

CS 417 – DISTRIBUTED SYSTEMS

Week 5: Part 4
Chubby

Paul Krzyzanowski



© 2022 Paul Krzyzanowski. No part of this content may be reproduced or reposted in whole or in part in any manner without the permission of the copyright owner.

Chubby

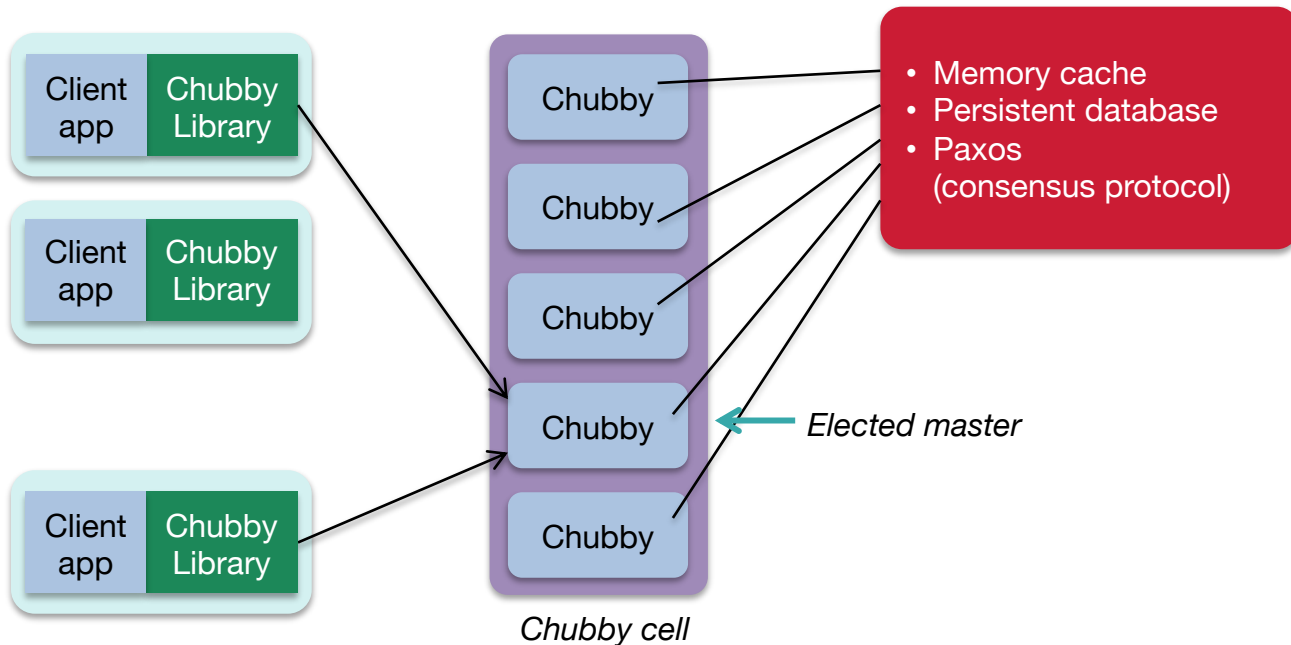
Distributed lock service + simple fault-tolerant file system

- Interfaces
 - File access
 - Event notification
 - File locking
- Chubby is used to:
 - Manage coarse-grained, **long-term locks** (hours or days, not < sec)
 - get/release/check lock – identified with a name
 - Store small **amounts of data** associated with a name
 - E.g., system configuration info, identification of primary coordinators
 - Elect masters

Design priority: availability rather than performance

Chubby Deployment

Client library + a Chubby cell (5 replica servers)



Chubby Master

- Chubby has **at most** one master
 - All requests from the client go to the master

- All other nodes (replicas) must agree on who the master is
 - Paxos consensus protocol used to elect a master
 - Master gets a lease time
 - Re-run master selection after lease time expires to extend the lease
...or if the master fails
 - When a Chubby node receives a proposal for a new master
It will accept it *only* if the old master's lease expired

Simple User-level API for Chubby

- User-level RPC interface
 - Not implemented as a true file system (e.g., under VFS in Linux)
 - Programs must access Chubby via an API
- Look up Chubby nodes via DNS
- Ask any Chubby node for the master node
- File system interface (names, content, and locks)

Chubby: File System Interface

- `/ls/cell/rest/of/name`
 - `/ls`: lock service (common to all Chubby names)
 - `cell`: resolved to a set of servers in a Chubby cell via DNS lookup
 - `/rest/of/name`: interpreted within the cell
- Each file has
 - Name
 - Data
 - Access control list
 - Lock
 - No modification, access times
 - No seek or partial reads/writes; no symbolic links; no moves

Chubby: API

<code>open()</code>	Set mode: read, write & lock, change ACL, event list, lock-delay, create
<code>close()</code>	Close access to an object
<code>GetContentsAndStat()</code>	Read file contents & metadata
<code>SetContents()</code> , <code>SetACL()</code>	Write file contents or ACL
<code>Delete()</code>	Delete an object
<code>Acquire()</code> , <code>TryAcquire()</code> , <code>Release()</code>	Lock operations
<code>GetSequencer()</code>	Sequence # for a lock
<code>SetSequencer()</code>	Associate a sequencer with a file handle
<code>CheckSequencer()</code>	Check if sequencer is valid

Chubby: Locks

- Every file & directory can act as a reader-writer lock
 - Either one client can hold an exclusive (writer) lock
 - Or multiple clients can hold reader locks
- Locks are advisory
- If a client releases a lock, the lock is immediately available
- If a client fails, the lock will be unavailable for a *lock-delay* period (typically 1 minute)

Using Locks for Leader Election

- **Using Chubby locks makes leader election easy**
 - No need for user servers to participate in a consensus protocol ... the programmer doesn't need to figure out Paxos (or Raft)
 - Chubby provides the fault tolerance
 - Participant tries to acquire a lock
 - If it gets it, then it's the master for whatever service it's providing!
- **Example: electing a master & using it to write to a file server**
 - Participant gets a lock, becomes master (*for its service, not Chubby*)
 - Gets a lock sequence count
 - In each RPC to a server, send the **sequence count** to the server
 - During request processing, a server will reject old (delayed) packets

```
if (sequence_count < current_sequence_count)
    reject request /* it must be from a delayed packet */
```

Clients may subscribe to events:

- File content modifications
- Child node added/removed/modified
- Chubby master failed over
- File handle & its lock became invalid
- Lock acquired
- Conflicting lock request from another client

Chubby client caching & master replication

- **At the client**

- Data cached in memory by chubby clients
 - Cache is maintained by a Chubby lease, which can be invalidated
- All clients write through to the Chubby master

- **At the master**

- Writes are propagated via Paxos consensus to all Chubby replicas
 - Data updated in total order – replicas remain synchronized
 - The master replies to a client *after* the writes reach a **majority** of replicas
 - Reads can be acknowledged immediately without consulting the replicas.
- Cache invalidations
 - Master keeps a list of what each client may be caching
 - Invalidations sent by master and are acknowledged by client
 - File is then cacheable again
- Chubby database is backed up to GFS every few hours

Chubby – Apache ZooKeeper™

- Chubby is an internal Google service
- Apache ZooKeeper is an open-source centralized service for locking and storing shared services
- Built based on the Chubby paper
 - Uses Zab instead of Paxos for consensus (another protocol – roughly similar to Paxos but with a focus on log replication)
 - Replicated servers
 - Shared hierarchical namespace
 - Stored in memory and organized into files and directories
 - Locking
 - Event notification



The End