CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

KEY-VALUE STORES NoSQL

Lecture B

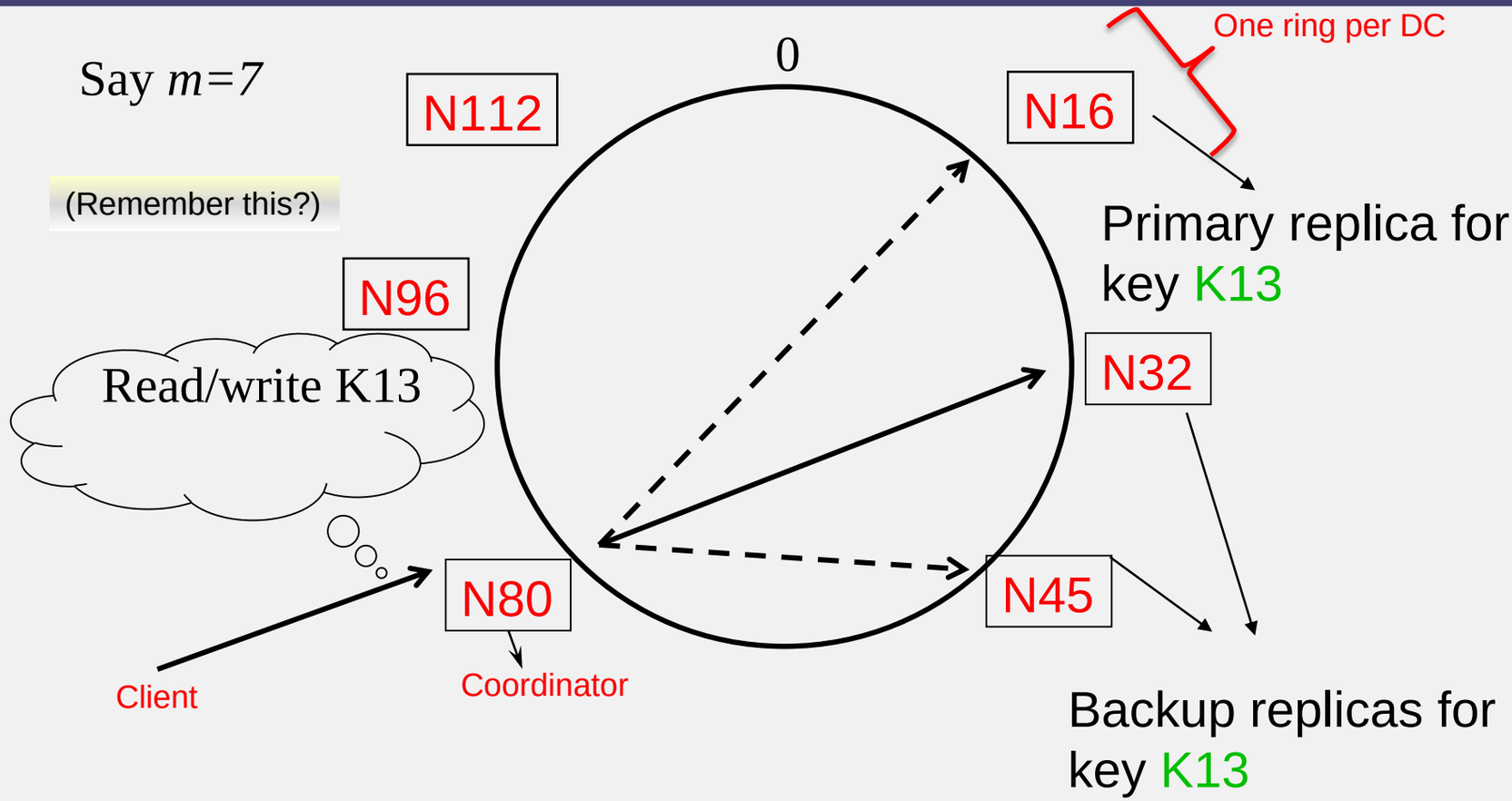
CASSANDRA

Cassandra

- A distributed key-value store
- Intended to run in a datacenter (and also across DCs)
- Originally designed at Facebook
- Open-sourced later, today an Apache project
- Some of the companies that use Cassandra in their production clusters
  - IBM, Adobe, HP, eBay, Ericsson, Symantec
  - Twitter, Spotify
  - PBS Kids
  - Netflix: uses Cassandra to keep track of your current position in the video you're watching

Let's go Inside Cassandra: Key -> Server Mapping

- How do you decide which server(s) a key-value resides on?



Cassandra uses a ring-based DHT but without finger tables or routing

Key→server mapping is the "Partitioner"

Data Placement Strategies

- Replication Strategy: two options:
  1. *SimpleStrategy*
  2. *NetworkTopologyStrategy*

1. SimpleStrategy: uses the Partitioner, of which there are two kinds
   1. *RandomPartitioner*: Chord-like hash partitioning
   2. *ByteOrderedPartitioner*: Assigns ranges of keys to servers.
      - Easier for *range queries* (e.g., get me all twitter users starting with [a-b])
2. NetworkTopologyStrategy: for multi-DC deployments
   - Two replicas per DC
   - Three replicas per DC
   - Per DC
     - First replica placed according to Partitioner
     - Then go clockwise around ring until you hit a different rack

Snitches

- Maps: IPs to racks and DCs. Configured in cassandra.yaml config file
- Some options:
  - SimpleSnitch: Unaware of Topology (Rack-unaware)
  - RackInferring: Assumes topology of network by octet of server's IP address
    - 101.201.301.401 = x.<DC octet>.<rack octet>.<node octet>
  - PropertyFileSnitch: uses a config file
  - EC2Snitch: uses EC2
    - EC2 Region = DC
    - Availability zone = rack
- Other snitch options available

Writes

- Need to be lock-free and fast (no reads or disk seeks)
- Client sends write to one coordinator node in Cassandra cluster
  - Coordinator may be per-key, per-client, or per-query
  - Per-key Coordinator ensures writes for the key are serialized
- Coordinator uses Partitioner to send query to all replica nodes responsible for key
- When X replicas respond, coordinator returns an acknowledgement to the client
  - X? We'll see later.

Writes (2)

- Always writable: Hinted Handoff mechanism
  - If any replica is down, the coordinator writes to all other replicas, and keeps the write locally until down replica comes back up.
  - When all replicas are down, the Coordinator (front end) buffers writes (for up to a few hours).
- One ring per datacenter
  - Per-DC coordinator elected to coordinate with other DCs
  - Election done via Zookeeper, which runs a Paxos (consensus) variant
    - Paxos: elsewhere in this course

Writes at a replica node

On receiving a write

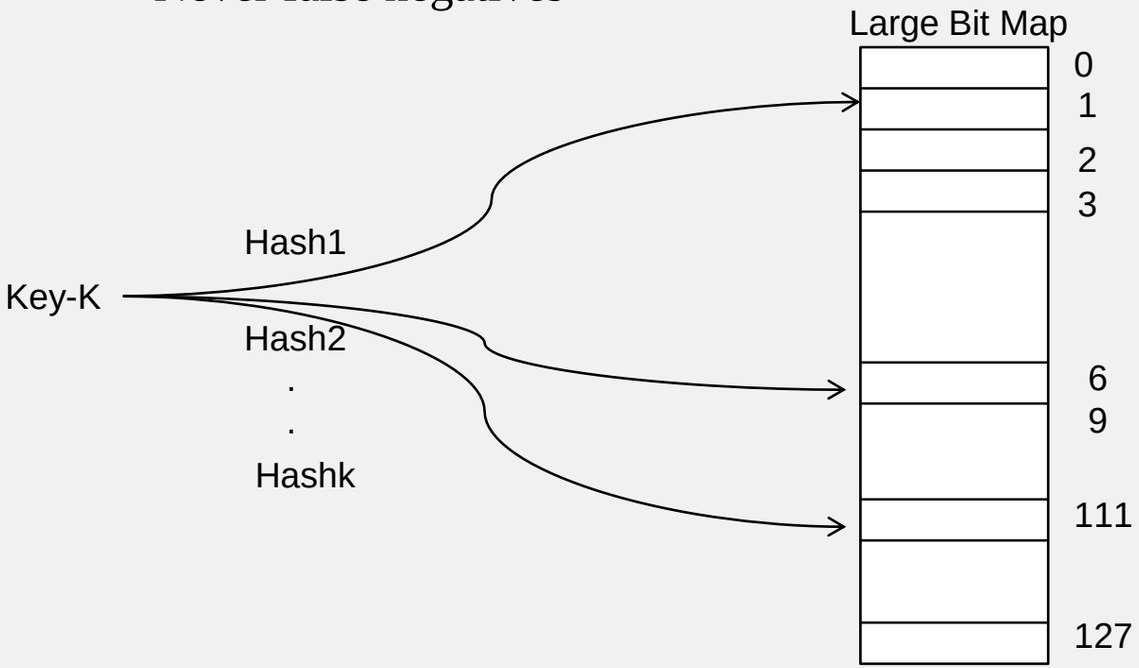
1. Log it in disk commit log (for failure recovery)
2. Make changes to appropriate memtables
    - **Memtable** = In-memory representation of multiple key-value pairs
    - Cache that can be searched by key
    - Write-back cache as opposed to write-through

Later, when memtable is full or old, flush to disk

- Data file: An **SSTable** (Sorted String Table) – list of key-value pairs, sorted by key
- Index file: An SSTable of (key, position in data sstable) pairs
- And a Bloom filter (for efficient search) – next slide

Bloom Filter

- Compact way of representing a set of items
- Checking for existence in set is cheap
- Some probability of false positives: an item not in set may check true as being in set
- Never false negatives



On insert, set all hashed bits.

On check-if-present, return true if all hashed bits set.

- False positives

False positive rate low

- k=4 hash functions
- 100 items
- 3200 bits
- FP rate = 0.02%

Compaction

Data updates accumulate over time and SStables and logs need to be compacted

- The process of compaction merges SSTables, i.e., by merging updates for a key
- Run periodically and locally at each server

Deletes

Delete: don’t delete item right away

- Add a **tombstone** to the log
- Eventually, when compaction encounters tombstone it will delete item

Reads

Read: Similar to writes, except

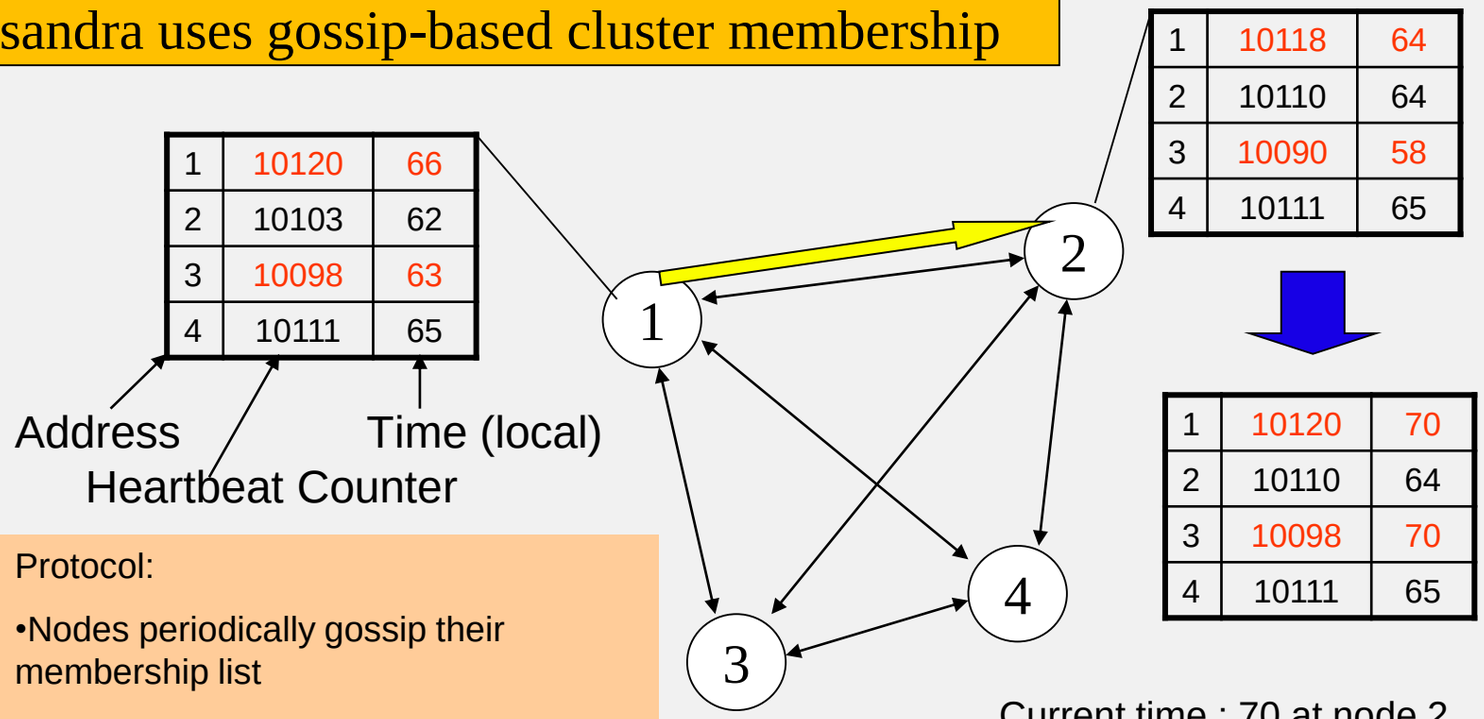
- Coordinator can contact X replicas (e.g., in same rack)
  - Coordinator sends read to replicas that have responded quickest in past
  - When X replicas respond, coordinator returns the latest-timestamped value from among those X
  - (X? We'll see later.)
- Coordinator also fetches value from other replicas
  - Checks consistency in the background, initiating a **read repair** if any two values are different
  - This mechanism seeks to eventually bring all replicas up to date
- A row may be split across multiple SSTables => reads need to touch multiple SSTables => reads slower than writes (but still fast)

Membership

- Any server in cluster could be the coordinator
- So every server needs to maintain a list of all the other servers that are currently in the server
- List needs to be updated automatically as servers join, leave, and fail

Cluster Membership – Gossip-Style

Cassandra uses gossip-based cluster membership



Protocol:

- Nodes periodically gossip their membership list
- On receipt, the local membership list is updated, as shown
- If any heartbeat older than Tfail, node is marked as failed

Current time : 70 at node 2

(asynchronous clocks)

(Remember this?)

Suspicion Mechanisms in Cassandra

- Suspicion mechanisms to adaptively set the timeout based on underlying network and failure behavior
- Accrual detector: Failure detector outputs a value (PHI) representing suspicion
- Apps set an appropriate threshold
- PHI calculation for a member
  - Inter-arrival times for gossip messages
  - PHI(t) =
    - log(CDF or Probability(t_now – t_last))/log 10
  - PHI basically determines the detection timeout, but takes into account historical inter-arrival time variations for gossiped heartbeats
- In practice, PHI = 5 => 10-15 sec detection time

Cassandra Vs. RDBMS

- MySQL is one of the most popular (and has been for a while)
- On > 50 GB data
- MySQL
  - Writes 300 ms avg
  - Reads 350 ms avg
- Cassandra
  - Writes 0.12 ms avg
  - Reads 15 ms avg
- Orders of magnitude faster
- What's the catch? What did we lose?