# Cluster Management in RAMCloud

Diego Ongaro

Stanford University

RAMCloud Design Review
April 1, 2010

# Managing a RAMCloud Cluster

## RAMCloud's design requires large clusters

- 4,000 servers for 250TB of storage
- More hosts = faster recovery times, faster burst speeds

## Cluster Management 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

- 4,000 servers for 250TB of storage
- More hosts = faster recovery times, faster burst speeds

## Cluster Management 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

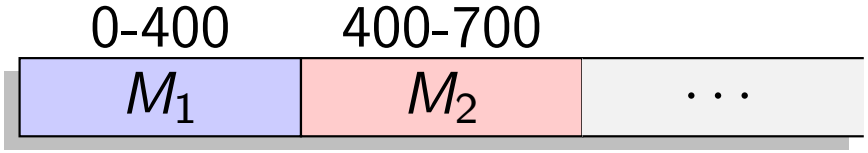
- A central coordinator is simple
- Global view useful for locations, load balancing, administration
- We think a single machine can handle it
  - Only thousands of hosts
  - We can service 1 million ops per second, *remember*?
- Coordinator is off the critical path
  - Apps and masters aggressively cache location info
- How do machines find the coordinator?
- What happens when it fails?

## Alternative: P2P

- Robust
- Duplicates substantial location information at every host
  - Won't respond as quickly to configuration changes
- May make sub-optimal load balancing decisions
- How do machines find each other?

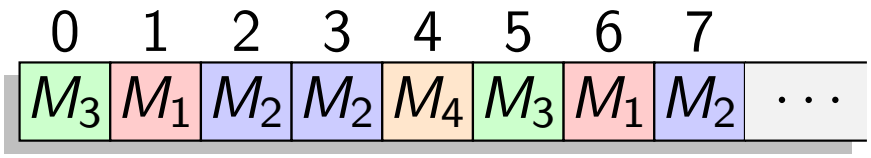
# Where to Place Objects

## By address ranges (tablets)




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

## Alternative: By hash ranges




- "Natural" load balancing
- Impractical to co-locate indexes with their data
- Table enumerate touches all masters
- Auto-increment object IDs inefficient
- Need full host list cached on apps

# Coordinator State

- Tablet Map: ~10K to 10M rows

| Workspace | Table | Objects | Master |
|---|---|---|---|
| 45 | 9 | 0 to $\infty$ | 10.0.3.52 |
| 72 | 3 | 0 to 30M | 10.0.9.33 |
| 72 | 3 | 30M to 50M | 10.0.3.52 |

  - Finding objects

# Coordinator State

- Tablet Map: ~10K to 10M rows

| Workspace | Table | Objects | Master |
|---|---|---|---|
| 45 | 9 | 0 to $\infty$ | 10.0.3.52 |
| 72 | 3 | 0 to 30M | 10.0.9.33 |
| 72 | 3 | 30M to 50M | 10.0.3.52 |

  - Finding objects

- Host List: ~10K rows

| Host | Status | Rack |
|---|---|---|
| 10.0.3.52 | Master + Backup | 3 |
| 10.0.2.17 | Powered Down | 2 |
| 10.0.9.23 | Master + Backup | 9 |

  - Finding available backup servers
  - Load balancing and recovery

# Coordinator State

- Tablet Map: ~10K to 10M rows

| Workspace | Table | Objects | Master |
|---|---|---|---|
| 45 | 9 | 0 to $\infty$ | 10.0.3.52 |
| 72 | 3 | 0 to 30M | 10.0.9.33 |
| 72 | 3 | 30M to 50M | 10.0.3.52 |

  - Finding objects

- Host List: ~10K rows

| Host | Status | Rack |
|---|---|---|
| 10.0.3.52 | Master + Backup | 3 |
| 10.0.2.17 | Powered Down | 2 |
| 10.0.9.23 | Master + Backup | 9 |

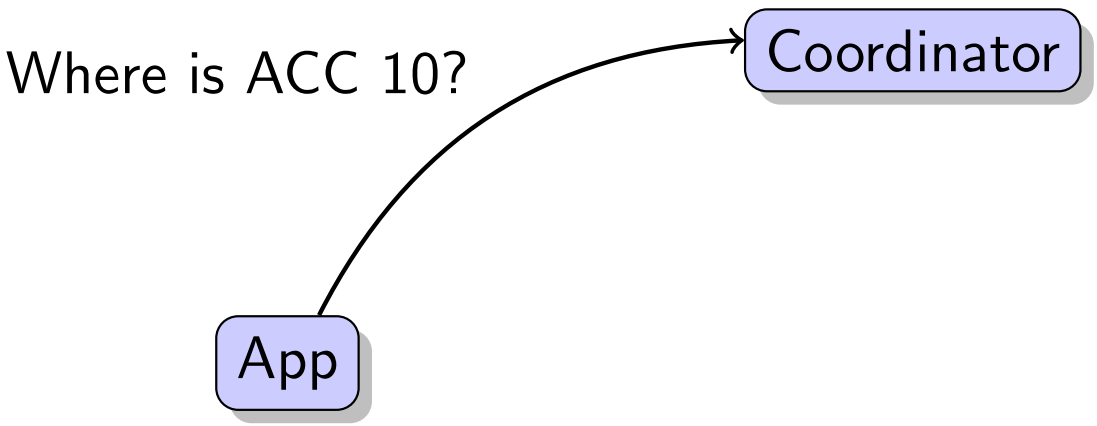
  - Finding available backup servers
  - Load balancing and recovery

- Authentication info
- Statistics for load balancing

# How Applications Find Objects

1. Query the coordinator's tablet map
2. Cache results
3. Invalidate cache when it leads to the wrong master

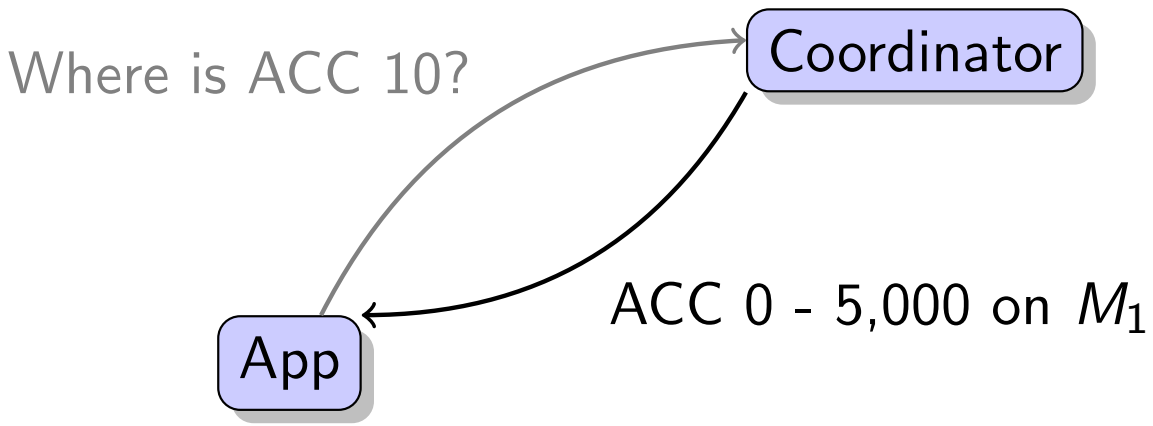
## Example: read(ACC, 10)



# How Applications Find Objects

1. Query the coordinator's tablet map
2. Cache results
3. Invalidate cache when it leads to the wrong master

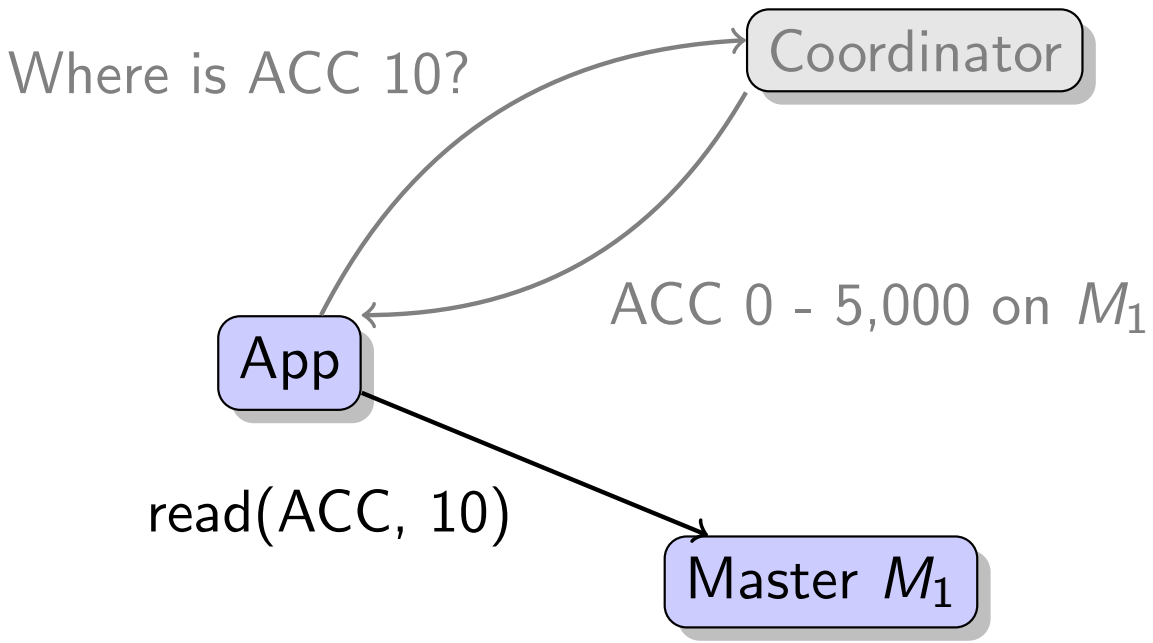
## Example: read(ACC, 10)



# How Applications Find Objects

1. Query the coordinator's tablet map
2. Cache results
3. Invalidate cache when it leads to the wrong master

## Example: read(ACC, 10)



# 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
  - ZooKeeper
  - A file in a shared filesystem

# Coordinator Failures

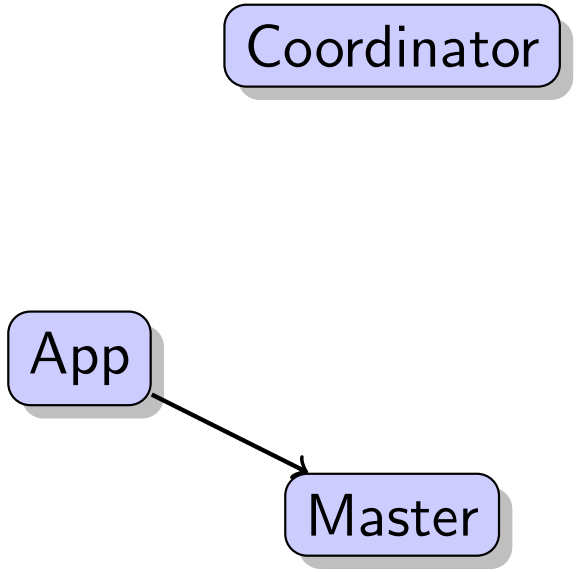
The coordinator is a RAMCloud app co-located with master $M_0$

- Coordinator 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

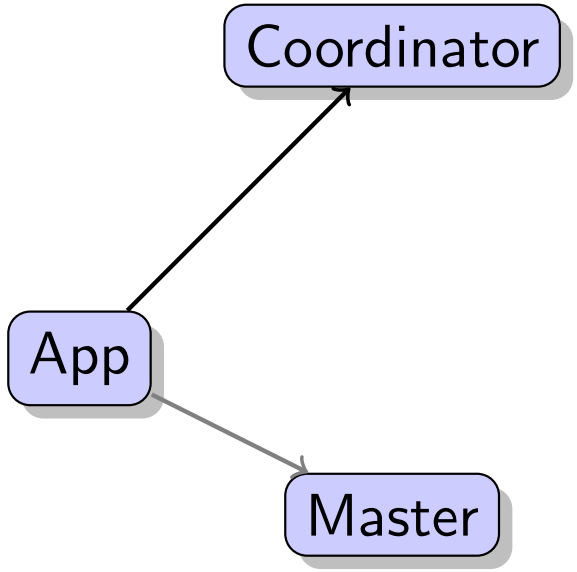
1. Chain of succession is pre-determined
2. $C'$ notices $C$ is down, disables $C$
3. $C'$ starts normal master recovery for $M_0'$ locally
   - Broadcast to backup hosts to find segments, so $C'$ needs some idea of the host list
4. $C'$ updates DNS entries, adds host to chain of succession

# Detecting Host Failures Example



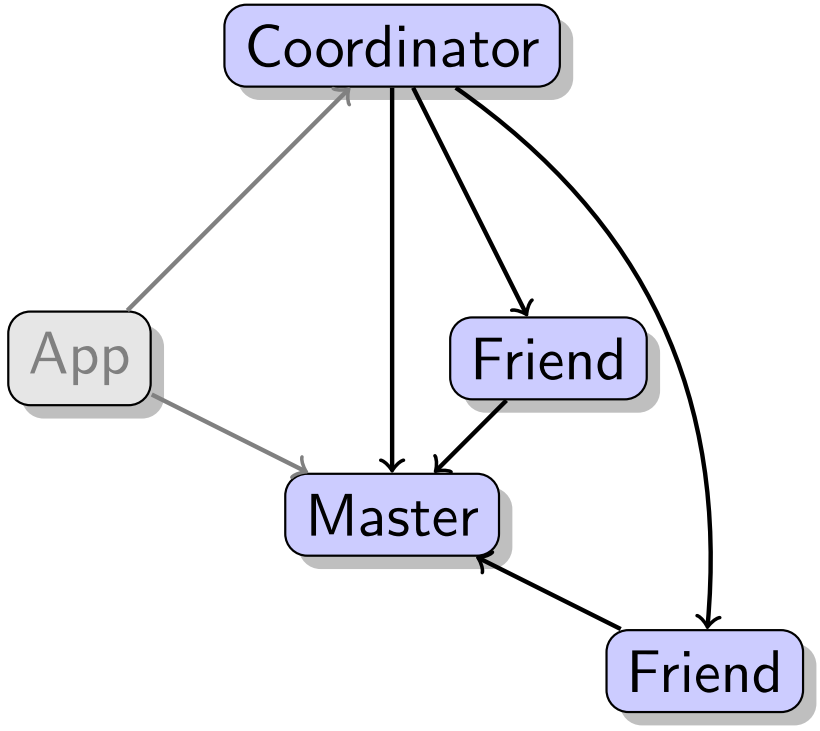

1. App's RPC to Master times out
2. App notifies Coordinator
3. Coordinator verifies report, asking others to check from different angles
4. Coordinator disables Master

# Detecting Host Failures Example



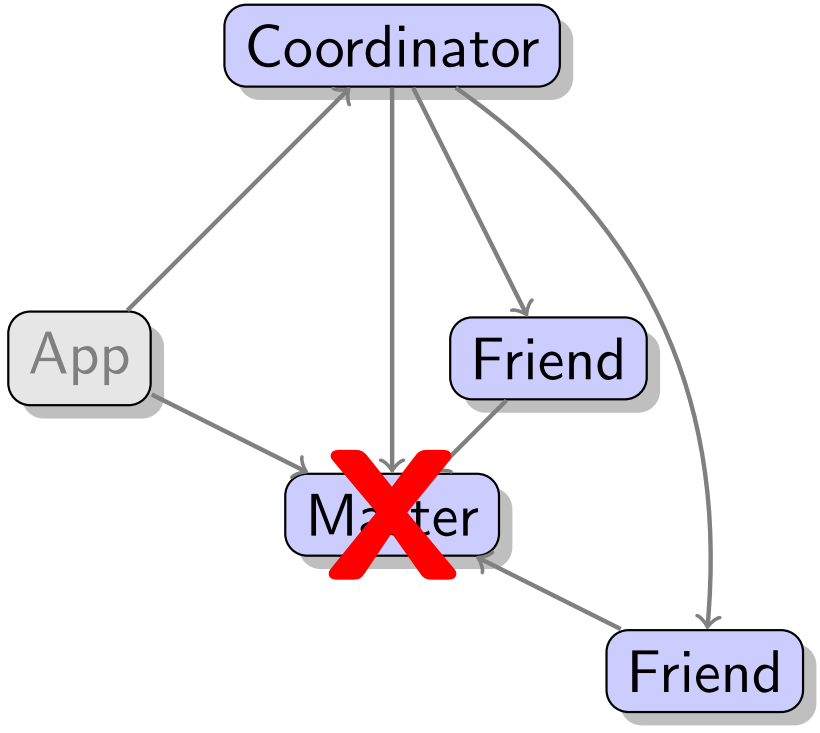

1. App's RPC to Master times out
2. App notifies Coordinator
3. Coordinator verifies report, asking others to check from different angles
4. Coordinator disables Master

# 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 disables Master

# 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 disables Master

# How to Disable a Machine

Once a master has been recovered, must prevent it from servicing requests

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

## Options

- Flush location caches on app servers – too expensive
- Cut service off from backups
  - Apps can still see inconsistent reads
- Have it disable itself 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?

# Quorum

Cluster must be unavailable until a quorum is met

- Major network partitions
- Rebooting after power outages

How do we define a quorum?

- All data is available
- All sufficiently replicated data (e.g., $r$=4) data is available
- Percentage of machines
- Number of racks

# Conclusion

Latency and scale make cluster management hard:

- Low latency:
  must react to failures quickly
- Large scale:
  cluster configuration continuously changing

But the same properties help, too:

- Low latency/high throughput:
  central coordinator manages the cluster
- Large scale:
  machines cooperate to detect failures

# Questions/Comments

Possible topics to revisit:

- Central Coordinator vs P2P
- In a P2P approach, how could machines find each other?
- Is DNS a good way for machines to find the coordinator?
- If we don't use something like ZooKeeper, will we regret it?
- What out-of-band controls are available to disable machines?
- Ideas for load balancing metrics