NoSQL/Key-value Stores

Lecture 20

Based mostly on
- Cassandra NoSQL presentation
- Cassandra 1.0 documentation at datastax.com
- Cassandra Apache project wiki
- HBase

© 2012, I. Gupta
Cassandra

- Originally designed at Facebook
- Open-sourced
- Some of its myriad users:

- Adobe
- AppScale
- IBM
- Symantec
- Twitter
- eBay
- Spotify
- Netflix
- Ericsson

- With this many users, one would think
  - Its design is very complex
  - We in our class won’t know anything about its internals
  - Let’s find out!
Why Key-value Store?

- (Business) Key -> Value
- (twitter.com) tweet id -> information about tweet
- (kayak.com) Flight number -> information about flight, e.g., availability
- (yourbank.com) Account number -> information about it
- (amazon.com) item number -> information about it

- Search is usually built on top of a key-value store
Isn’t that just a database?

- Yes
- Relational Databases (RDBMSs) have been around for ages
- MySQL is the most popular among them
- Data stored in tables
- Schema-based, i.e., structured tables
- Queried using SQL

Example's Source
Issues with today’s workloads

• Data: Large and unstructured
• Lots of random reads and writes
• Foreign keys rarely needed
• Need
  – Incremental Scalability
  – Speed
  – No Single point of failure
  – Low TCO and admin
  – Scale out, not up
**CAP Theorem**

- Proposed by Eric Brewer (Berkeley)
- Subsequently proved by Gilbert and Lynch
- In a distributed system you can satisfy at most 2 out of the 3 guarantees
  1. Consistency: all nodes have same data at any time
  2. Availability: the system allows operations all the time
  3. Partition-tolerance: the system continues to work in spite of network partitions

- **Cassandra**
  - Eventual (weak) consistency, Availability, Partition-tolerance

- **Traditional RDBMSs**
  - Strong consistency over availability under a partition
Cassandra Data Model

- Column Families:
  - Like SQL tables
  - but may be unstructured (client-specified)
  - Can have index tables
- Hence “column-oriented databases”/“NoSQL”
  - No schemas
  - Some columns missing from some entries
  - “Not Only SQL”
  - Supports get(key) and put(key, value) operations
  - Often write-heavy workloads
Let’s go Inside: Key -> Server Mapping

- How do you decide which server(s) a key-value resides on?
Cassandra uses a Ring-based DHT but without routing.

Say $m=7$

Read/write $K_{13}$

Primary replica for key $K_{13}$

Backup replicas for key $K_{13}$

Coordinator (typically one per DC)
 Writes

• Need to be lock-free and fast (no reads or disk seeks)

• Client sends write to one front-end node in Cassandra cluster (Coordinator)

• Which (via Partitioning function) sends it to all replica nodes responsible for key
  – Always writable: Hinted Handoff
    » If any replica is down, the coordinator writes to all other replicas, and keeps the write until down replica comes back up.
    » When all replicas are down, the Coordinator (front end) buffers writes (for up to an hour).
  – Provides Atomicity for a given key (i.e., within ColumnFamily)

• One ring per datacenter
  – Coordinator can also send write to one replica per remote datacenter
On receiving a write

1. log it in disk commit log
2. Make changes to appropriate memtables
   - In-memory representation of multiple key-value pairs

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 – (key, position in data sstable) pairs
  » And a Bloom filter

Compaction: Data updates accumulate over time and sstables and logs need to be compacted
- Merge key updates, etc.

Reads need to touch log and multiple SSTables
- May be slower than writes
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%
Deletes and Reads

- **Delete**: don’t delete item right away
  - add a tombstone to the log
  - Compaction will remove tombstone and delete item

- **Read**: Similar to writes, except
  - Coordinator can contact closest replica (e.g., in same rack)
  - Coordinator also fetches from multiple replicas
    » check consistency in the background, initiating a read-repair if any two values are different
    » Makes read slower than writes (but still fast)
    » Read repair: uses gossip (remember this?)
Cassandra uses Quorums

(Remember this?)

• **Reads**
  - Wait for R replicas (R specified by clients)
  - In background check for consistency of remaining N-R replicas, and initiate read repair if needed (N = total number of replicas for this key)

• **Writes come in two flavors**
  - Block until quorum is reached
  - Async: Write to any node

• **Quorum Q = N/2 + 1**

• **R = read replica count, W = write replica count**

• If W+R > N and W > N/2, you have consistency

• Allowed (W=1, R=N) or (W=N, R=1) or (W=Q, R=Q)
Cassandra uses Quorums

• In reality, a client can choose one of these levels for a read/write operation:
  – ANY: any node (may not be replica)
  – ONE: at least one replica
  – QUORUM: quorum across all replicas in all datacenters
  – LOCAL_QUORUM: in coordinator’s DC
  – EACH_QUORUM: quorum in every DC
  – ALL: all replicas all DCs
Cluster Membership

Protocol:
- Nodes periodically gossip their membership list.
- On receipt, the local membership list is updated.

Cassandra uses gossip-based cluster membership.
Cluster Membership, contd.

- Suspicion mechanisms
- Accrual detector: FD outputs a value (PHI) representing suspicion
- Apps set an appropriate threshold
- PHI = 5 => 10-15 sec detection time
- 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 is sensitive to actual inter-arrival time variations for gossiped heartbeats

Cassandra uses gossip-based cluster membership
Vs. SQL

- MySQL is 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
Cassandra Summary

• While RDBMS provide ACID (Atomicity Consistency Isolation Durability)
• Cassandra provides BASE
  – Basically Available Soft-state Eventual Consistency
  – Prefers Availability over consistency
• Other NoSQL products
  – MongoDB, Riak (look them up!)
• Next: HBase
  – Prefers (strong) Consistency over Availability
HBase

- Google’s BigTable was first “blob-based” storage system
- Yahoo! Open-sourced it -> HBase
- Major Apache project today
- Facebook uses HBase internally
- API
  - Get/Put(row)
  - Scan(row range, filter) – range queries
  - MultiPut
HBase Architecture

Small group of servers running Zab, a Paxos-like protocol

HBase Storage hierarchy

- **HBase Table**
  - Split it into multiple regions: replicated across servers
    » One Store per ColumnFamily (subset of columns with similar query patterns) per region
      • Memstore for each Store: in-memory updates to Store; flushed to disk when full
        - StoreFiles for each store for each region: where the data lives
          - Blocks

- **HFile**
  - SSTable from Google’s BigTable
HFile

Source: http://blog.cloudera.com/blog/2012/06/hbase-io-hfile-input-output/
Strong Consistency: HBase Write-Ahead Log

1. Write to HLog before writing to MemStore
2. Can recover from failure

Log Replay

- **After recovery from failure, or upon bootup (HRegionServer/HMaster)**
  - Replay any stale logs (use timestamps to find out where the database is w.r.t. the logs)
  - Replay: add edits to the MemStore

- **Why one HLog per HRegionServer rather than per region?**
  - Avoids many concurrent writes, which on the local file system may involve many disk seeks
**Cross-data center replication**

Zookeeper actually a file system for control information:
1. /hbase/replication/state
2. /hbase/replication/peers /<peer cluster number>
3. /hbase/replication/rs/<hlog>
Summary

- Key-value stores and NoSQL faster but provide weaker guarantees

- MP3: By now, you must have a basic working system (may not yet satisfy all the requirements)

- HW3: due next Tuesday

- Free Flu shot in Grainger Library today 3.30-6.30 pm – take your id card