Cluster Management in RAMCloud

Diego Ongaro

Stanford University

RAMCloud Design Review
April 01, 2010

Managing a RAMCloud Cluster

RAMCloud's design requires large clusters

- 16 servers per TB of storage
- More hosts = faster recovery times, faster burst speeds

Configuration Issues

- Where to place objects, and how to find them
- How to balance load
- How masters select backups
- How to detect and recover from failures
- How to bootstrap the cluster, and
  how to restart it after power outages
- How to keep statistics and logs
- How to authenticate hosts and apps

Managing a RAMCloud Cluster

RAMCloud's design requires large clusters

- 16 servers per TB of storage
- More hosts = faster recovery times, faster burst speeds

Configuration Issues

- Where to place objects, and how to find them
- How to balance load
- How masters select backups
- How to detect and recover from failures
- How to bootstrap the cluster, and
  how to restart it after power outages
- How to keep statistics and logs
- How to authenticate hosts and apps

Central Coordinator

- A central coordinator is simple
- Global view useful for locations, load balancing, administration
- We think a single machine can handle it
  - Only tens of thousands of hosts
  - We can service 1 million ops per second, *remember*?
- Coordinator is off the critical path
  - Apps and masters aggressively cache location info
- **Problems:** finding the coordinator, coordinator failures

Alternative: P2P

- Robust
- Duplicates location information at every host
- May make sub-optimal load balancing decisions
- How do machines find each other?

Where to Place Objects

Partition objects by address ranges (tablets)

- Apps touch few hosts to access their workspace
  - Smaller location state to cache
  - Cheaper transactions
- Expect better segment compression
- Need to manually balance load

Alternative: Partition objects by hash ranges

- "Natural" load balancing
- Inefficient to co-locate indexes with their data
- Table enumerate touches all masters
- Server-assigned (auto-increment) object IDs inefficient

Critical Coordinator State

- Tablet Map: ~10K to 10M rows

Workspace	Table	Objects	Master	Load
45	9	0 to ∞	10.0.3.52	10%
72	3	0 to 30M	10.0.9.33	90%
72	3	30M to 50M	10.0.3.52	50%

  - Finding objects
  - Load balancing

Critical Coordinator State

- Tablet Map: ~10K to 10M rows

Workspace	Table	Objects	Master	Load
45	9	0 to ∞	10.0.3.52	10%
72	3	0 to 30M	10.0.9.33	90%
72	3	30M to 50M	10.0.3.52	50%

  - Finding objects
  - Load balancing

- Host List: ~10K rows

Host	Status	Rack
10.0.3.52	Master + Backup	3
10.0.2.17	Hot Spare	2
10.0.9.23	Master + Backup	9

  - Finding available backup servers
  - Load balancing and recovery (find idle hosts)

How to Find the Coordinator

Need some out-of-band channel

- Well-known IP address
  - Successors will take over this address
- Well-known DNS name
  - Pre-determined successors listed under additional addresses for this name
  - DNS is everywhere
  - Issues with caching/TTLs
- Other existing infrastructure
  - Chubby/ZooKeeper

Coordinator Failures

The coordinator is a RAMCloud app co-located with master M_0

- State is stored as objects in M_0
  - Except during early bootstrapping
  - Eat our own dog food
- On failures, the coordinator and M_0 die together
  - Collapses number of cases to worry about

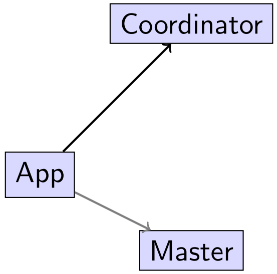
Recovery Mechanism

1. Chain of succession is pre-determined
2. C' notices C is down, shoots C in the head
3. C' starts normal master recovery for M_0' locally
   - Broadcast to backup hosts to find segments, so C' needs some idea of the host list
4. C' updates DNS entries, adds host to chain of succession

Detecting Host Failures Example

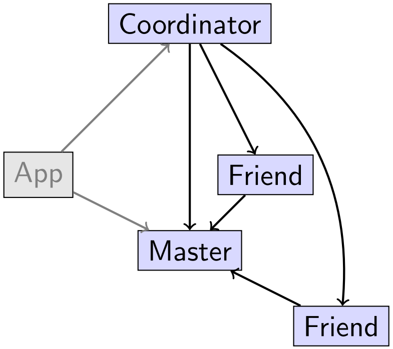
1. App's RPC to Master times out
2. App notifies Coordinator
3. Coordinator verifies report, asking others to check from different angles
4. Coordinator shoots Master in the head

Detecting Host Failures Example



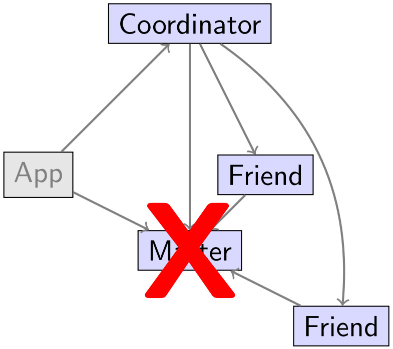
1. App's RPC to Master times out
2. App notifies Coordinator
3. Coordinator verifies report, asking others to check from different angles
4. Coordinator shoots Master in the head

Detecting Host Failures Example



1. App's RPC to Master times out
2. App notifies Coordinator
3. Coordinator verifies report, asking others to check from different angles
4. Coordinator shoots Master in the head

Detecting Host Failures Example



1. App's RPC to Master times out
2. App notifies Coordinator
3. Coordinator verifies report, asking others to check from different angles
4. Coordinator shoots Master in the head

How to Shoot a Machine in the Head

Once a master has been recovered, it is unsafe for the old host to service requests.

- Apps cache locations, may still access old host
- This would break our consistency model

Options

- Shoot down location caches on every host – too expensive
- Cut it off from backups
  - Apps can still see inconsistent reads
- Have it commit suicide if it doesn't receive watchdog pings
  - TTL must be less than the *minimum* recovery time
- Out-of-band controls
  - Cut off power or network port – is this possible?
  - Others?

Questions/Comments

Possible topics to revisit:

- Central Coordinator vs P2P
- How could machines find each other?
- How could they find the coordinator?
- Partitioning objects by address ranges vs hash ranges
- Metrics and algorithms for load balancing
- Out of band controls for killing a machine