Cassandra

A Decentralized Structured Storage System
Motivation

• Facebook Inbox search:
  – Billions of write per day
  – Geographical distribution of servers and users
Data Model

• A table is a distributed multi-dimensional map indexed by a key

• Columns are grouped together into sets called column families
API

- `insert(table, key, rowMutation)`
- `get(table, key, columnName)`
- `insert(table, key, columnName)`
System Architecture: Partitioning

- Partitions data across the cluster using consistent hashing
- Each node in the system is assigned a random value on the ring space
- A data item belong on the first node with a position larger than the item’s position
- Only direct neighbour affected by a node
- Incoming node alleviate heavily loaded nodes
System Architecture: Replication

- Each data item is replicated at N hosts
- Coordinator node is in charge of the replication of the data
- “Rack Unaware”: use N-1 successors
- “Rack Aware” or “Data Centre Aware”: nodes elect a leader who assigns a replica range to every node
System Architecture: Membership

• Membership is based on Scuttlebutt: an anti-entropi Gossip based mechanism
• Use Failure detection to avoid attempts to communicate with unreachable nodes
System Architecture: Bootstrapping

• When a node starts for the first time, it chooses a random token for its position in the ring
• This information is then gossiped
• When a node needs to join the cluster, it reads its configuration file which contains a few contact points within the cluster
System Architecture: Scaling

• When a new node is added, it gets assigned a token such that it can alleviate a heavily loaded node.
System Architecture: Local Persistence

• Write:
  – Use an in-memory data structure
  – Write to in-memory only performed after successful write into a commit log
  – When the in-memory data structure goes over a threshold, it dumps itself to disk

• Read:
  – First look at in-memory data
  – Then check a bloom filter for each file in which the key could be