

Distributed Systems 1

CUCS Course 4113

<https://systems.cs.columbia.edu/ds1-class/>

Instructor: Roxana Geambasu

Chubby

Fault-tolerant Lock Service

Slides based on Mike Burrows' [OSDI'06 paper](#) and
Shimin Chen's [presentation](#) of it

Introduction

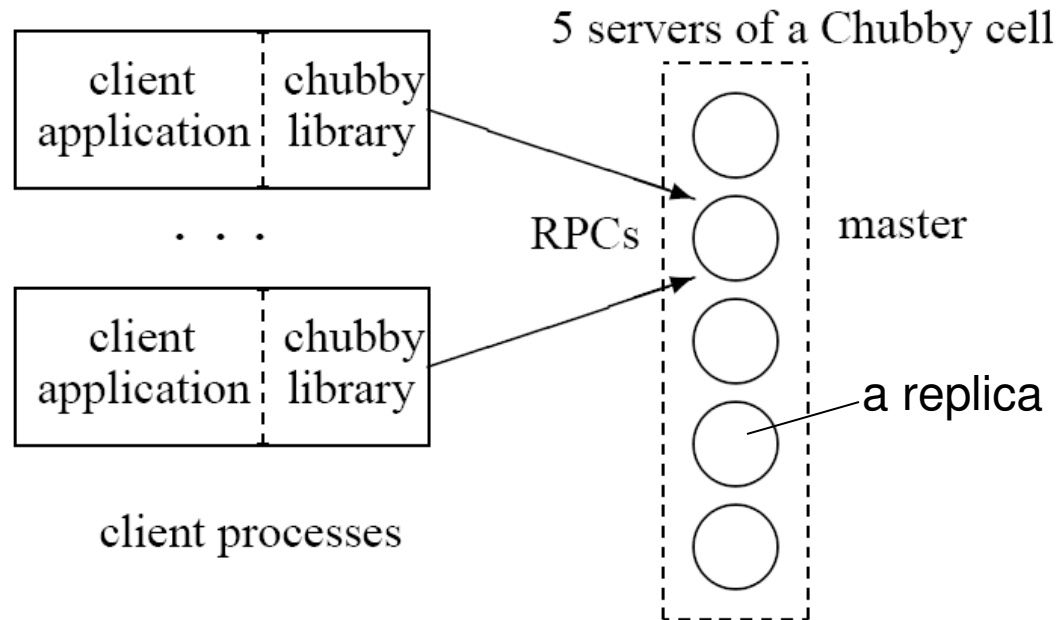
- What is Chubby?
 - Lock service in a loosely-coupled distributed system
 - Client interface similar to:
 - Whole-file advisory locks (think .lck files)
 - Notification of various events (e.g., file modifications, think inotify)
 - Primary goals: **reliability**, **availability**, **easy-to-understand semantics**
- How is it used?
 - Used in Google: **GFS**, **Bigtable**, etc.
 - Purposes: **elect masters**; **store small amount of metadata**, such as the root of the distributed data structures
 - Open-source version of Bigtable, Hbase, uses an open-source lock service, called Zookeeper

What Was the Chubby Paper About?

*“Building Chubby was an **engineering effort** ... it was not research. We claim no new algorithms or techniques. The purpose of this paper is to describe what we did and why, rather than to advocate it.”*

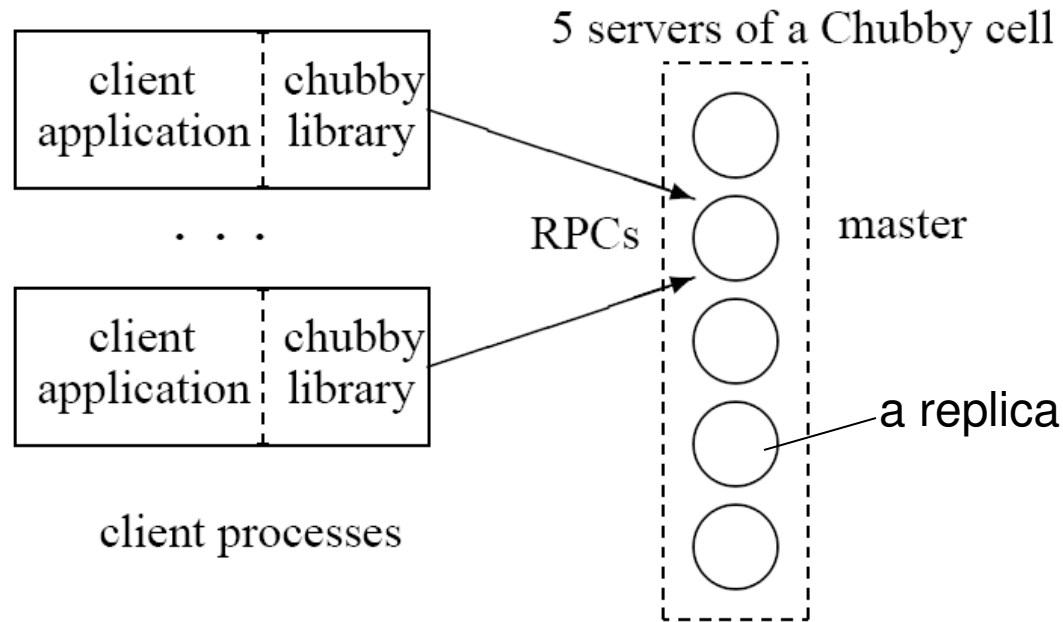
Chubby paper, OSDI 2006

System Architecture



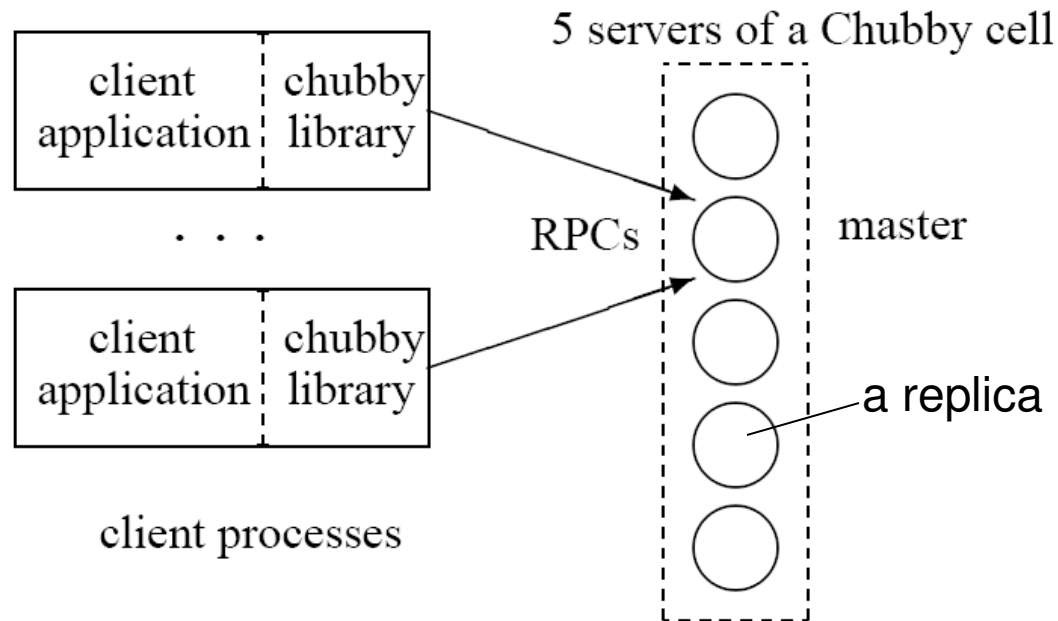
- A chubby cell consists of a **small set of servers** (replicas)
 - Placed in different racks, so as to minimize chance of correlated failures
- A **master** is elected from the replicas via **Paxos**
 - Master lease: several seconds
 - If master fails, a new one will be elected, but only **after master leases expire**
- Client talks to the master via the chubby library
 - All replicas are listed in DNS; clients discover master by talking to any replica

System Architecture (2)



- Replicas maintain copies of a simple database
- Clients send read/write requests only to the master
- For a write:
 - The master propagates it to replicas via Paxos
 - **Replies after the write reaches a majority of replicas**
- For a read:
 - The master satisfies the read alone

System Architecture (3)



- If a replica fails and does not recover for a long time (a few hours)
 - A fresh machine is selected to be a new replica, replacing the failed one
 - It updates the DNS
 - Obtains a recent copy of the database
 - The current master polls DNS periodically to discover new replicas
 - Integrating the new replica into the group is another Paxos run

Paxos Use in Master Election

- At any point in time, there must be **at most one master**
 - No two nodes must think they are masters at same time
- Example:
 - Suppose A is master and it gets disconnected from B
 - B times out trying to talk to A, thinks A is dead, and proposes that it be the master
 - If other nodes agree and A doesn't hear about the new master, then A will continue to act as master for a while, accepting read requests for what could be stale data, for example
 - How would you solve this?

Paxos Use in Master Election

- Chubby combines Paxos with a lease mechanism
 - When a master dies, a node proposes a master change through Paxos
 - When nodes receive the proposal, they will only accept it if the old master's lease has expired
 - A node becomes the master if a majority of nodes have given it the accept to become the master
 - Once a node becomes a master, it knows that it will remain so for at least the lease period
 - It can extend the lease by getting the accept from a majority of the nodes

Chubby Interface: UNIX File System

- Chubby supports a strict tree of files and directories
 - The way to think about these files is that they are locks with a little bit of contents (e.g., identity and location of a primary)
 - No symbolic links, no hard links
 - /ls/foo/wombat/pouch
 - 1st component (ls): lock service (common to all names)
 - 2nd component (foo): the chubby cell (used in DNS lookup to find the cell master)
 - The rest: name inside the cell
- Support most normal operations
 - Create, delete, open, write, ...
- Support reader/writer lock on a node

Chubby Events

- Clients can subscribe to **events**
 - File contents modified: e.g., if the file contains the location of a service, this event can be used to track the service's location
 - Master failed over
 - Child node added, removed, modified
 - Handle becomes invalid: probably communication problem
 - Lock acquired (rarely used)
 - Locks are conflicting (rarely used)

APIs

- **Open()**
 - Mode: read/write/change ACL; Events; Lock-delay
 - Create new file or directory?
- **Close()**
- **GetContentsAndStat(), GetStat(), ReadDir()**
- **SetContents(): set all contents; SetACL()**
- **Delete()**
- **Locks: Acquire(), TryAcquire(), Release()**
- **Sequencers: GetSequencer(), SetSequencer(), CheckSequencer()**

Example: Primary Election

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

Five Nodes?! Is that Enough?

- It seems so, but Chubby's users need to be very careful!
- Most typical use: as a name service
 - Bigtable, systems that need a more consistent DNS, ...
- Abusive uses:
 - Publish/subscribe system on Chubby!
 - Repeated open/close calls for polling a file
 - Solution for the above: cache file handles
 - General solution: Review clients' code before they can use shared Chubby cells!
- How can Chubby scale??
 - E.g., to support more files

Chubby Summary

- Lock Service
- UNIX-like file system interface
- Reliability and availability
- Chubby uses Paxos for everything
 - Propagate writes to a file
 - Choosing a Master
 - Even for adding new Chubby servers to a Chubby cell
- Paxos transforms a multi-node service into something that looks very much like one fault-tolerant, albeit slower, server! → pretty close to distributed systems' core goal