Dynamo: Amazon's Highly Available Key-value Store

Phil Gibbons

15-712 F15

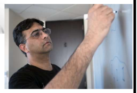
Lecture 15

Today's Reminders

- **Projects**
  - Good discussions last Friday with all groups
  - 3 page project proposals due Friday October 30

Dynamo: Amazon's Highly Available Key-value Store [SOSP'07]

- **Giuseppe DeCandia** (Midokura)
- **Deniz Hastonrun** (Facebook)
- **Madan Jampani** (Open Networking Lab)
- **Gunavardhan Kakulapati** (founded Bidstalk)
- **Avinash Lakshman** (founded Hedvig)
- **Alex Pilchin** (Architech)
- **Swaminathan Sivasubramanian** (GM NoSQL @Amazon)
- **Peter Vosshall** (VP @ Amazon)
- **Werner Vogels** (CTO @ Amazon)



Contributions

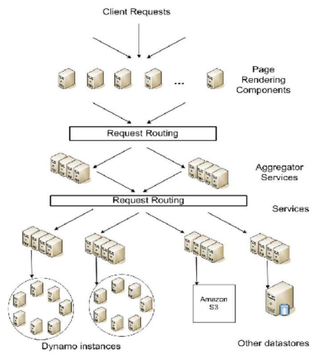
"The main contribution of this work for the research community is the evaluation of how different techniques can be combined to provide a single highly-available system."

"The paper is mainly about the lessons learned."

System Assumptions & Requirements

- **Query Model & ACID Properties**
  - Key-value queries; State stored as blobs; No schema; Ops on single data item; Objects < 1 MB
  - Sacrifice consistency, isolation
- **Efficiency**
  - Commodity HW
  - Stringent latency SLAs (e.g., 99.9% within 300 millisecs)
- **Trust**
  - Non-hostile environment, no authentication
- **Scale**
  - 100s hosts

Service-oriented Architecture



"The choice for 99.9% [SLA] over an even higher percentile has been made based on a cost-benefit analysis which demonstrated a significant increase in cost to improve performance that much."

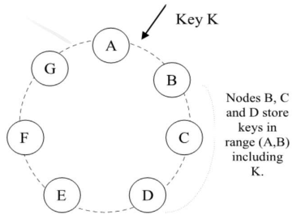
Design Considerations

"One of the main design considerations for Dynamo is to give services control over their system properties, such as durability and consistency, and to let services make their own tradeoffs between functionality, performance and cost-effectiveness."

- **Conflict resolution after disconnection**
  - When: During reads in order to ensure writes never rejected
  - Who: Application, fall back to "last write wins" at data store
- **Incremental scalability in storage hosts**
- **Exploit heterogeneity in hosts**
- **Symmetry: Each node has same responsibilities as its peers**
- **Decentralization of control**

Zero-Hop DHT

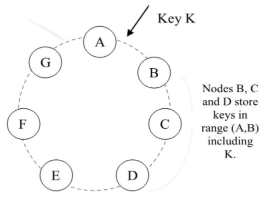
- **Distributed Hash Tables (DHTs) such as Chord [Stoica et al, Sigcomm'01] are based on consistency hashing [Karger et al, STOC'97]**

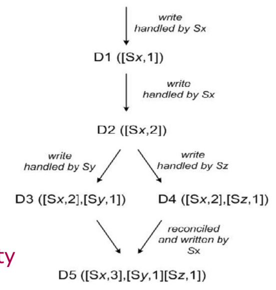


- **Why use DHTs?**
- **Zero-hop: Each node maintains enough routing info locally to route directly to destination node**

Partitioning, Replication, Versioning

- **Each storage node assigned multiple positions on ring, across data centers**
- **Replication in successor nodes around the ring**

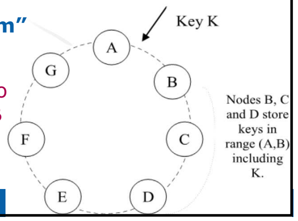




- **Data versioning**
  - Vector clocks capture version causality
  - Truncate clocks if get too long

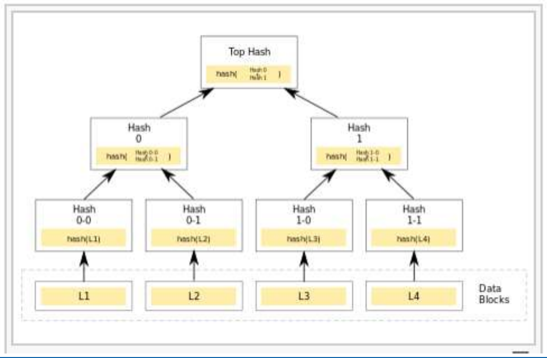
Execution of get() & put() Operations

- **Client can route request directly to coordinator OR to node based on load info, who will send to coordinator**
- **Quorum system: need R nodes to read, W to write**
  - R+W > N, where N is replication factor
- **Return all versions that are causally unrelated (by VCs)**
  - Also do read repair of stale versions
- **For availability, use "sloppy quorum"**
  - Send to first N healthy nodes
  - Hinted handoff: If B is down, send to E instead with hint that belongs to B
  - E sends to B when B recovers



Replica Synchronization

- **Node maintains Merkle tree for each key range it hosts**



Membership & Failure Detection

- **Explicit command for adding or removing nodes from ring**
- **Gossip-based protocol propagates membership changes**
  - Each node contacts a random peer every second
  - Seed nodes help avoid logical partitions
- **Local view of failures suffices**

Latency Optimization

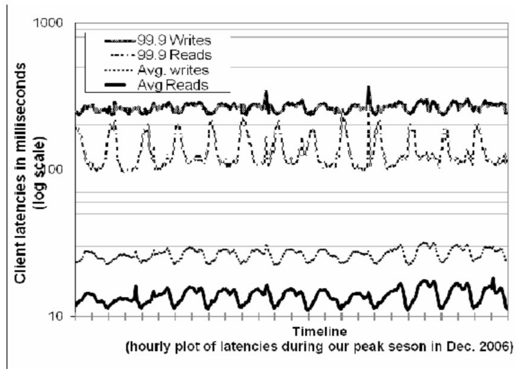
- Coordinator for a write is the node that replied fastest to the previous read operation
- Also, increases chance of "read-your-writes" consistency

Popular Configurations

- **Business-logic-specific reconciliation**
  - E.g., Shopping cart
- **Timestamp-based reconciliation: "last write wins"**
  - E.g., Customer session information
- **High performance read engine**
  - E.g., Product catalog, Promotional items

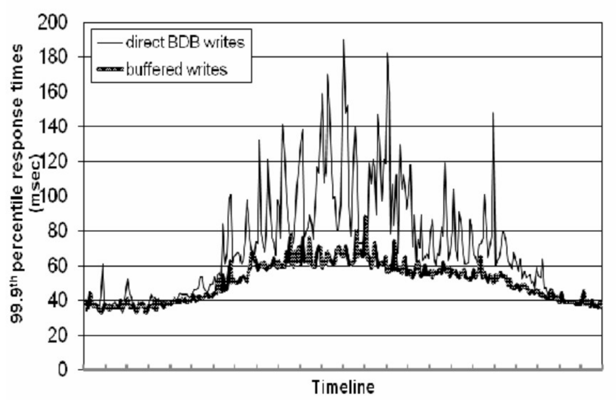
Availability/Performance trade-off
- Typical services use N=3 replicas
- Common to use N=3, R=2, W=2

Read/Write Latency

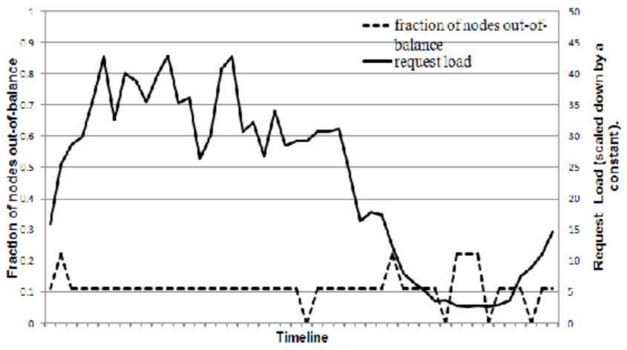


Couple hundred nodes with (3,2,2) configuration

Benefits of Buffering Writes

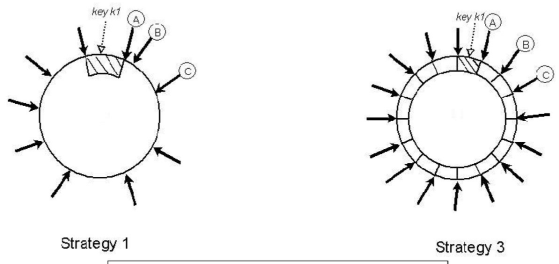


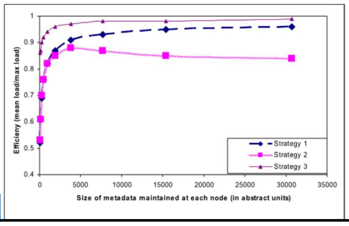
Fraction of Nodes Out-of-Balance



load deviation threshold = 15%

Partitioning & Placement Strategies





Divergent Versions

- **Good metric for consistency: number of divergent versions seen by the application in production environment**
  - From failures
  - From concurrent writes to a single data item
- **Shopping cart service: 99.94% of requests saw 1 version**
  - Blame robots

Server-driven vs. Client-driven Coordination

- **Server-driven: Load balancer assigns each client read request to a random node that acts as coordinator**
- **Client-driven: Client caches membership state (refresh by polling random node every 10 secs). Coordinates reads locally. Sends write requests to preference list. Avoids extra network hop of going to random node**

	99.9th percentile read latency (ms)	99.9th percentile write latency (ms)	Average read latency (ms)	Average write latency (ms)
Server-driven	68.9	68.5	3.9	4.02
Client-driven	30.4	30.4	1.55	1.9

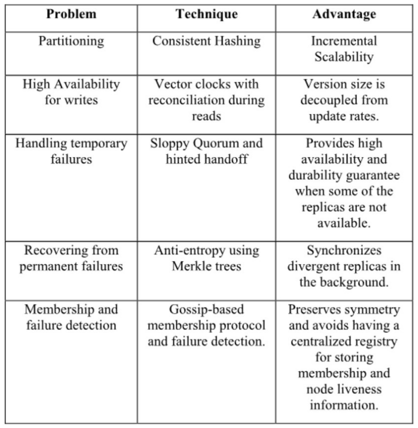
Background vs. Foreground Tasks

- **Use admission control on background tasks**
- **Feedback mechanism determines admitting rate**

Discussion

- **Availability (in 2 years of production runs)**
  - Applications have received successful responses (without timing out) for 99.9995% of requests
  - No data loss event
- **Key feature: Tunable (N,R,W)**
  - Requires tuning to get right
- **Scalability challenge**
  - Each node has a full routing table for all data
  - Could introduce hierarchical extensions to Dynamo

Dynamo Techniques & 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.

Wednesday’s Papers

Memory Coherence in Shared Virtual Memory Systems
Kai Li and Paul Hudak

SigOps HoF paper

Scaling Distributed Machine Learning with the Parameter Server
Mu Li, David Andersen, Jun Woo Park, Alexander Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene Shekita, Bor-Yiing Su

OSDI’14