Fault Tolerant Cluster Coordination in RAMCloud: Lessons Learnt

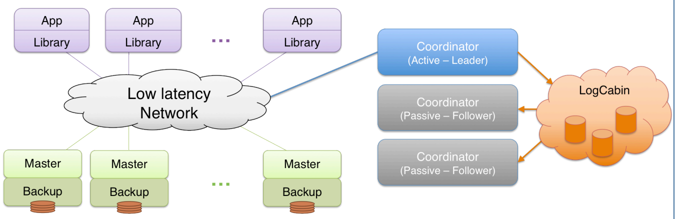
Ankita Arvind Kejriwal, John Ousterhout

Status

- Presented a first design – SEDCL 2012 Retreat
- Implementation, re-design, unit testing, high level testing cycle – June to Dec 2012
- Currently: Extensive crash / recovery testing (Arjun) and bug fixes, improvements in LogCabin (Diego)

Coordinator in RAMCloud

- Manages cluster membership and tablet configuration
- Stores core metadata



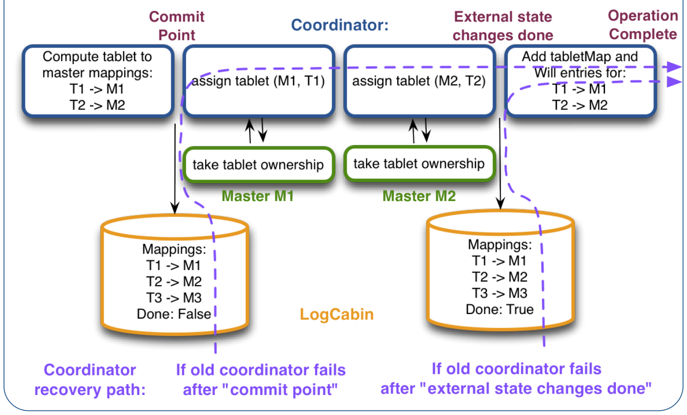
- Coordinator affects state of other nodes in cluster

Goal

Atomic distributed state change

(even in case of failures)

Simplified Coordinator Design


Commit Point
Coordinator:
External state changes done
Operation Complete
Compute tablet to master mappings: T1 -> M1 T2 -> M2
assign tablet (M1, T1)
assign tablet (M2, T2)
Add tabletMap and Will entries for: T1 -> M1 T2 -> M2
take tablet ownership
take tablet ownership
Master M1
Master M2
Mappings: T1 -> M1 T2 -> M2 T3 -> M3 Done: False
LogCabin
Mappings: T1 -> M1 T2 -> M2 T3 -> M3 Done: True
Coordinator recovery path:
If old coordinator fails after "commit point"
If old coordinator fails after "external state changes done"

Coordinator holds the ground truth

Reduces errors, simplifies some failure scenarios.

Eg.: Master crash during coordinator replay of create table. Simply modify coordinator's local state, master recovery will assign ownership to recovery master.

LogCabin –> TreeHouse

Original intuition: Ordering between operations would important for replay

➡ Log Structured Persistent Storage (LogCabin)

Lesson: Ordering does **not** matter between ops, only matters between classes of operations (server-related be replayed before table-related)

➡ Tree Structured Persistent Storage (TreeHouse)

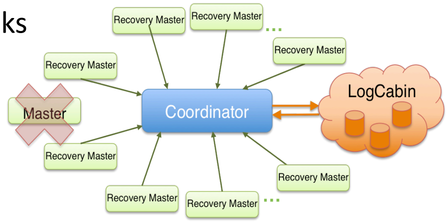
➡ More natural API, makes programming easier, replay classes of ops.

Operations are not **that** rare

Original intuition: Operations to coordinator rare, not on critical path.

➡ Handle sequentially, synchronous disk writes

Lesson: Completion of master recoveries bottlenecks



➡ Need batching + pipelining

Many more low-level lessons Examples:

- Atomicity + replays ➡ all operations need to be idempotent. Timing effects made this harder
- Some ops comprise of multiple independent components ➡ need atomic-multi-append
- Master crashes during updates would deadlock coordinator ➡ solution: Asynchronous updates (Stephen's poster)
- New async updates ➡ log entries can be cleaned up only on acknowledgement
- Many subtle issues resulting in bugs, many of them timing dependent (Arjun's poster)

SEDCL Annual Retreat 2013