Chubby lock service, which is intended to provide coarse-grained locking as well as reliable (though low-volume) storage for a loosely-coupled distributed system. Chubby provides an interface much like a distributed file system with advisory locks, but the design emphasis is on availability and reliability, as opposed to high performance. Many instances of the service have been used for over a year, with several of them each handling a few tens of thousands of clients concurrently

It is intended for use within a loosely-coupled distributed system consisting of moderately large numbers of small machines connected by a high-speed network. For example, a Chubby instance (also known as a Chubby cell) might serve ten thousand 4-processor machines connected by 1Gbit/s Ethernet. Most Chubby cells are confined to a single data centre or machine room, though we do run at least one Chubby cell whose replicas are separated by thousands of kilometres. The purpose of the lock service is to allow its clients to synchronize their activities and to agree on basic information about their environment. The primary goals included reliability, availability to a moderately large set of clients, and easy-to-understand semantics; throughput and storage capacity were considered secondary.

Chubby’s client interface is similar to that of a simple file system that performs whole-file reads and writes, augmented with advisory locks and with notification of various events such as file modification.

Chubby has two main components that communicate via RPC:

* a server,
* a library that client applications link against, with optional third component, a proxy server.

A Chubby cell consists of a small set of servers (typically five) known as replicas, placed so as to reduce the likelihood of correlated failure (for example, in different racks). The replicas use a distributed consensus protocol to elect a master; the master must obtain votes from a majority of the replicas, plus promises that those replicas will not elect a different master for an interval of a few seconds known as the master lease. The master lease is periodically renewed by the replicas provided the master continues to win a majority of the vote

The replicas maintain copies of a simple database, but only the master initiates reads and writes of this database. All other replicas simply copy updates from the master, sent using the consensus protocol.

Clients find the master by sending master location requests to the replicas listed in the DNS. Non-master replicas respond to such requests by returning the identity of the master. Once a client has located the master, the client directs all requests to it either until it ceases to respond, or until it indicates that it is no longer the master. Write requests are propagated via the consensus protocol to all replicas; such requests are acknowledged when the write has reached a majority of the replicas in the cell. Read requests are satisfied by the master alone; this is safe provided the master lease has not expired, as no other master can possibly exist. If a master fails, the other replicas run the election protocol when their master leases expire; a new master will typically be elected in a few seconds