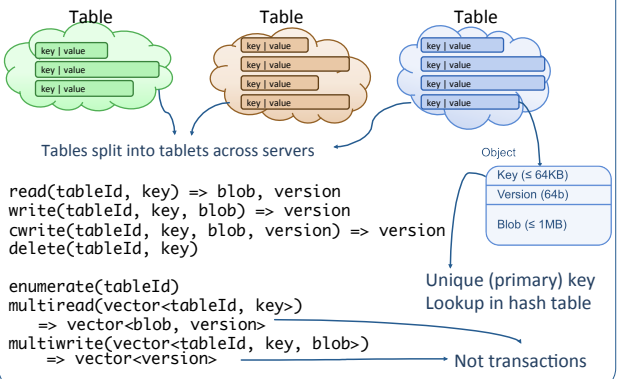
Richer data models for highly-scalable, low-latency datacenter storage systems

Ankita Arvind Kejriwal, John Ousterhout

Current Data Model: Key-Value Store



Richer Data Models

- Currently in preliminary design phase.
- Would love to hear your ideas, cautionary tales and feedback!

Directions

- Multi-object transactions
- Secondary Indexes
- Constraint checking
- High-level operations, code shipping

Transactions

- Optimistic concurrency control if conflicts rare

Example of atomic block

```
X = read(table1, "foo")
write(table2, "bar", blob)
write(table3, "baz", blob)
```

- Client side (using conditional ops and some server-side support) vs. Server side (2PC?)

Constrain Checking

- In a table: Data type, length, null values, uniqueness
- Across tables: Referential integrity

Secondary Indexes

API

- Keys declared at the level of tables, during table creation and can be modified later
- Objects potentially indexed on different keys?
- RAMCloud treats "value" as opaque, provide search keys explicitly

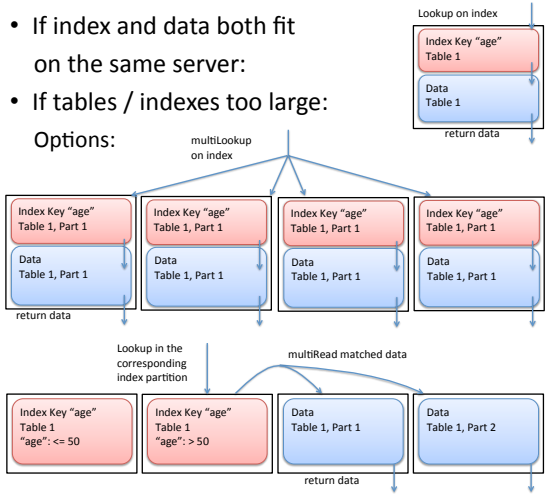
  `<tableId, key0, key1, key2, …, value>`

  Eg.: `<1, id: 101, name: "foo", year: 3, "…">`

Index Location

- If index and data both fit on the same server:
- If tables / indexes too large:

Options:



Index Entry Format

`<index key, obj pointer>`

- Data relocation, + Fixed size pointer

`<index key, primary key>`

+ Data relocation, – Large keys

`<index key, primary key hash>`

+ Data relocation, + Fixed size key hash,

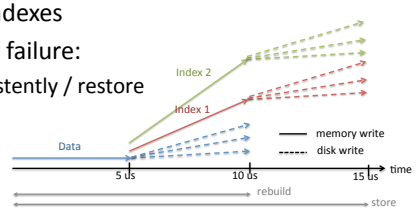
– Complex mechanism for lookup, garbage collection

Index Format

- Hash table: Lookups on exact match
- Tree: Range queries
  - What kind of tree implementation?

Failure / Recovery

- Master failure: Normal crash recovery, no impact on indexes
- Index server failure:
  - Store persistently / restore
  - Rebuild



Consistency

Object Lookup

- Found in index + data matches – return data
- Not in index or no match – return not found

Object Write

- Insert index entries, then write object

⇒ Object available as soon as object written

Object Deletion

- Delete object then remove index entries

⇒ Object not found as soon as object deleted

Object Update

- Insert new index entries, then update object, then remove old index entries

Example: