Table Enumeration in RAMCloud

Elliott Slaughter

What is enumeration?

Enumeration of a table allows a client to iterate through all key/value pairs from the given table stored in RAMCloud. Enumeration might be useful for debugging, for example, or for making offline backups of a live system while it is running. However, since enumeration is not presumed to be a common operation, we must be careful not to burden the system with additional data structures which might impact the latencies of reads and writes in the system.

Potential Pitfalls

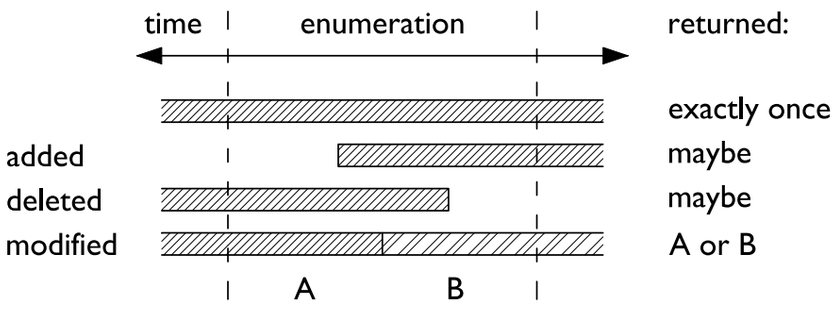
RAMCloud is a distributed store, so we must handle changes in the system during enumeration.

- Table can be modified during enumeration
- Table might be distributed on multiple servers
- Need multiple requests to fetch a table
- Servers can go up and down
- Table can be redistributed across the cluster
- Etc.

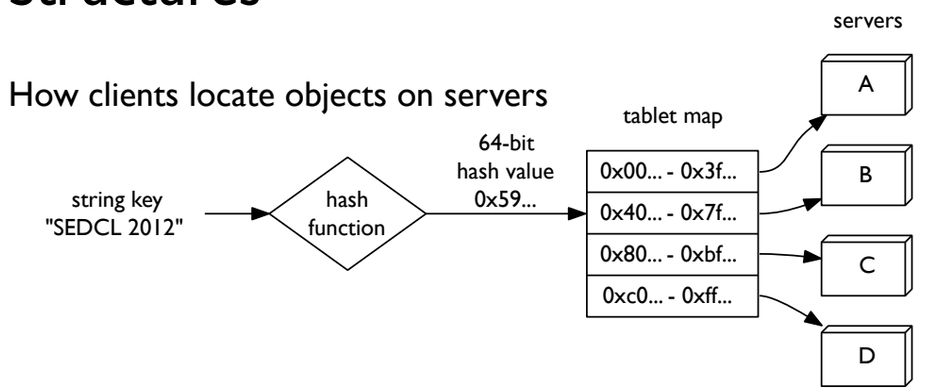
Consistency Model

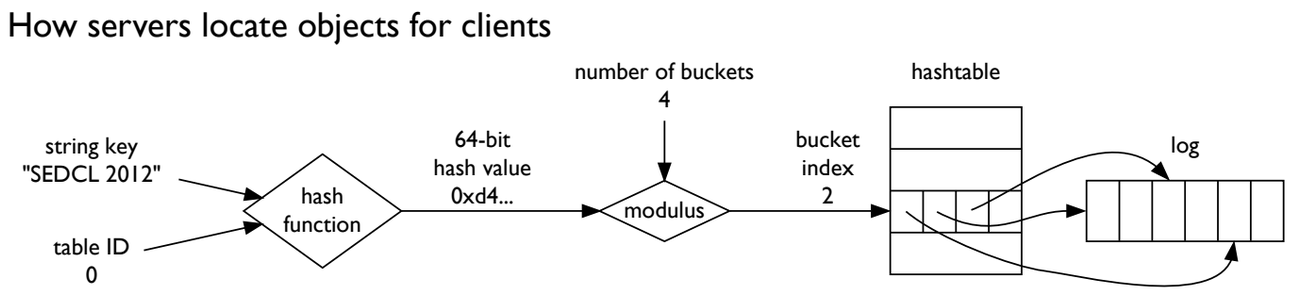
We propose a "once-only" consistency model for enumeration:

- Any object whose lifetime spans the entire enumeration will be returned exactly once
- No object will be returned more than once



Client and Server Data Structures





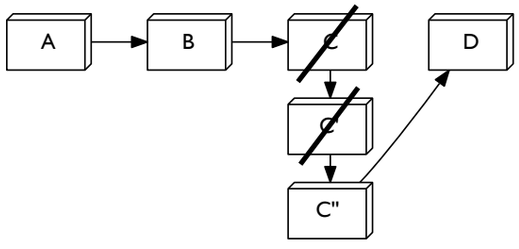
Client algorithm

Since we might need to make multiple round trips between the client and any given server, we need an "iterator" to resume iteration through the portions of a table stored on a server. Supposing this iterator is opaque to the client, we can illustrate the client-side enumeration algorithm as follows. (See "Server algorithm" for a description of what needs to go inside the iterator.)

- iter := empty buffer
- tabletStartHash := 0
- done := false
- While not done:
  - Locate the server that owns the tablet starting at tabletStartHash
  - Get objects, nextIter, nextTabletStartHash from server via EnumerateRPC
  - iter := nextIter
  - tabletStartHash := nextTabletStartHash
  - done := no objects returned and nextTabletStartHash <= tabletStartHash

Server algorithm

The naive algorithm for the server is to iterate through the hash table in bucket order and collect objects. But enumeration might take multiple round trips, so we need to know where to resume iteration. And we need to deal with table reconfiguration at any time.



We make the iterator a stack, where each frame contains sufficient information to specify the state of iteration at a specific machine. When the table configuration changes, we push a frame onto the stack to track the new configuration. We can then use old frames to filter out entries that have already been returned.

```
bucketIndex
nextBucketHash
tabletStartHash    *
tabletEndHash
numBuckets
```