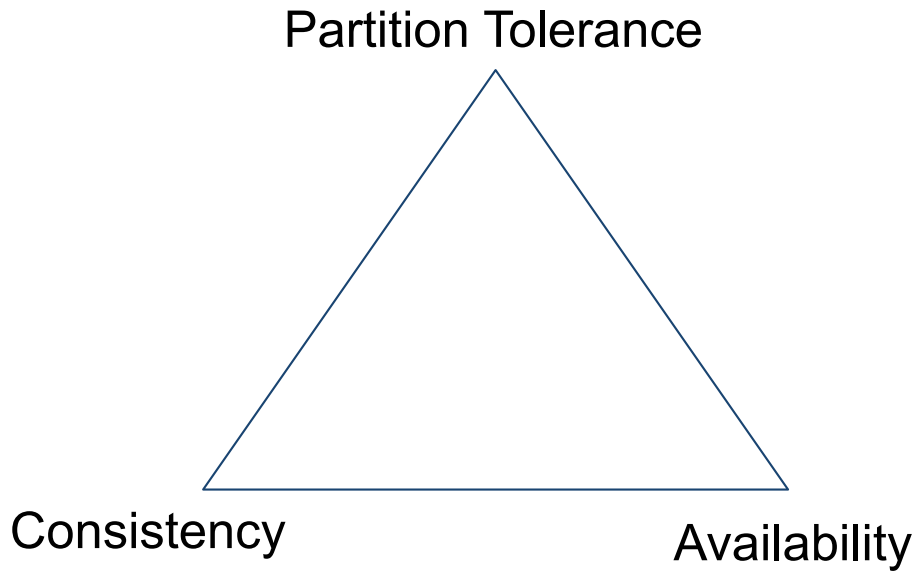
Key-Value Stores

Davide Frey
WIDE Team
Inria

Key Motivation

- CAP theorem
  - [Conjectured by Brewer in 2000]
  - [Proven true by Lynch and Gilbert in 2002]



No SQL

- Simpler Interface than SQL
- Only access by primary key
- No complex query operations
- Goals
  - Elasticity
  - Scalability
  - Fault Tolerance
  - Partition Tolerance

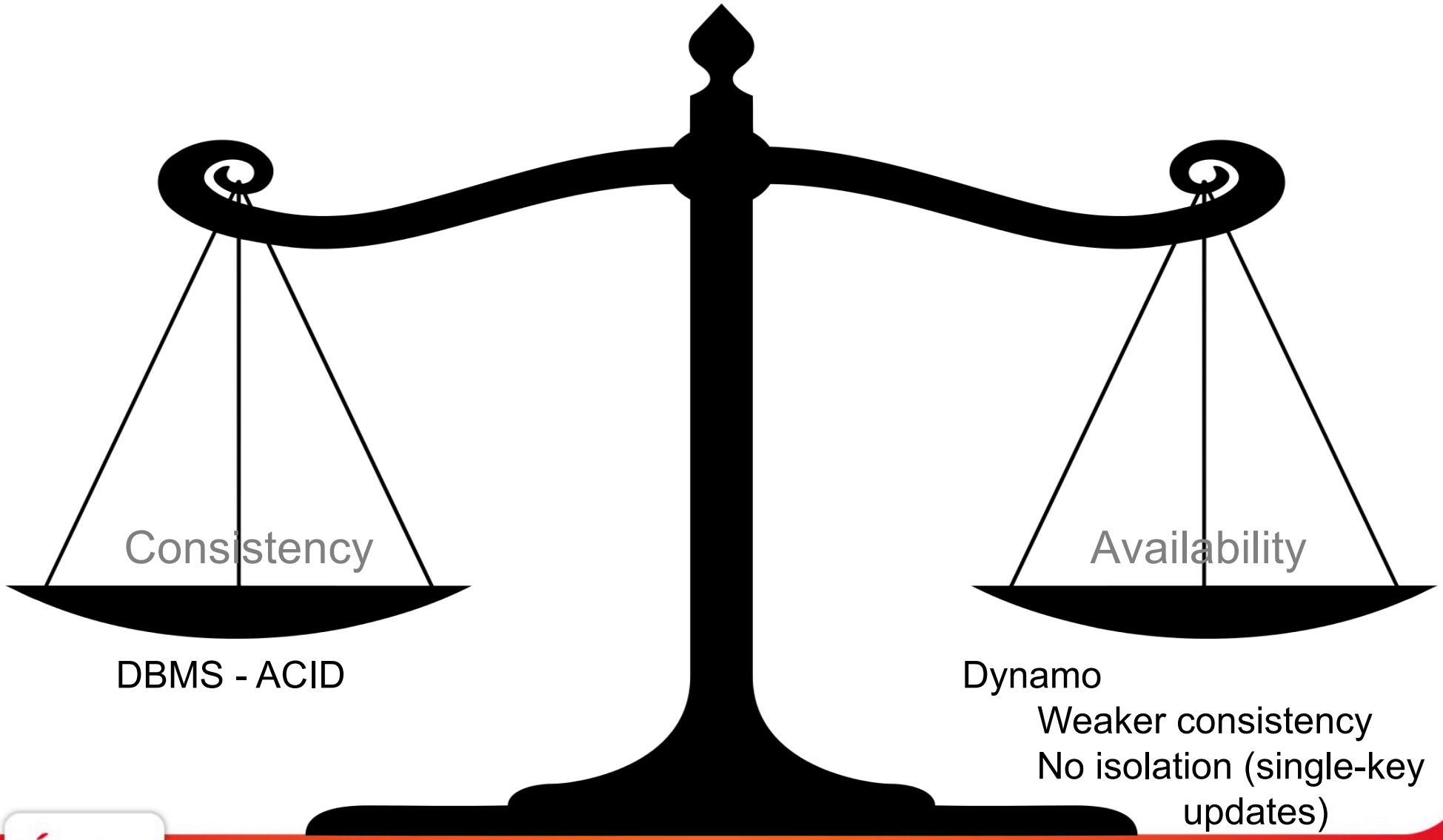
Amazon Dynamo

- Partition and replicate
  - Consistent Hashing
  - Similar to DHT
- Consistency Management
  - Quorum-system
  - Object versioning
  - Decentralized replica synchronization
- Failure detection and membership
  - Gossip

Dynamo's Assumptions

- Objects identified by a Key.
- Read / Write operations
- Small objects <1MB
- Run on commodity hardware
- Trusted environment

Key Trade-Off



DBMS - ACID

Dynamo

Weaker consistency

No isolation (single-key updates)

Performance Goal

- 99.9th Percentile SLA
- Average or Median not enough
- Example
  - 300ms response time for 99.9% of requests given peak load of 500 req/sec

Eventually Consistent

- Always writable
  - As opposed to conflict avoidance
- Conflict resolution at reads
  - Mostly after reads by the application
  - If done by the data store: last update wins
- Data eventually reaches all replicas

Key Principles

- Eventual Consistency
- Incremental Scalablility
- Symmetry
- Decentralization
- Heterogeneity

Dynamo & Peer-To-Peer Techniques

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.

Table from [DeCandia et al. 2007]

But no routing: Zero-Hop DHT

System Interface

- Interface
  - Get(key) -> {(object, context)}
  - Put(key, context, value)
- Context encodes internal information such as object version
- MD5(Key) -> 128-bit Identifier -> storage node -> Disk

Dynamo Details

- Partitioning
- Replication
- Versioning
- Membership
- Failure Handling
- Scaling

Partitioning

- Consistent Hashing
  - Each node takes random position
  - Hash (key) -> position
  - Store on node following key
- Dynamo's variant
  - Multiple points per node
    - Virtual nodes (tokens)
    - More uniform load
    - Capacity -> #virtual nodes

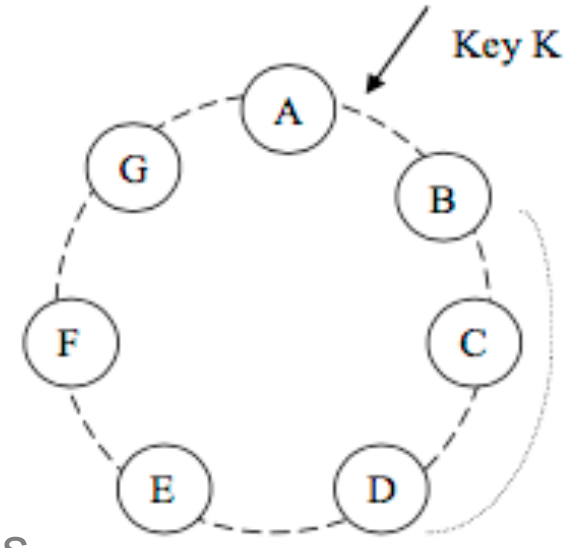


Image from [DeCandia et al. 2007]

Replication

- Replicate each object instance on N replicas
- Coordinator (responsible node) replicates on N-1 nodes that follow
- Skip positions to have distinct physical nodes.

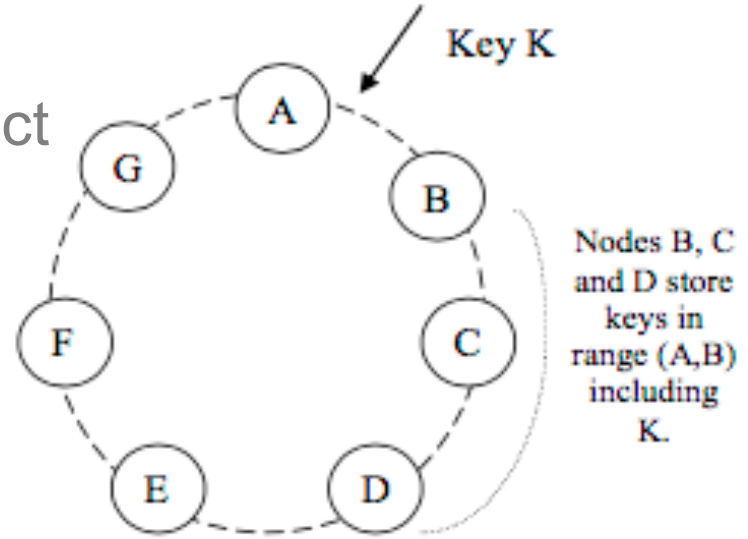
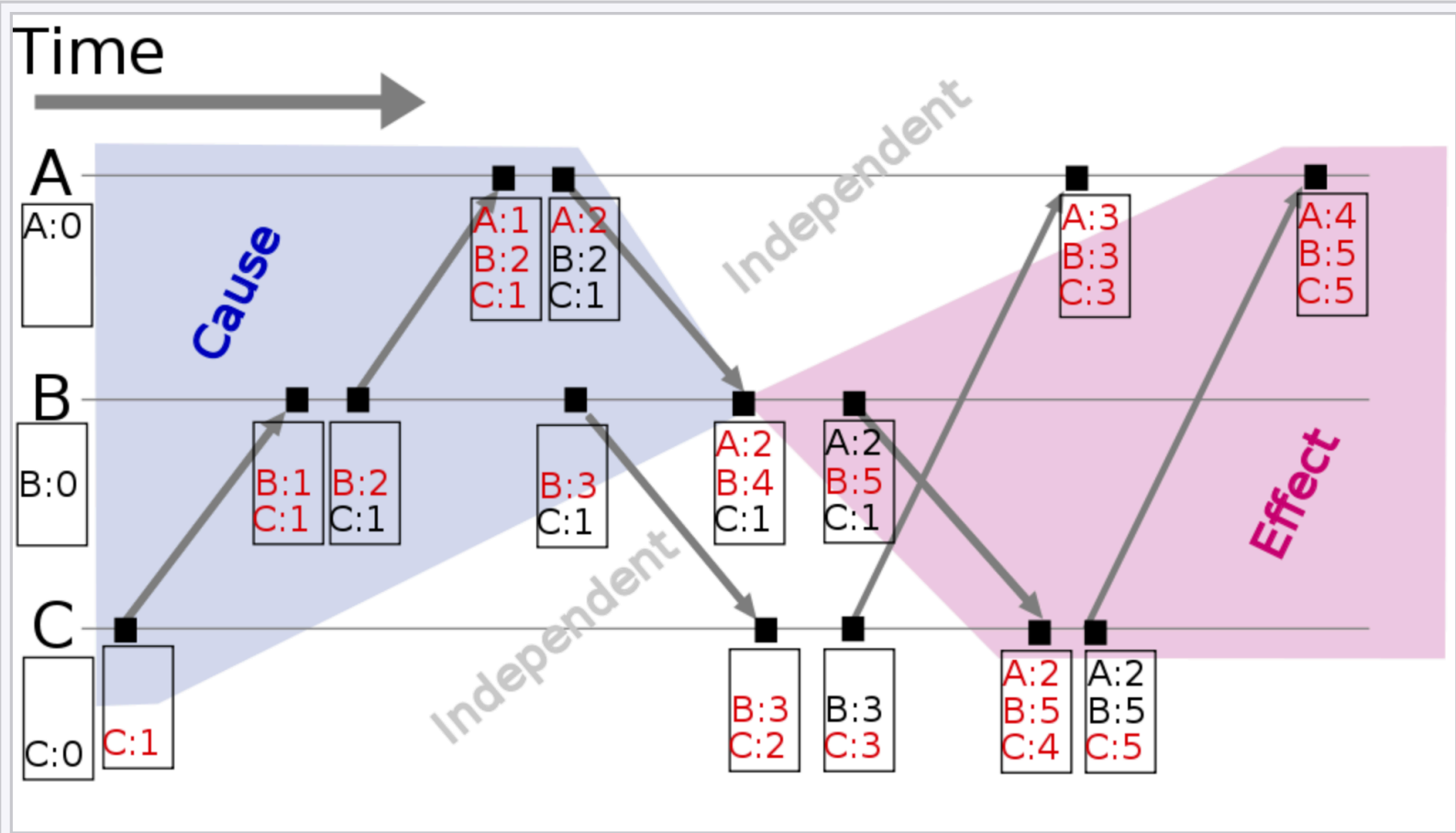


Image from [DeCandia et al. 2007]

Versioning

- Eventual consistency -> asynchronous updates
  - Dynamo maintains multiple versions of each object
  - E.g. multiple versions of shopping cart
  - Use Vector clocks to establish order of updates
    - Concurrent
    - Causally related
  - Client encodes version in context
    - Put (key, context, object)
  - Client reconciles conflicting versions

Vector Clock



Example of a system of vector clocks. Events in the blue region are the causes leading to event B4, whereas those in the red region are the effects of event B4

Diagram from wikipedia

Operation Execution

- Clients access nodes
  - Through load balancer
  - Through a library that determines appropriate node for key
- Coordinator (one of the top N nodes following key)
  - Read and write from/to first N healthy nodes
    - Min W responses for writes
    - Min R responses for reads
    - W+R>N

Quorum

- Read and write from/to first N healthy nodes
  - Min W responses for writes
  - Min R responses for reads
  - W+R>N

Guarantees an intersection between read set and write set

But may not work in case of partitions

Sloppy Quorum

- Send update to the first N “healthy” nodes
  - nodes may receive update not for them
- Hinted Hand-off
  - Updates contain hint for “right recipient”
  - Hand off data to right recipient when available
- Works well for transient failures
- Additionally: make sure object across datacenters

Replica Synchronization

- Use Merkle tree and Anti-Entropy Gossip
  - Exchange merkle hashes
    - starting from root
    - Descend towards children if necessary
  - Effectively identify out-of-sync data
- One separate Merkle Tree for each Key range

Membership Maintenance

- Special case of RPS
  - Dynamo maintains full view
  - One-exchange -> multiple purposes
    - Partitioning
    - Membership
- External discovery mechanism for a few seed nodes
    - A starts a network
    - B starts a network
    - A and B communicate externally
- Reconcile partitioning upon node addition-removal

Google's BigTable

- Distributed multidimensional sorted map
- BT(row: **string**, column: **string**, timestamp: **int**) -> **String**
    - Read/Write: Atomic under single row key
    - Sorted by row key
    - Rows grouped in ranges: tablets
    - Columns grouped in families

Big Table's Architecture

- Master node stores location information
- Tablet servers store the actual data
- Replication for fault tolerance (Chubby lock service)
- A 1-hop P2P DHT with additional features
  - Multidimensional
  - Fault tolerance
  - Atomic row access

Facebook/Apache Cassandra

- Multi dimensional
- 0-hop DHT-like
- Simple API
  - insert (table, key, rowMutation)
  - get (table, key, columnName)
  - delete(table,key,columnName)
- Consistent Hashing Improvement
  - Lightly loaded nodes move to loaded areas

    inspired by [Chord DHT]
- Virtual Nodes
  - Initial versions: no virtual nodes
  - From v1.2 Virtual nodes
  - From v3, smart choice of virtual node Ids

Replication in Cassandra

- Responsible node replicates on N-1 other hosts
  - Rack Unaware
    - N-1 nodes that follow
  - Rack Aware
    - Based on leader
  - Datacenter Aware
    - Based on leader
  - Leader election through ZooKeeper

Membership in Cassandra

- Anti-entropy gossip
  - ScuttleButt
  - Everyone knows about everyone's position in ring
- Probabilistic Failure Detection
  - Accrual Failure Detector
  - Avoid communicating with unreachable nodes
  - Only for temporary failures
- Manual mechanism for addition removal

Bibliography

- Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. **Dynamo**: amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles* (SOSP '07). ACM, New York, NY, USA, 205-220. DOI=http://dx.doi.org/10.1145/1294261.1294281
- Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2008. **Bigtable**: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.* 26, 2, Article 4 (June 2008), 26 pages. DOI=http://dx.doi.org/10.1145/1365815.1365816
- Avinash Lakshman and Prashant Malik. 2010. Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.* 44, 2 (April 2010), 35-40. DOI=http://dx.doi.org/10.1145/1773912.1773922

Bibliography

- Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, and Benjamin Reed. 2010. ZooKeeper: wait-free coordination for internet-scale systems. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference* (USENIXATC'10). USENIX Association, Berkeley, CA, USA, 11-11.
- Mike Burrows. 2006. The Chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation* (OSDI '06). USENIX Association, Berkeley, CA, USA, 335-350
- Robbert van Renesse, Dan Dumitriu, Valient Gough, and Chris Thomas. 2008. Efficient reconciliation and flow control for anti-entropy protocols. In *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware* (LADIS '08). ACM, New York, NY, USA, , Article 6 , 7 pages. DOI=http://dx.doi.org/10.1145/1529974.1529983