Chubby: Introduction

This lesson will introduce Chubby and its use cases.

We'll cover the following

- Goal
- What is Chubby?
- Chubby use cases
  - Leader/master election
  - Naming service (like DNS)
  - Storage (small objects that rarely change)
  - Distributed locking mechanism
  - When not to use Chubby
- Background
  - Chubby and Paxos

Goal#

Design a highly available and consistent service that can store small objects and provide a locking mechanism on those objects.

What is Chubby?#

Chubby is a service that provides a distributed locking mechanism and also stores small files. Internally, it is implemented as a **key/value store** that also provides a locking mechanism on each object stored in it. It is extensively used in various systems inside Google to provide storage and coordination services for systems like **GFS** and **BigTable**. Apache **ZooKeeper** is the open-source alternative to Chubby.

In sum, Chubby is a centralized service offering developer-friendly interfaces (to acquire/release locks and create/read/delete small files). Moreover, it does all this with just a few extra lines of code to any existing application without a lot of modification to application logic.

Chubby use cases#

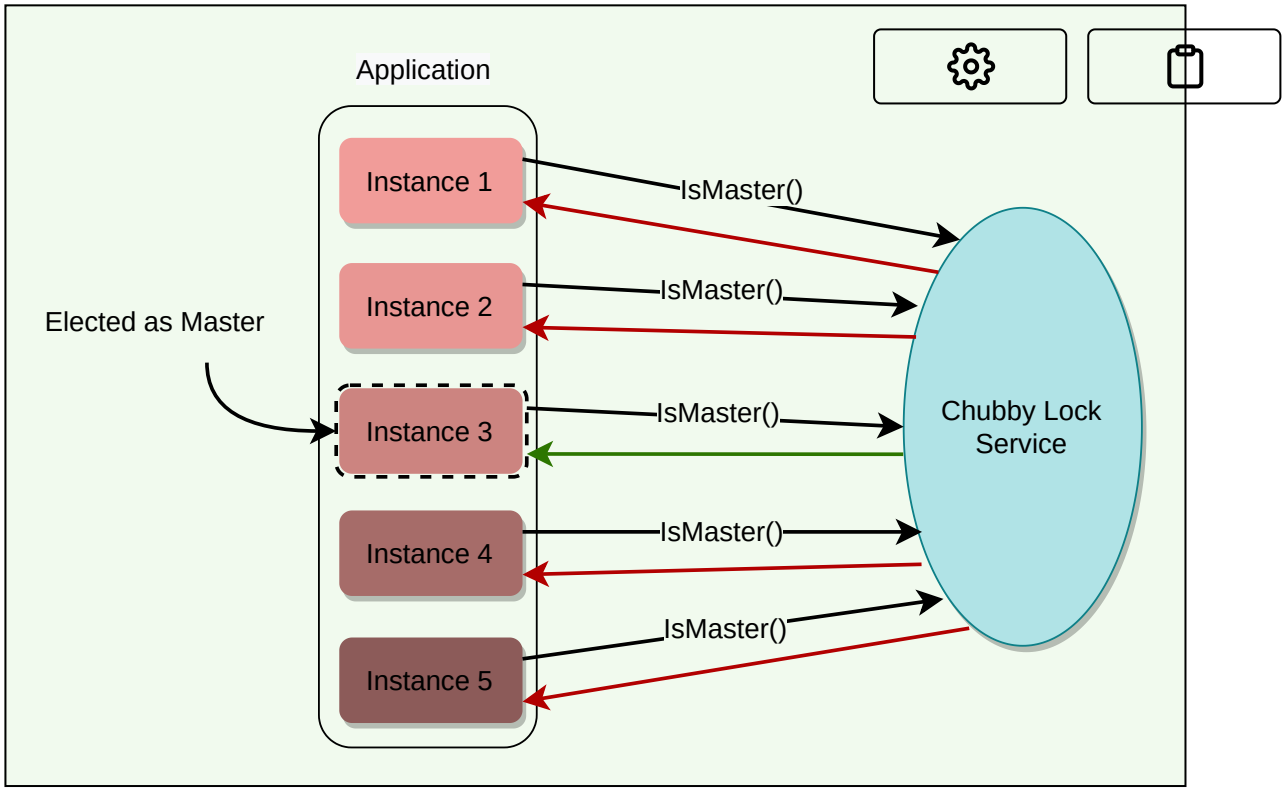
Primarily Chubby was developed to provide a reliable locking service. Over time, some interesting uses of Chubby have evolved. Following are the top use cases where Chubby is practically being used:

- Leader/master election
- Naming service (like DNS)
- Storage (small objects that rarely change)
- Distributed locking mechanism

Let’s look into these use cases in detail.

Leader/master election#

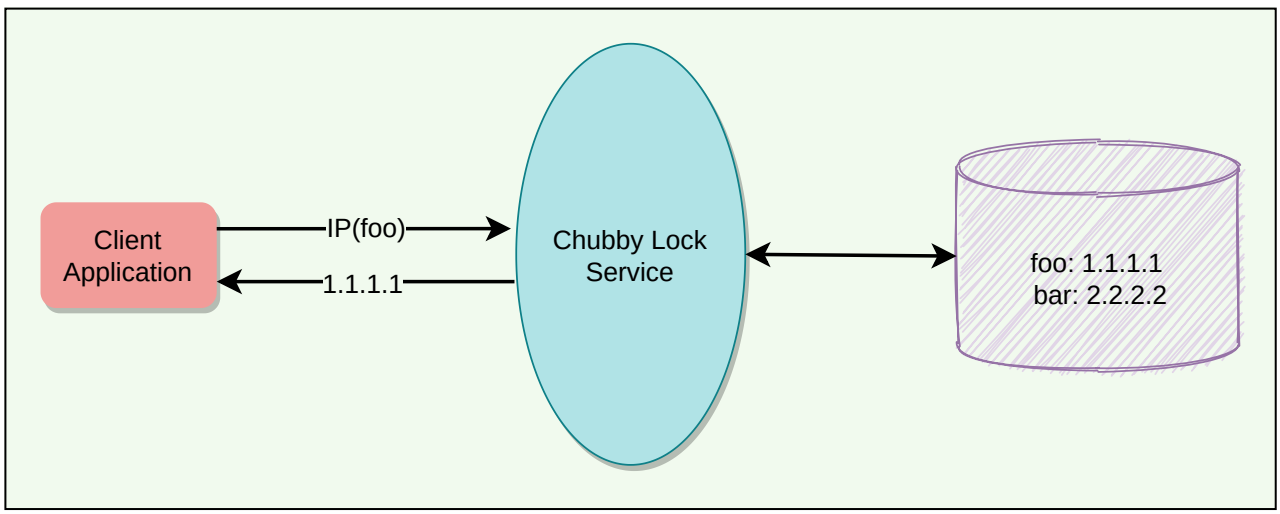
Any lock service can be seen as a consensus service, as it converts the problem of reaching consensus to handing out locks. Basically, a set of distributed applications compete to acquire a lock, and whoever gets the lock first gets the resource. Similarly, an application can have multiple replicas running and wants one of them to be chosen as the leader. Chubby can be used for leader election among a set of replicas, e.g., the leader/master for GFS and BigTable.



Chubby being used for leader election

Naming service (like DNS)#

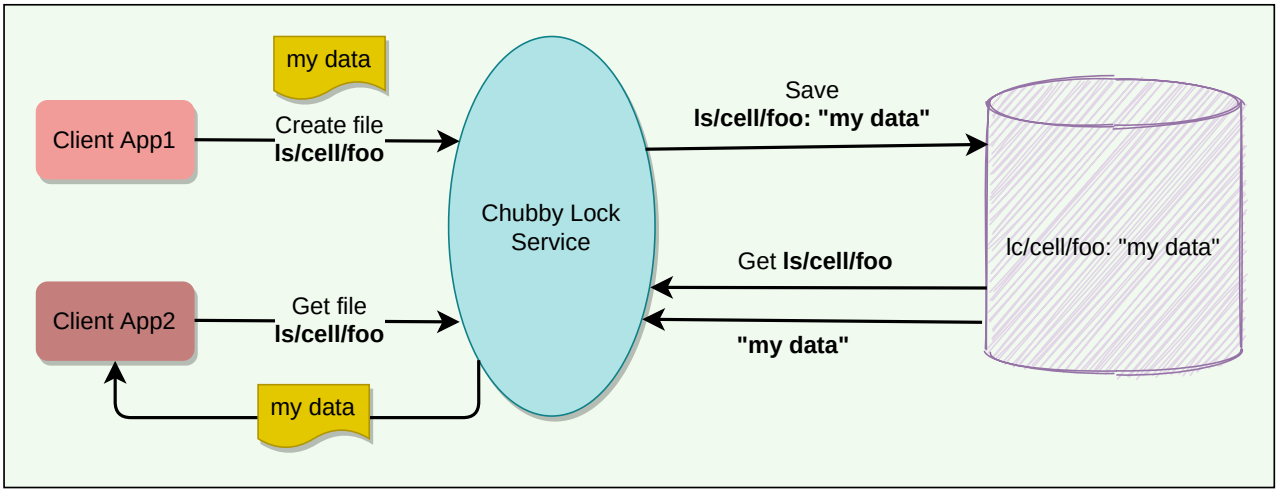
It is hard to make faster updates to DNS due to its time-based caching nature, which means there is generally a potential delay before the latest DNS mapping is effective. As a result, chubby has replaced DNS inside Google as the main way to discover servers.



Chubby as DNS

Storage (small objects that rarely change)#

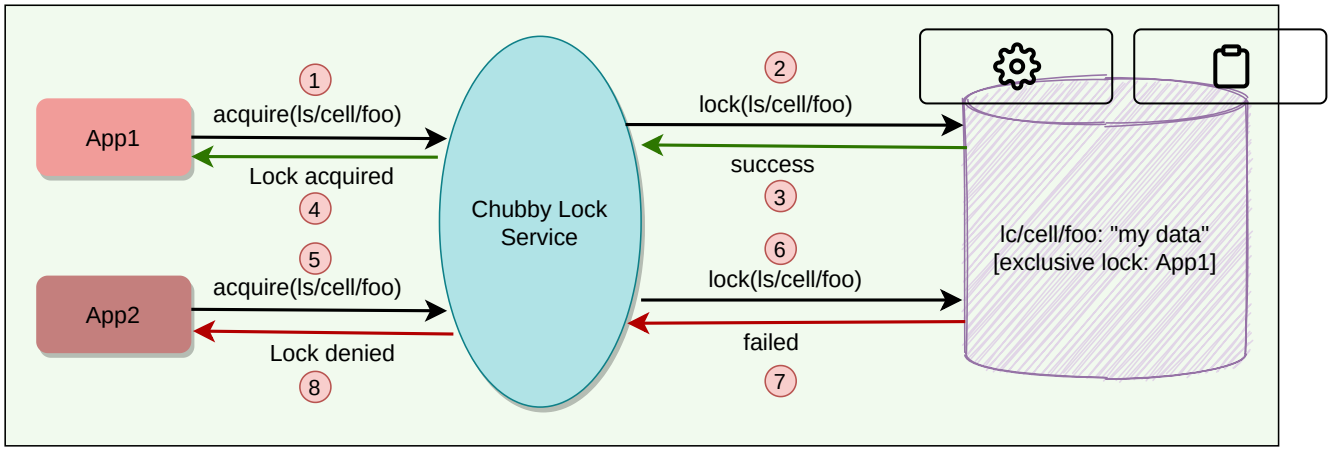
Chubby provides a Unix-style interface to reliably store small files that do not change frequently (complementing the service offered by GFS). Applications can then use these files for any usage like DNS, configs, etc. GFS and Bigtable store their metadata in Chubby. Some services use Chubby to store ACL files.



Chubby as a storage service for small objects

Distributed locking mechanism#

Chubby provides a developer-friendly interface for coarse-grained distributed locks (as opposed to fine-grained locks) to synchronize distributed activities in a distributed environment. All an application needs is a few code lines, and Chubby service takes care of all the lock management so that developers can focus on application business logic. In other words, we can say that Chubby provides mechanisms like semaphores and mutexes for a distributed environment.



Chubby as a distributed locking service

All these use cases are discussed in detail later.

At a high level, Chubby provides a framework for distributed consensus. All the above-mentioned use cases have emerged from this core service.

When not to use Chubby#

Because of its design choices and proposed usage, Chubby should not be used when:

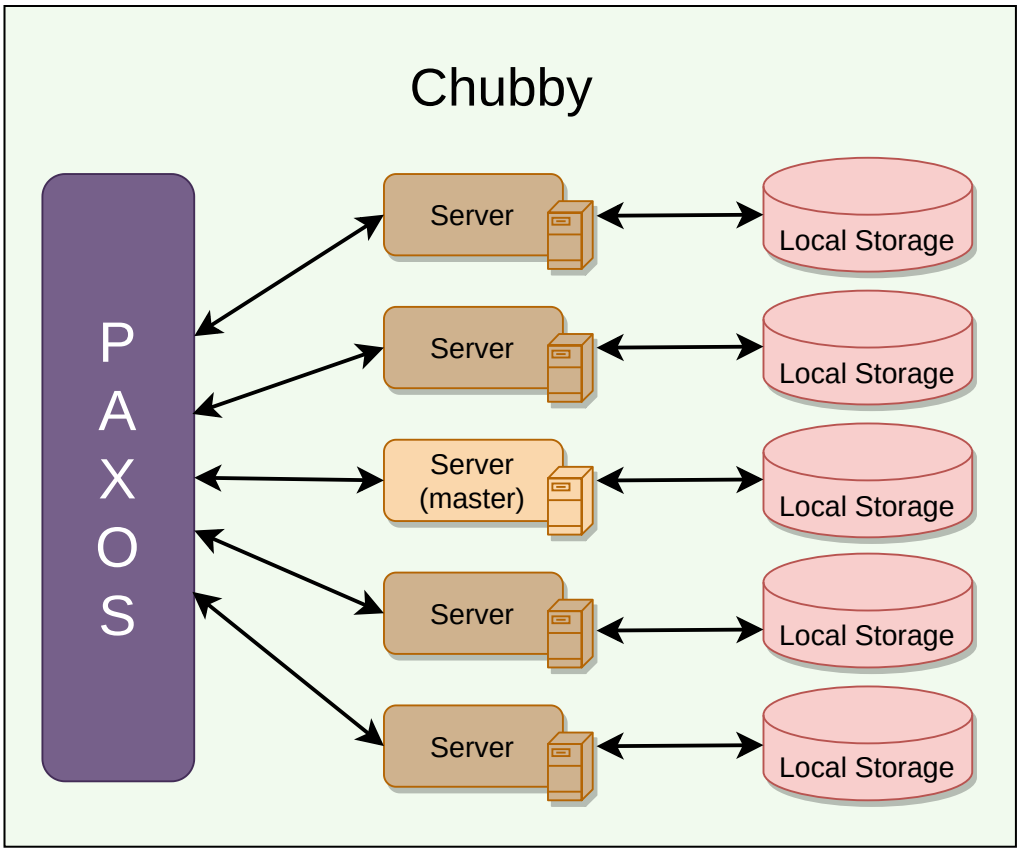
- Bulk storage is needed.
- Data update rate is high.
- Locks are acquired/released frequently.
- Usage is more like a publish/subscribe model.

Background#

Chubby is neither really a research effort, nor does it claim to introduce any new algorithms. Instead, chubby describes a certain design and implementation done at Google in order to provide a way for its clients to synchronize their activities and agree on basic information about their environment. More precisely, at Google, it has become primary to implement the above-mentioned use cases.

Chubby and Paxos#

Paxos (https://en.wikipedia.org/wiki/Paxos_(computer_science)) plays a major role inside Chubby. Readers familiar with Distributed Computing recognize that getting all nodes in a distributed system to agree on anything (e.g., election of primary among peers) is basically a kind of distributed consensus problem. Distributed consensus using Asynchronous Communication is already solved by Paxos protocol, and Chubby actually uses Paxos underneath to manage the state of the Chubby system at any point in time.



Chubby uses Paxos to manage its system

← Back
Mock Interview: Kafka

Next →
High-level Architecture

Mark as Completed


Report an Issue