Today

Cassandra & Dynamo

If you reloaded a page (Amazon / Gmail) which you prefer?

- [ ] an empty / blank draft
- [ ] broken site / No page

looks better
- can still do other things
- you have hope
- can you do the work

happy user (I do)

Wouldn't add items twice
- no silent failures
- 2x emails being sent

Honesty
Always writeable

Users can always interact with "system"

Dynamo / Cassandra are a storage layer

all data = Keys/Values

use this in Hash to determine the server

pick random server

maintains a list of all other servers
Dealing with uneven load in CHash

Cassandra

1. Coordinator gets metrics
2. Coordinator makes decisions
   - depending on the load
3. Coordinator moves servers around the key space

Dynamo (Virtual Node)

1. Each server is assigned tokens
2. Tokens = \text{func}(H/w)
   - bigger CPU = more tokens
3. Human that manually allocates the tokens

D = \text{Hash(String)} \mod N

- Random & Works
- Probabilistic guaranteed
What does it mean to be always writeable?

Always writeable: notes in a partition self-organize into a new hash-based network. The new server stores a key may have no data about a key (none of items in cart will be on new server).
Conflicts when partition heals?

• during partition heal
  How to deal w/ conflicts?

- me on laptop
- during partition each client keeps writing => leads to conflicts
  1. items in shopping cat
  2. set title for some content
  3. add comments/likes
Inspired by lazy replication

1. read & get VC
2. write & updates VC
3. conflict = use VC to break conflicts

Conflicts with time

"last writer wins"

\[ V = 5 \text{ (1 pm)} \]

\[ V = 7 \text{ (7 pm)} \]

If updates are concurrent then you merge "the cots" may lead to duplicate items

Cassandra: most recent time stamps win as the final version
Conflict resolution

1) partition heads

2) conflicting data is stored on servers

3) during the next read: conflict resolution gets triggered

last writer wins which a general approach

based on version clocks and this needs app-specific logic to address "merge"
Dirty Quorum

Quorum in Raft $\Rightarrow$ majority ($\frac{N}{2} + 1$) of $N$ required for write

always replicates to $N$ codes (node in charge of a key & the next $N-1$ nodes in hash key space)

The $N$ nodes can be any $N$ nodes

$R/W/N$ quorum

# of replies before read is done

# of replies before write is done

# of nodes to store
You only need one node to complete a read.

Done data is on one node.

Try to get data on 3 nodes.

O: Inconsistent data for other clients.
O: Data is lost.

Speed vs. Consistency.
Versions of quorum

- **full**
  - all $N$ nodes must respond

- **majority**
  - only a majority must respond

- **dirty**
  - where the developer configures & decides how many should respond

---

- consistent
- blazingly fast
- slowest
- one failure halts the system
- as fast as the slowest node
Dynamo / Cassandra

1. Partition data / load balance
2. Replicating data
3. Conflict resolution
4. Membership

\[ \Rightarrow \text{Consistent hashing} \]
\[ \Rightarrow \text{dirty quorum} \]
\[ \Rightarrow \text{Vector clocks / real time} \]
\[ \Rightarrow \text{gossip} \]