage collectible)
The Shopping Cart ...
Vector Clocks

\[
\begin{align*}
D1 & (\{Sx, 1\}) \\
D2 & (\{Sx, 2\}) \\
D3 & (\{Sx, 2, Sy, 1\}) \\
D4 & (\{Sx, 2, Sz, 1\}) \\
D5 & (\{Sx, 3, Sy, 1, Sz, 1\})
\end{align*}
\]
Reconciliation

- Reconciliation done only on reads, which may return multiple values
  - easy reconciliation if values are causally ordered
  - application handles it otherwise
Quorums

• *gets* and *puts* go to coordinator, if available

• *put:*
  – coordinator writes new version locally
  – sends it to next n-1 nodes
  – when w-1 respond, put is considered successful

• *get:*
  – coordinator requests existing versions from next n-1 nodes
  – waits for r responses before returning to client

• r+w>n
  – typically n=3, r=w=2
“Sloppy” Quorum

• What if not all n nodes (or even w nodes) are available?
  – don’t want to deny a put request
• Use first n healthy nodes encountered
  – data tagged with the node it should go on
  – written back to that node when available
• Handles failures of entire data centers!
  – storage nodes spread across data centers
Anti-Entropy

- Replicas synced in background
- How to determine whether replicas differ?
- Merkle trees used to compare contents
  - leaves are hashes of contents
  - parents are hashes of children
- Nodes maintain Merkle trees of key ranges
  - each virtual node defines a key range
Ring Membership

• Gossip protocol used to distribute membership info and key ranges
• Failure detection
  – simple time-out on communication
Performance: Latency

(hourly plot of latencies during our peak season in Dec. 2006)
Performance: Buffering

- direct BDB writes
- buffered writes

99.9th percentile response times (msec)

Timeline
Performance: Load Balancing
Example Uses

• Business-logic-specific reconciliation
  – data replicated across multiple nodes
  – reconciliation is application-specific
    - e.g.: shopping carts merged

• Timestamp-based reconciliation
  – real-time stamp used: “last write wins”
  – used to maintain customer session information

• High-performance read engine
  – r=1, w=n
  – used for product catalog and promotional items
Divergent Versions

• Is it a real problem with shopping carts?
• In one 24-hour period:
  – 99.94% of requests saw exactly one version
• Divergent versions usually not due to failures
• Due to concurrent writers
  – writers probably aren’t human!
Post 2007

• Dynamo successful, but not popular among developers
  – difficult to integrate and manage
• SimpleDB preferred
  – easier to use
  – much less robust
• DynamoDB released in January 2012
  – combines both
  – no details