Chubby Lock Service

Amir H. Payberah
amir@sics.se

Amirkabir University of Technology
(Tehran Polytechnic)

What is the Problem?

What is the Problem?

- Large distributed system
- How to ensure that only one process across a fleet of processes acts on a resource?

What is the Problem?

- Large distributed system
- How to ensure that only one process across a fleet of processes acts on a resource?
  - Ensure only one server can write to a database.
  - Ensure that only one server can perform a particular action.
  - Ensure that there is a single master that processes all writes.
  - ...

What is the Problem?

- All these scenarios need a simple way to coordinate execution of process to ensure that only one entity is performing an action.

What is the Problem?

- All these scenarios need a simple way to coordinate execution of process to ensure that only one entity is performing an action.
- We need agreement (consensus)

What is the Problem?

- All these scenarios need a simple way to coordinate execution of process to ensure that only one entity is performing an action.
- We need agreement (consensus)
- Paxos ...

How Can We Use Paxos to Solve the Problem of Coordination?

Possible Solutions

- Building a consensus library
- Building a centralized lock service

Consensus Library vs. Centralized Service

- Consensus library

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.
  - Need to wait on majority (quorums).

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.
  - Need to wait on majority (quorums).
- Centralized lock service

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.
  - Need to wait on majority (quorums).
- Centralized lock service
  - Clean interface.

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.
  - Need to wait on majority (quorums).
- Centralized lock service
  - Clean interface.
  - Service can be modified independently.

Consensus Library vs. Centralized Service

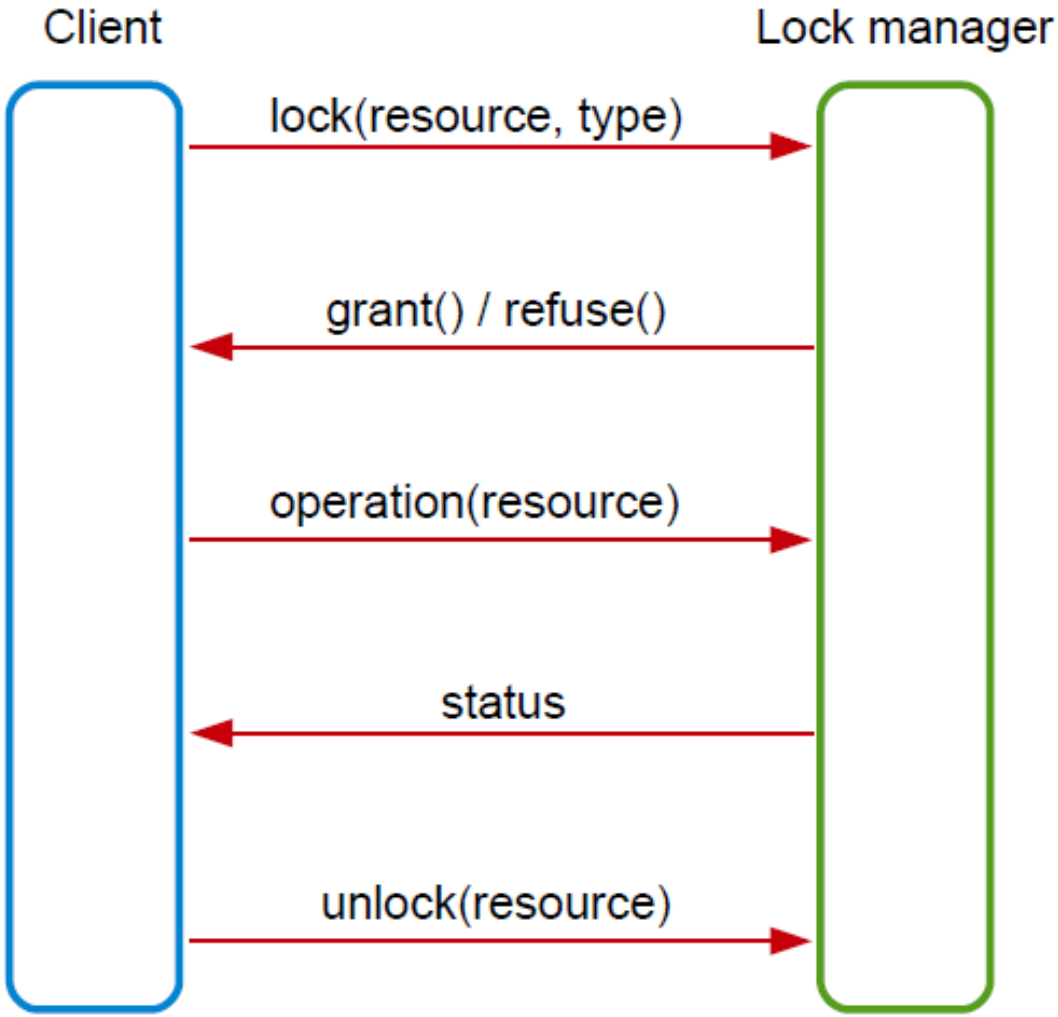
- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.
  - Need to wait on majority (quorums).
- Centralized lock service
  - Clean interface.
  - Service can be modified independently.
  - A single client can obtain a lock and make progress (non-quorum based decisions, the lock service takes care of it)

Consensus Library vs. Centralized Service

- Consensus library
  - Library code has to be included in every program.
  - Application developers to be experts in distributed system.
  - Need to wait on majority (quorums).

- Centralized lock service
  - Clean interface.
  - Service can be modified independently.
  - A single client can obtain a lock and make progress (non-quorum based decisions, the lock service takes care of it)
  - Application developers do not have to worry about dealing with operations, various failure modes debugging etc.

Centralized Lock Service



Chubby is a Lock Service

Chubby

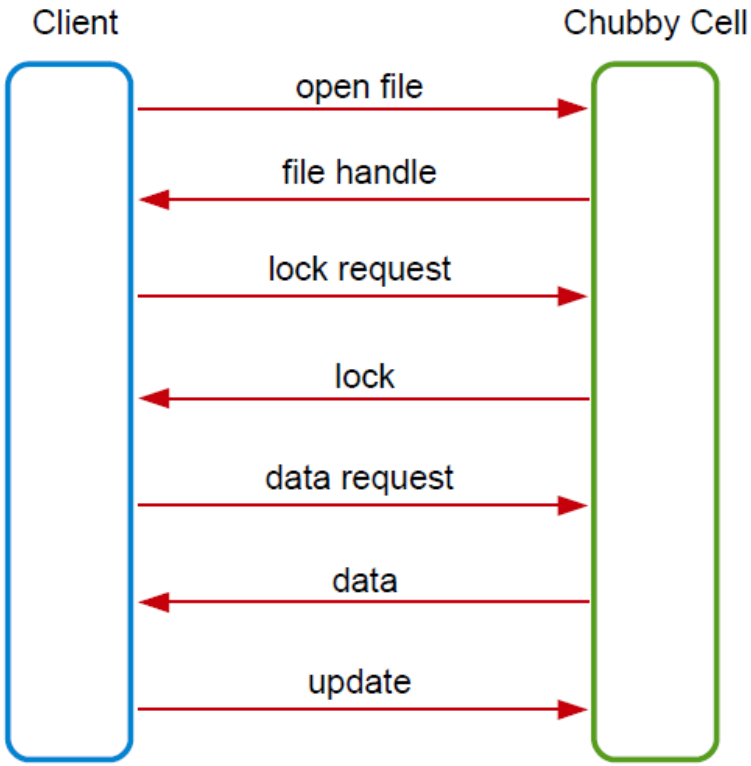
- Primary goals: reliability and availability (not high performance)

Chubby

- Primary goals: reliability and availability (not high performance)
- Used in Google: GFS, Bigtable, etc.

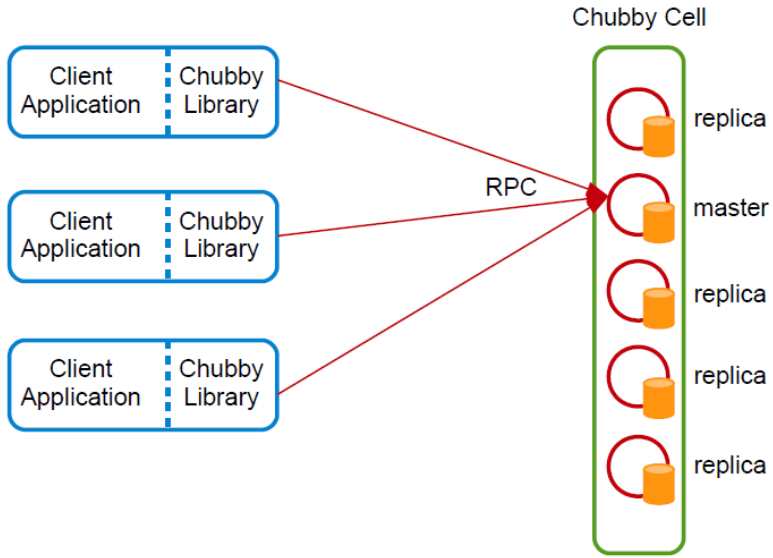
Chubby Structure

- Two main components:
  - Server (chubby cell)
  - Client library
- Communicate via RPC

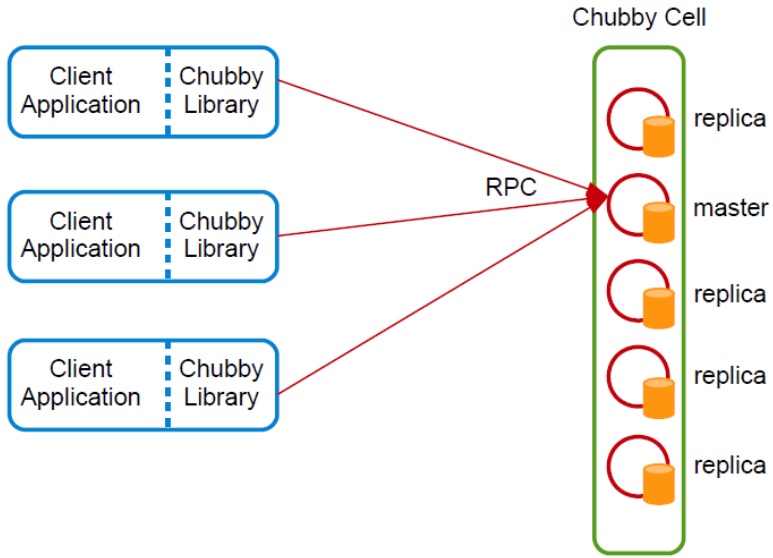


Chubby Cell

- A small set of servers (replicas)

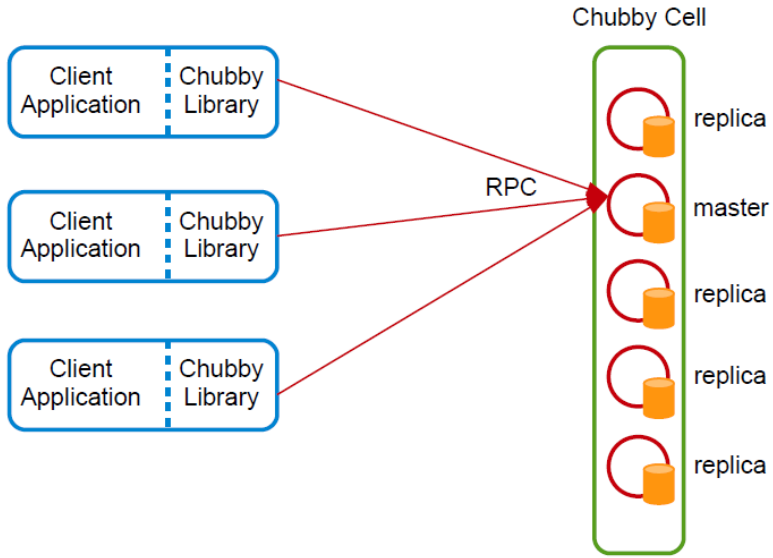


Chubby Cell



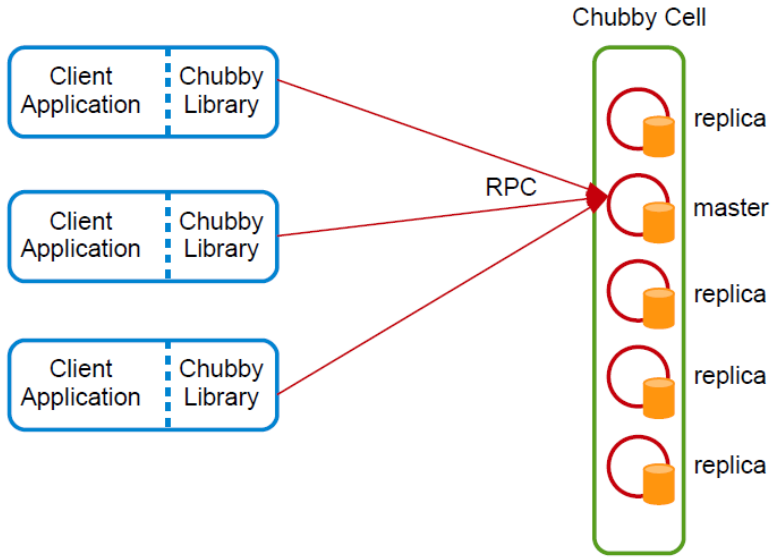
- A small set of servers (replicas)
- One is the master (elected from the replicas via Paxos)

Chubby Cell



- A small set of servers (replicas)
- One is the master (elected from the replicas via Paxos)
- Maintain copies of simple database (replicated state machines)

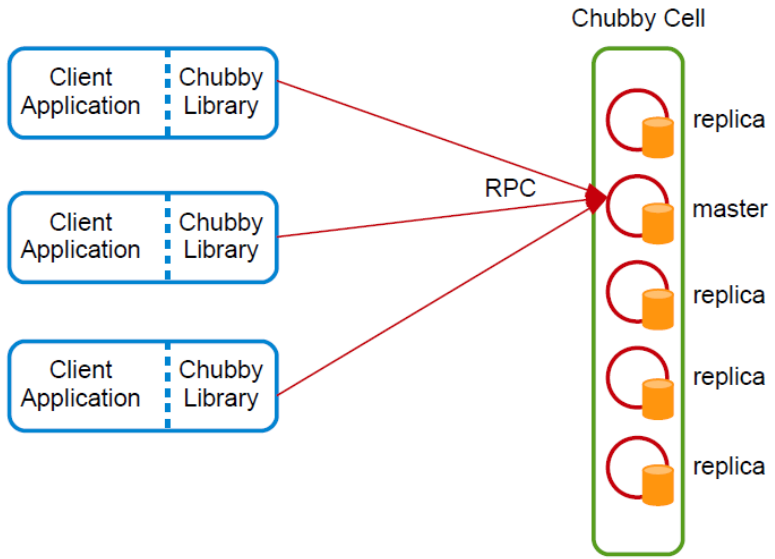
Chubby Cell



- A small set of servers (replicas)
- One is the master (elected from the replicas via Paxos)
- Maintain copies of simple database (replicated state machines)
  - Only the master initiates reads and writes of this database.
  - All other replicas simply copy updates from the master (using the Paxos protocol).

Chubby Client

- Find the master: sending master location requests to replicas.
- All requests are sent directly to the master.



Chubby Operations

- Write requests

Chubby Operations

- Write requests
  - The master receives the request.

Chubby Operations

- Write requests
  - The master receives the request.
  - Write requests are propagated via the consensus protocol to all replicas.

Chubby Operations

- Write requests
  - The master receives the request.
  - Write requests are propagated via the consensus protocol to all replicas.
  - Acknowledges when write request reaches the majority of the replicas

Chubby Operations

- Write requests
  - The master receives the request.
  - Write requests are propagated via the consensus protocol to all replicas.
  - Acknowledges when write request reaches the majority of the replicas
- Read requests

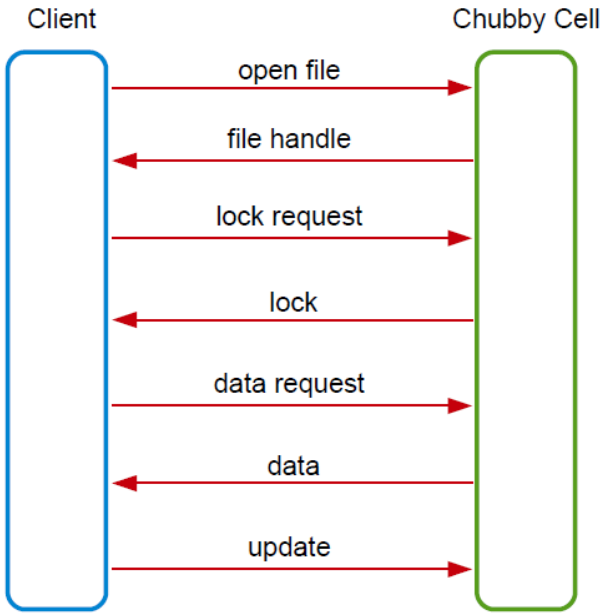
Chubby Operations

- Write requests
  - The master receives the request.
  - Write requests are propagated via the consensus protocol to all replicas.
  - Acknowledges when write request reaches the majority of the replicas
- Read requests
  - Satisfied by the master alone.

Let's Go into More Details

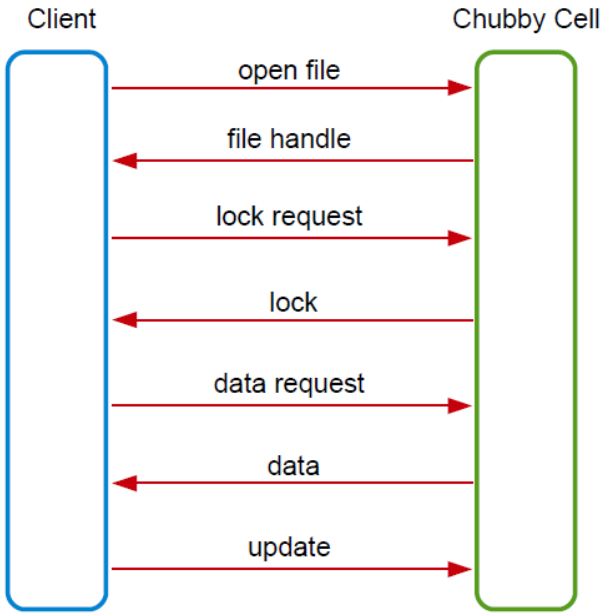
Chubby Interface (1/2)

- Chubby exports a unix-like filesystem interface.



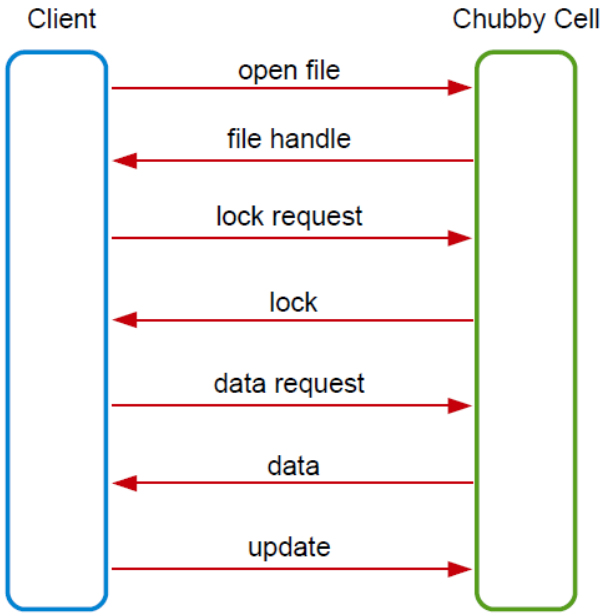
Chubby Interface (1/2)

- Chubby exports a unix-like filesystem interface.
- Tree of files with names separated by slashes (namesapace): `/ls/cell1/aut/cc14`:



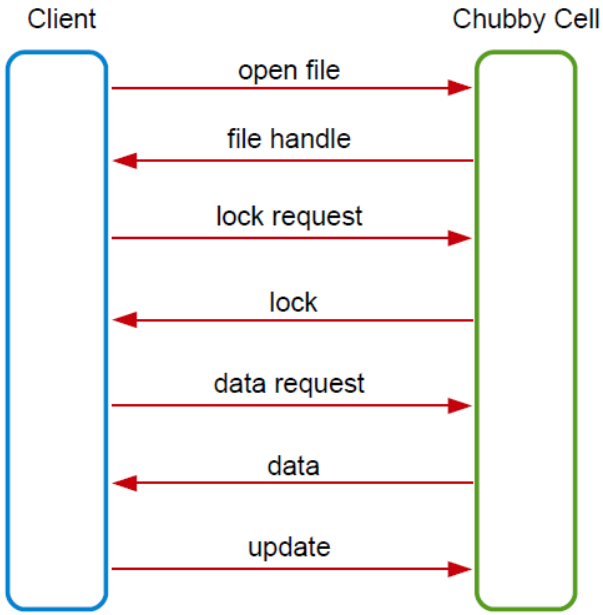
Chubby Interface (1/2)

- Chubby exports a unix-like filesystem interface.
- Tree of files with names separated by slashes (namesapace): `/ls/cell1/aut/cc14`:
  - 1st component (`ls`): lock service (common to all names)



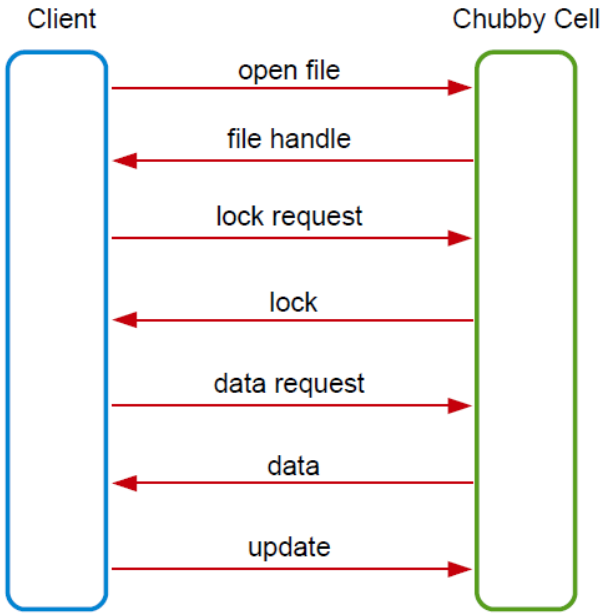
Chubby Interface (1/2)

- Chubby exports a unix-like filesystem interface.
- Tree of files with names separated by slashes (namesapace): `/ls/cell1/aut/cc14`:
  - 1st component (`ls`): lock service (common to all names)
  - 2nd component (`cell1`): the chubby cell name



Chubby Interface (1/2)

- Chubby exports a unix-like filesystem interface.
- Tree of files with names separated by slashes (namesapace): `/ls/cell1/aut/cc14`:
  - 1st component (`ls`): lock service (common to all names)
  - 2nd component (`cell1`): the chubby cell name
  - The rest: the name of the directory and the file (name inside the cell)



Chubby Interface (2/2)

- Chubby maintains a namespace that contains files and directories, called nodes.

Chubby Interface (2/2)

- Chubby maintains a namespace that contains files and directories, called nodes.
- Clients can create nodes and add contents to nodes.

Chubby Interface (2/2)

- Chubby maintains a namespace that contains files and directories, called nodes.
- Clients can create nodes and add contents to nodes.
- Support most normal operations:
  - create, delete, open, write, ...

Chubby Interface (2/2)

- Chubby maintains a namespace that contains files and directories, called nodes.
- Clients can create nodes and add contents to nodes.
- Support most normal operations:
  - create, delete, open, write, ...
- Clients open nodes to obtain handles that are analogous to unix file descriptors.

Locks (1/2)

- Each Chubby file and directory (node) can act as a reader/writer lock.

Locks (1/2)

- Each Chubby file and directory (node) can act as a reader/writer lock.
- A client handle may hold the lock in two modes: writer (exclusive) mode or reader (shared) mode.

Locks (1/2)

- Each Chubby file and directory (node) can act as a reader/writer lock.
- A client handle may hold the lock in two modes: writer (exclusive) mode or reader (shared) mode.
  - Only one client can hold lock in writer mode.
  - Many clients can hold lock in reader mode.

Locks (2/2)

- Locks are advisory in Chubby.
  - Holding a lock is not necessary to access file.
  - Holding a lock does not prevent other clients accessing file.
  - Conflicts only with other attempts to acquire the same lock.

Chubby Example (1/2)

- Electing a primary (leader) process.
- Steps in primary election:

Chubby Example (1/2)

- Electing a primary (leader) process.
- Steps in primary election:
  - Clients open a lock file and attempt to acquire the lock in write mode.

Chubby Example (1/2)

- Electing a primary (leader) process.
- Steps in primary election:
  - Clients open a lock file and attempt to acquire the lock in write mode.
  - One client succeeds (becomes the primary) and writes its name/identity into the file.

Chubby Example (1/2)

- Electing a primary (leader) process.
- Steps in primary election:
  - Clients open a lock file and attempt to acquire the lock in write mode.
  - One client succeeds (becomes the primary) and writes its name/identity into the file.
  - Other clients fail (become replicas) and discover the name of the primary by reading the file.

Chubby Example (1/2)

- Electing a primary (leader) process.
- Steps in primary election:
  - Clients open a lock file and attempt to acquire the lock in write mode.
  - One client succeeds (becomes the primary) and writes its name/identity into the file.
  - Other clients fail (become replicas) and discover the name of the primary by reading the file.
  - Primary obtains sequencer and passes it to servers (servers can insure that the elected primary is still valid).

Chubby Example (2/2)

- Leader election

```
Open("/ls/foo/OurServicePrimary", "write mode")
if (successful) { // primary
  SetContents(primary_identity)
} else { // replica
  Open("/ls/foo/OurServicePrimary", "read mode",
       "file-modification event")
  when notified of file modification:
    primary = GetContentsAndStat()
}
```

APIs

- `Open()`, `Close()`, `Delete()`
- `GetContentsAndStat()`, `GetStat()`, `ReadDir()`
- `SetContents()`
- `SetACL()`
- Locks: `Acquire()`, `TryAcquire()`, `Release()`
- Sequencers: `GetSequencer()`, `SetSequencer()`, `CheckSequencer()`

Events

- Chubby clients may subscribe to many events when they create a handle.

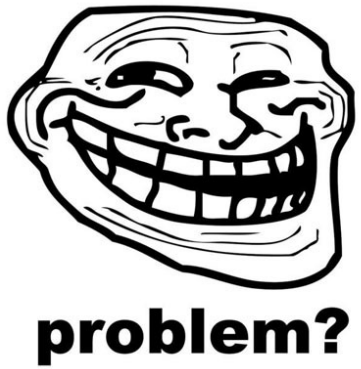
Events

- Chubby clients may subscribe to many events when they create a handle.

- Events include:
  - File contents modified
  - Child node added, removed, or modified
  - Chubby master failed over
  - Handle has become invalid
  - Lock acquired
  - Conflicting lock requested from another client

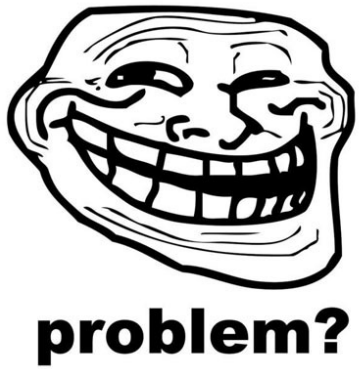
Locking - Problem

- After acquiring a lock and issuing a request (R1) a client may fail.



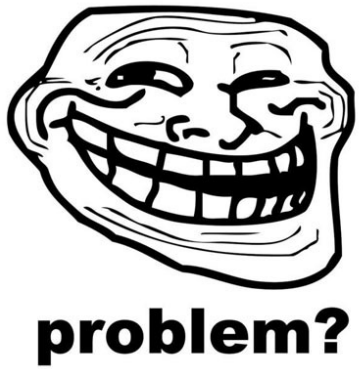
Locking - Problem

- After acquiring a lock and issuing a request (R1) a client may fail.
- Another client may acquire the lock and issue its own request (R2).



Locking - Problem

- After acquiring a lock and issuing a request (R1) a client may fail.
- Another client may acquire the lock and issue its own request (R2).
- R1 arrive later at the server and be acted upon (possible inconsistency).



Locking - Solutions (1/3)

- Solution 1: virtual time

Locking - Solutions (1/3)

- Solution 1: virtual time
  - But, it is costly to introduce sequence numbers into all the interactions.

Locking - Solutions (2/3)

- Solution 2: sequencers

Locking - Solutions (2/3)

- Solution 2: sequencers
  - Sequence numbers are introduced into only those interactions that make use of locks.

Locking - Solutions (2/3)

- Solution 2: sequencers
  - Sequence numbers are introduced into only those interactions that make use of locks.
  - A lock holder can obtain a sequencer from Chubby.

Locking - Solutions (2/3)

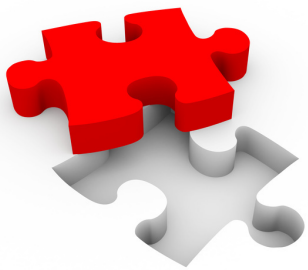
- Solution 2: sequencers
  - Sequence numbers are introduced into only those interactions that make use of locks.
  - A lock holder can obtain a sequencer from Chubby.
  - It attaches the sequencer to any requests that it sends to other servers (e.g., Bigtable)

Locking - Solutions (2/3)

- Solution 2: sequencers
  - Sequence numbers are introduced into only those interactions that make use of locks.
  - A lock holder can obtain a sequencer from Chubby.
  - It attaches the sequencer to any requests that it sends to other servers (e.g., Bigtable)
  - The other servers can verify the sequencer information

Locking - Solutions (3/3)

- Solution 3: lock-delay

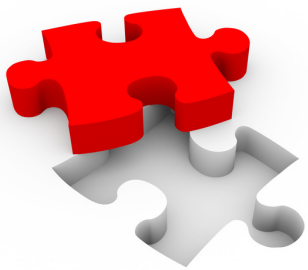


Locking - Solutions (3/3)

- Solution 3: lock-delay
  - Correctly released locks are immediately available.

Locking - Solutions (3/3)

- Solution 3: lock-delay
  - Correctly released locks are immediately available.
  - If a lock becomes free because holder failed or becomes inaccessible, lock cannot be claimed by another client until lock-delay expires.



Caching

- To reduce read traffic, Chubby clients cache file data and node meta-data.

Caching

- To reduce read traffic, Chubby clients cache file data and node meta-data.
- Write-through: write is done synchronously both to the cache and to the primary copy.

Caching

- To reduce read traffic, Chubby clients cache file data and node meta-data.
- Write-through: write is done synchronously both to the cache and to the primary copy.
- Master will invalidate cached copies upon a write request.

Caching

- To reduce read traffic, Chubby clients cache file data and node meta-data.
- Write-through: write is done synchronously both to the cache and to the primary copy.
- Master will invalidate cached copies upon a write request.
- The client also can allow its cache lease to expire.

Session Lease (1/3)

- Lease: a promise by one party to abide by an agreement for a given interval of time unless the promise is explicitly revoked.

Session Lease (1/3)

- Lease: a promise by one party to abide by an agreement for a given interval of time unless the promise is explicitly revoked.
- Session: a relationship between a Chubby cell and a Chubby client.

Session Lease (1/3)

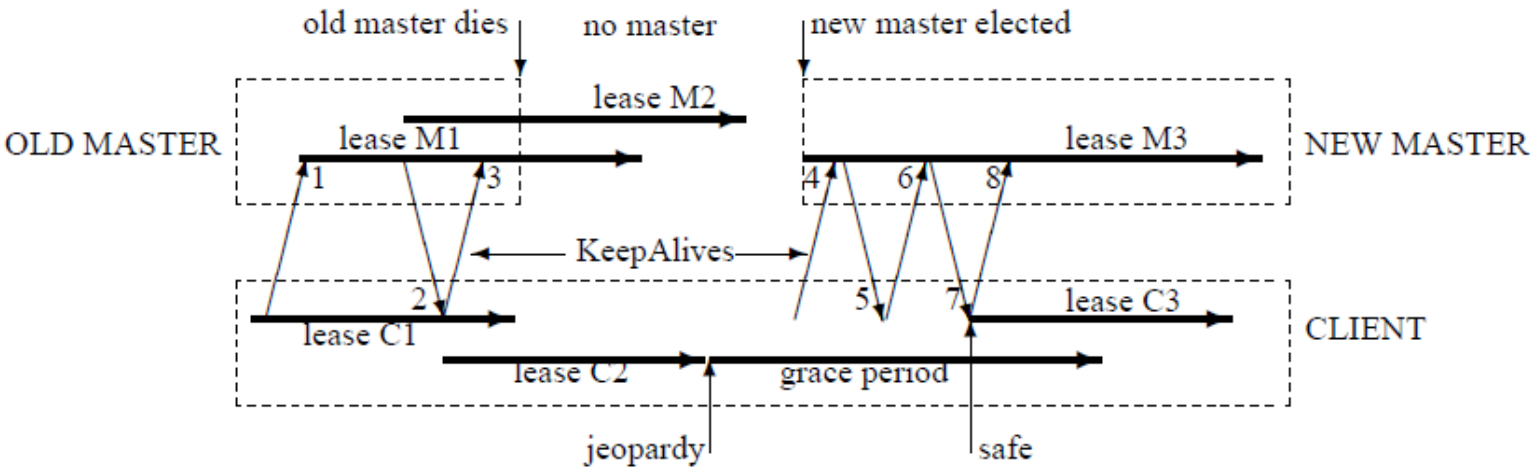
- Lease: a promise by one party to abide by an agreement for a given interval of time unless the promise is explicitly revoked.
- Session: a relationship between a Chubby cell and a Chubby client.
- Session lease: interval during which client's cached items (handles, locks, files) are valid.

Session Lease (1/3)

- Lease: a promise by one party to abide by an agreement for a given interval of time unless the promise is explicitly revoked.
- Session: a relationship between a Chubby cell and a Chubby client.
- Session lease: interval during which client's cached items (handles, locks, files) are valid.
- Session maintained through KeepAlives messages.

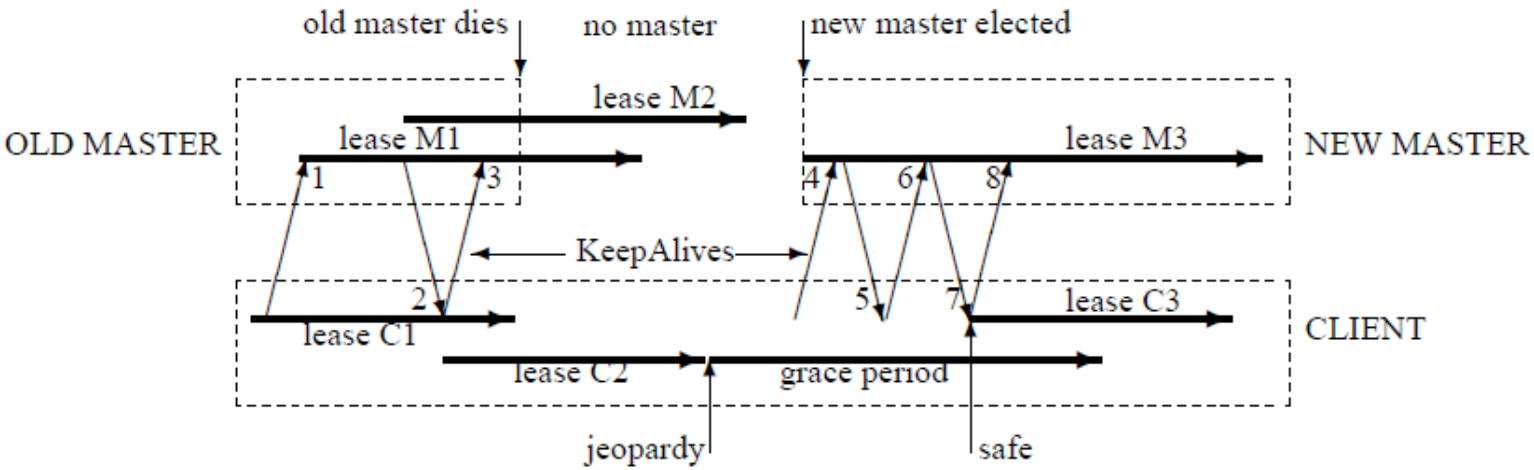
Session Lease (2/3)

- If client's local lease expires (happens when a master fails over)



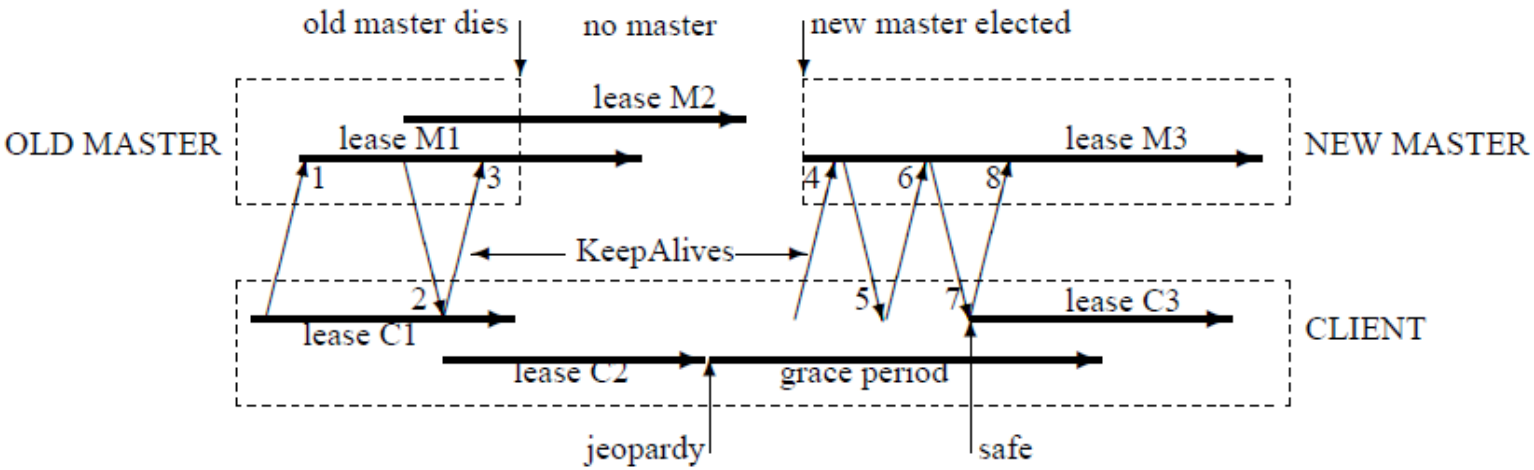
Session Lease (2/3)

- If client's local lease expires (happens when a master fails over)
  - Client disables cache.



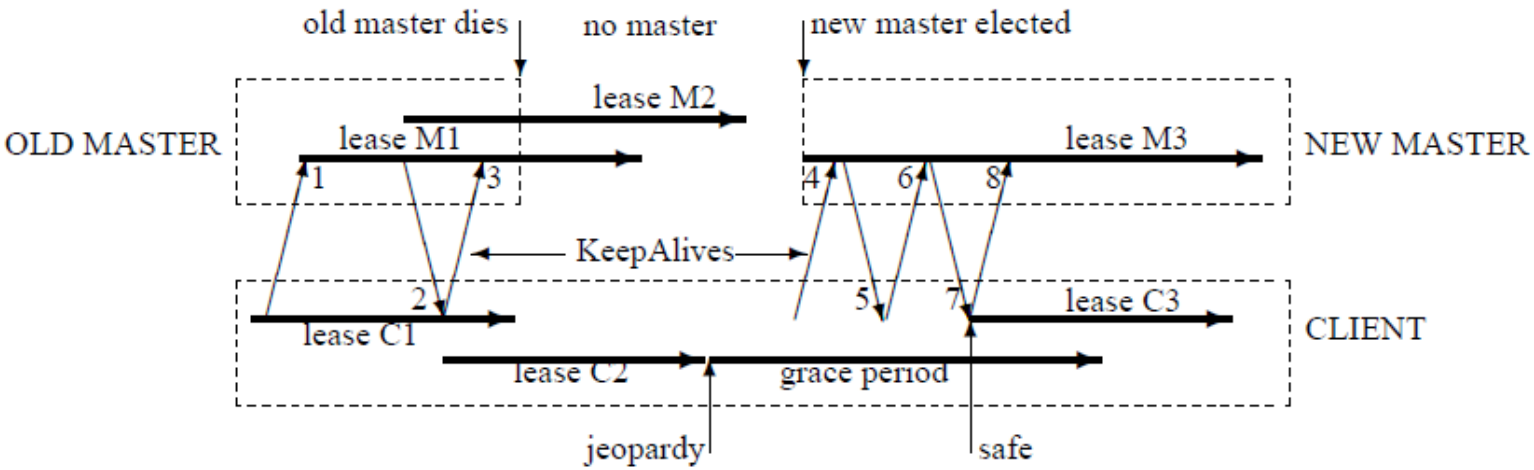
Session Lease (2/3)

- If client's local lease expires (happens when a master fails over)
  - Client disables cache.
  - Session in jeopardy, client waits in grace period (45s).



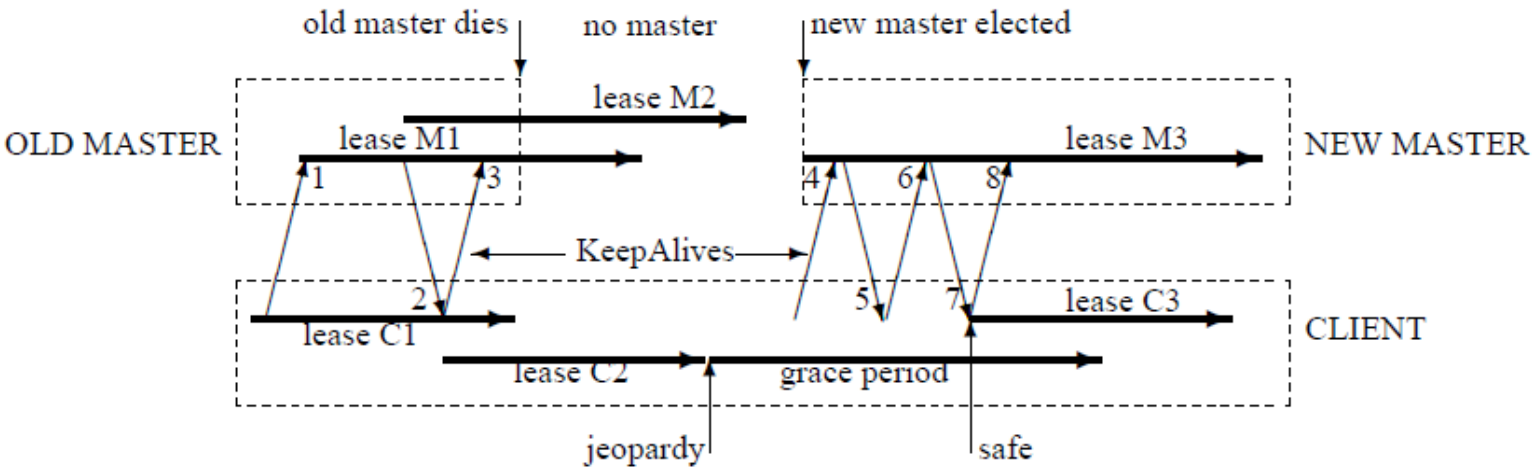
Session Lease (2/3)

- If client's local lease expires (happens when a master fails over)
  - Client disables cache.
  - Session in jeopardy, client waits in grace period (45s).
  - Sends jeopardy event to application.



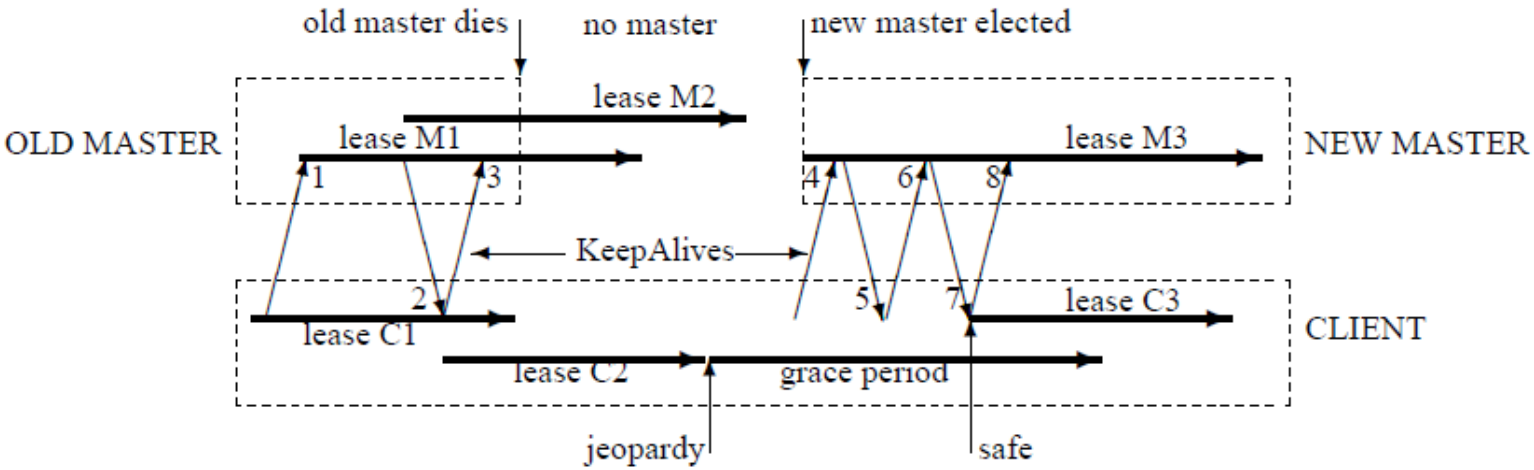
Session Lease (2/3)

- If client's local lease expires (happens when a master fails over)
  - Client disables cache.
  - Session in jeopardy, client waits in grace period (45s).
  - Sends jeopardy event to application.
  - Application can suspend activity.



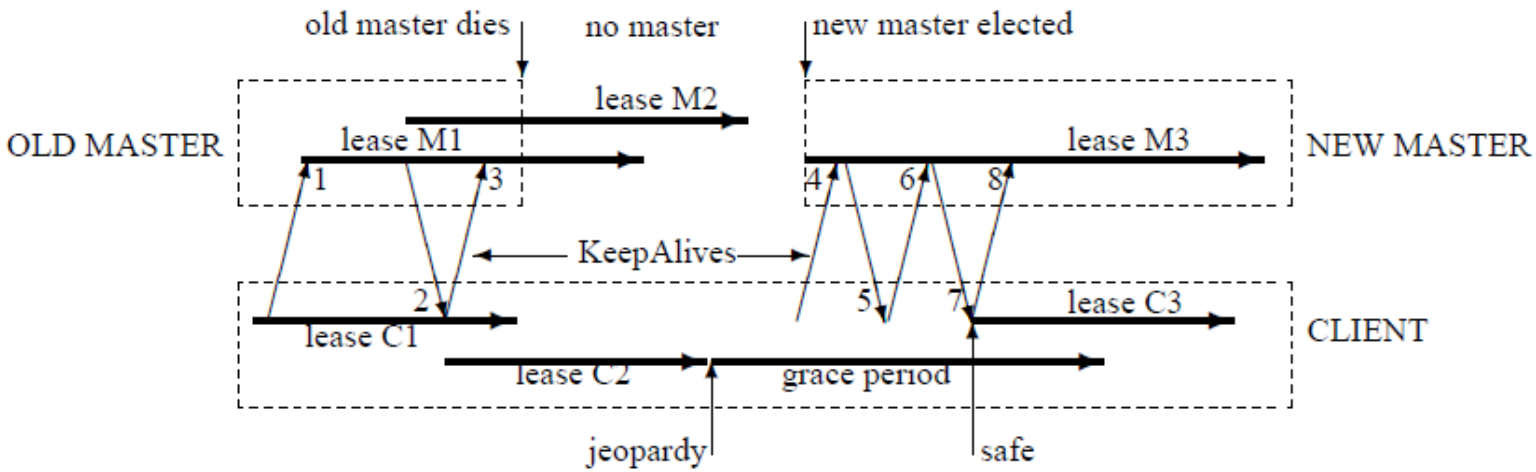
Session Lease (3/3)

- Client attempts to exchange KeepAlive messages with master during grace period.



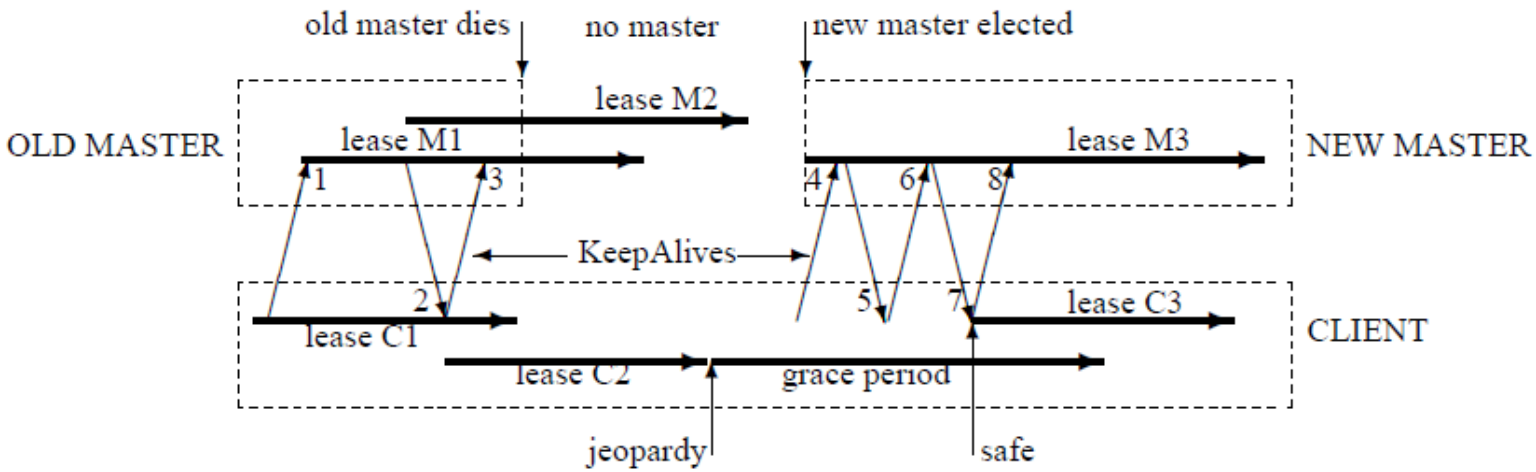
Session Lease (3/3)

- Client attempts to exchange KeepAlive messages with master during grace period.
  - Succeed: re-enables cache; send safe event to application



Session Lease (3/3)

- Client attempts to exchange KeepAlive messages with master during grace period.
  - Succeed: re-enables cache; send safe event to application
  - Failed: discards cache; sends expired event to application



Scaling

- Reducing communication with the master.
- Two techniques:
  - Proxies
  - Partitioning

Scaling - Proxies

- Proxy is a trusted process that pass requests from other clients to a Chubby cell.

Scaling - Proxies

- Proxy is a trusted process that pass requests from other clients to a Chubby cell.
- Can handle KeepAlives and reads $\rightarrow$ reduce the master load.

Scaling - Proxies

- Proxy is a trusted process that pass requests from other clients to a Chubby cell.
- Can handle KeepAlives and reads $\rightarrow$ reduce the master load.
- Cannot reduce write loads, but they are $<< 1\%$ of workload.

Scaling - Proxies

- Proxy is a trusted process that pass requests from other clients to a Chubby cell.
- Can handle KeepAlives and reads $\rightarrow$ reduce the master load.
- Cannot reduce write loads, but they are $<< 1\%$ of workload.
- Introduces another point of failure.

Scaling - Partitioning

- Namespace partitioned between servers.

Scaling - Partitioning

- Namespace partitioned between servers.
- N partitions, each with master and replicas
- Node D/C stored on P(D/C) = hash(D) mod N

Summary

Summary

- Library vs. Service
- Chubby: coarse-grained lock service
- Chubby cell and clients
- Unix-like interface

References:

- M. Burrows, The Chubby lock service for loosely-coupled distributed systems, OSDI, 2006.

Questions?