Distributed Co-ordination Apache Zookeeper

Pramod Bhatotia

http://homepages.inf.ed.ac.uk/pbhatoti/

Credits for the lecture material:
Zookeeper ATC paper and presentation

THE UNIVERSITY *of* EDINBURGH

Co-ordination in distributed systems

- (Dynamic) configuration
- Synchronization
- Leader election
- Group membership
- Barriers
- Locks
- ...

Challenges in distributed systems

- The network is unreliable
- Process may crash/fail in arbitrary ways
- The network messages may arrive arbitrarily

Zookeeper

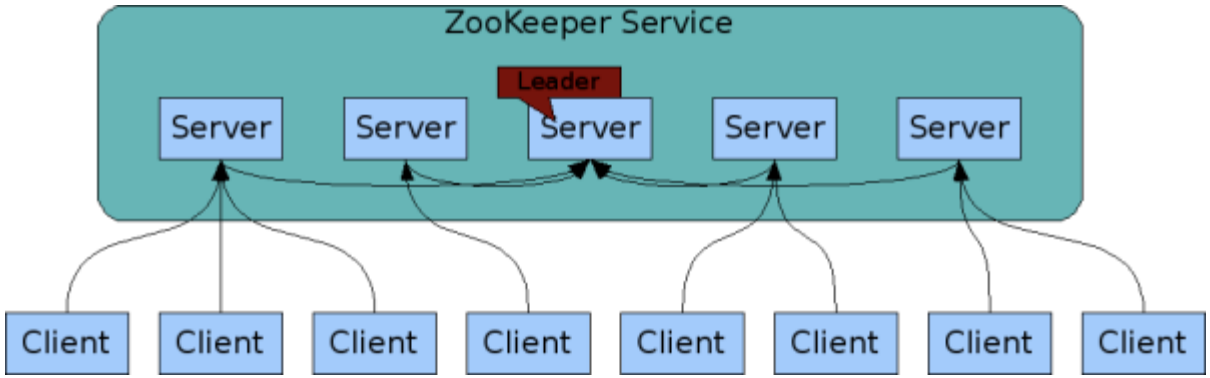
- A co-ordination (micro-)kernel
  - Minimalistic APIs that can be used to build a wide-range of co-ordination primitives
- APIs are wait-free
  - No blocking primitives in ZooKeeper
  - Blocking can be implemented by a client
  - Deadlock free!

Zookeeper design principles

- Zookeeper = FIFO ordering for clients + Linearzible writes + Wait-free APIS

- Guarantees
  - Client requests are processed in FIFO order
  - Writes to ZooKeeper are linearizable
  - Clients receive notifications of changes before the changed data becomes visible

Zookeeper architecture

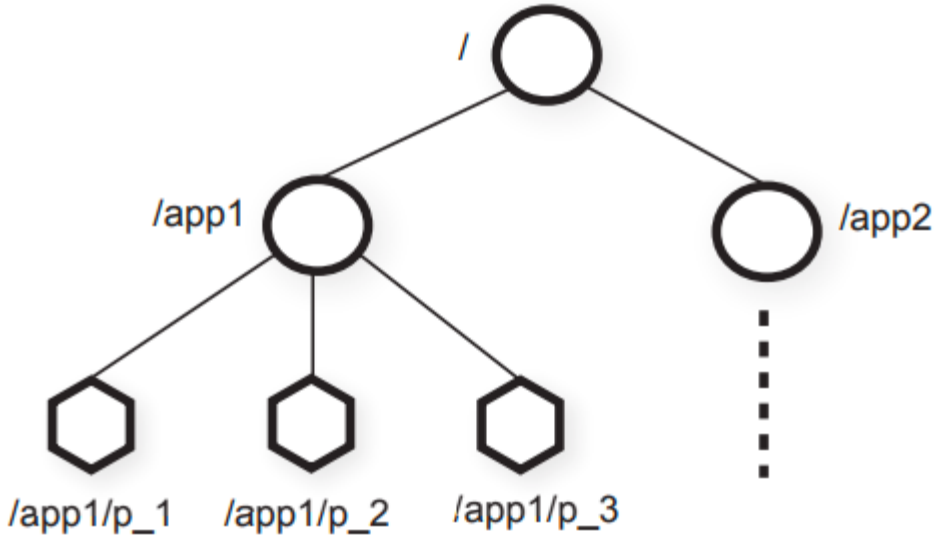


Terminology:

- **Clients:** users of the Zookeeper service
- **Servers:** process providing the Zookeeper service
- **Session:** Clients establish a session when connecting to Zookeeper

Zookeeper data model

Abstraction: A set of data nodes (znodes) organized in a hierarchal namespace



Znodes can store data

znodes

- Znodes are accessed similar to UNIX filesystem namespace
- Znodes can be classified as:
  - **Regular:** created and deleted by clients explicitly
  - **Ephemeral:** can be deleted explicitly or by the system itself when the session terminates (the node that created it)
- Flags:
  - **Sequential:** montonically increasing counters
  - **Watch flags**

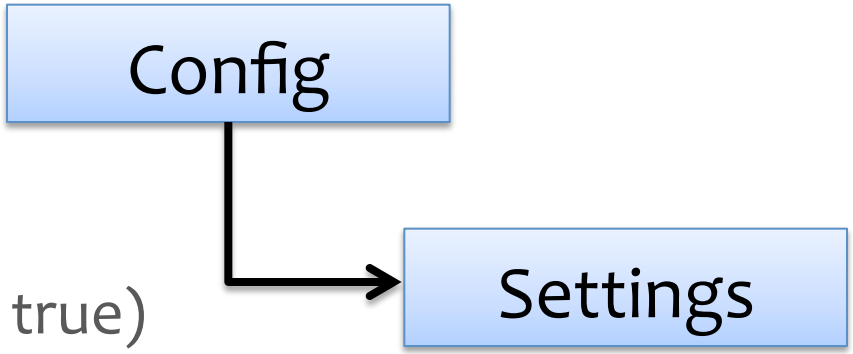
Zookeeper APIs

- create(path, data, flags)
- delete(path, version)
- exists(path, watch)
- getData(path, watch)
- setData(path, data, version)
- getChildren(path, watch)
- Sync()

Zookeeper use-cases

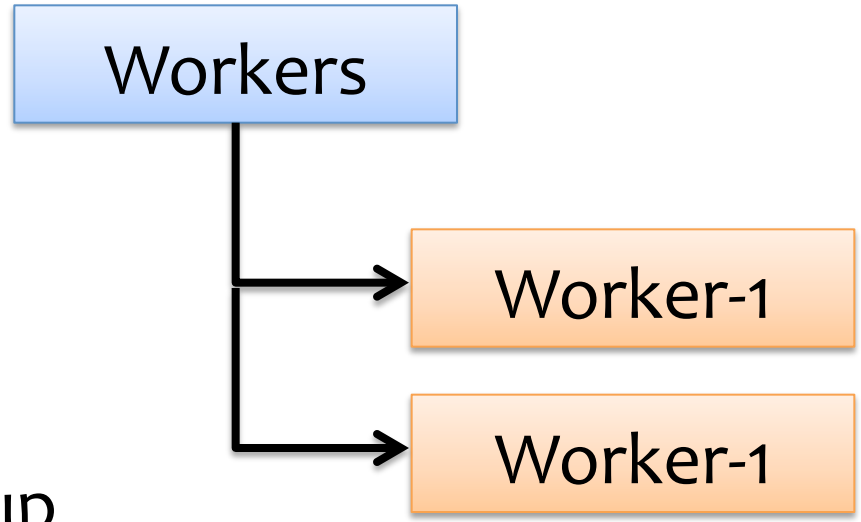
How to use Zookeeper to implement distributed co-ordination protocols?

Example 1: Configuration



- Workers get configuration
  - getData("…/config/settings", true)
- Admin change the config
  - setData("…/config/settings", newConf-1)
- Workers notified of change and get the new settings
  - getData("…/config/settings", true)

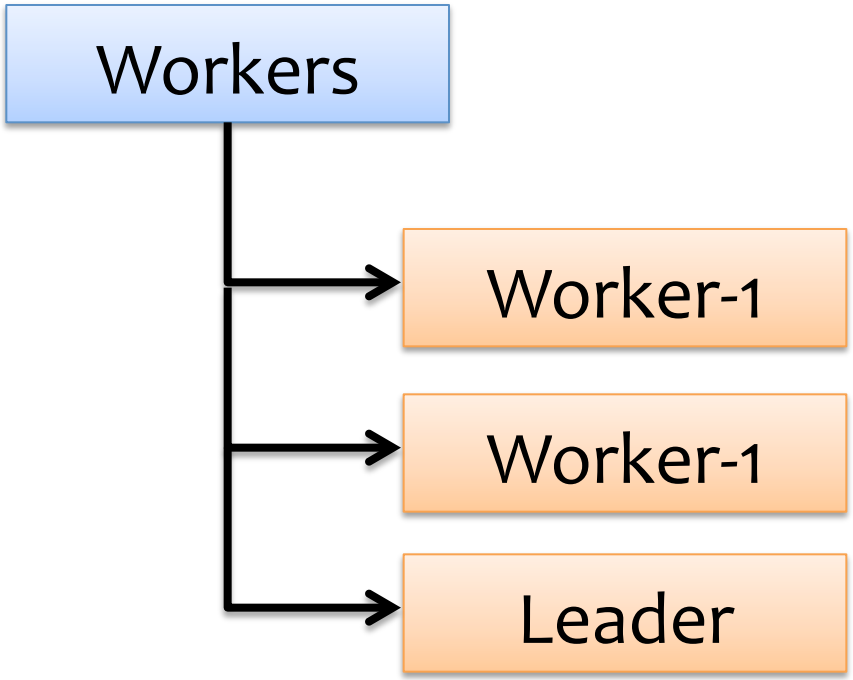
Example 2: Group membership



- Register serverName in group
  - Create("…/workers/workerName", hostInfo, EPHEMERAL)
- List group members
  - getChildren("…/workers", true)

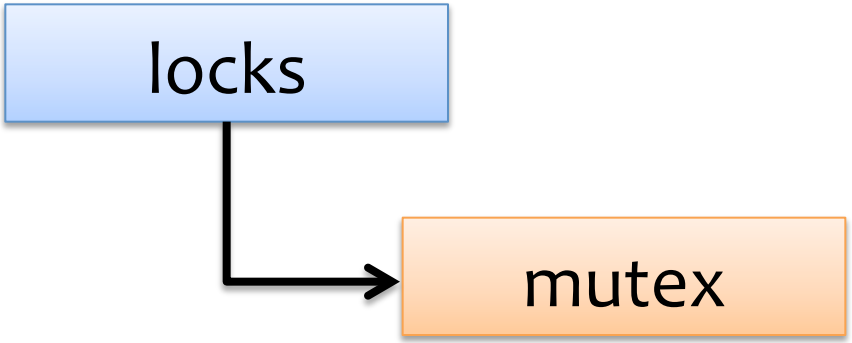
Example 3: Leader Election

- getData("…/workers/leader", true)
- If successful follow the leader described in the data and exit
- create("…/workers/leader", hostname, EPHERMERAL)
- If successful lead and exit
- Goto step 1



Example 4: Locks

- create("…/locks/mutex", EPHEMERAL)
- If succeed then lock acquired
- Else, getData("…/locks/mutex", true)
- Goto step 1



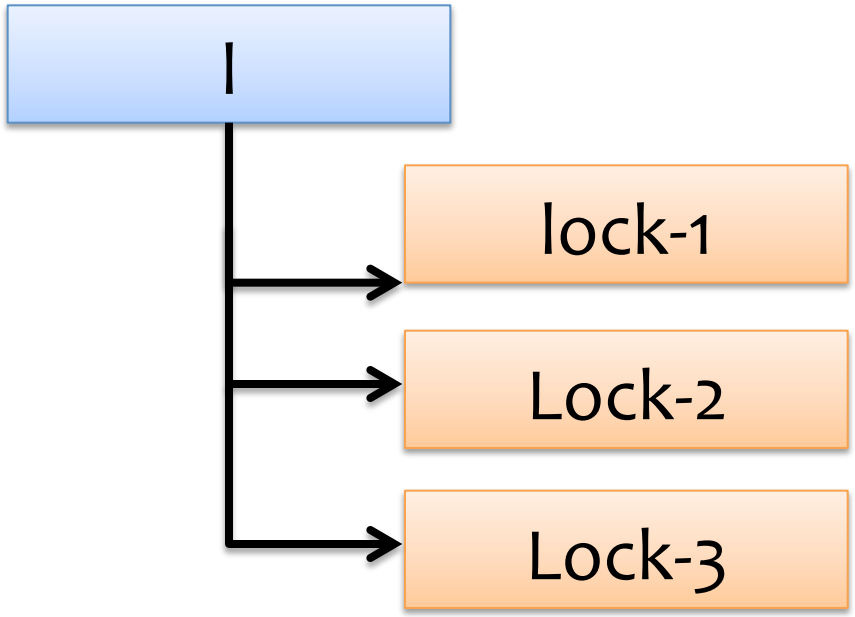
Example 5: Locks without herding

Lock

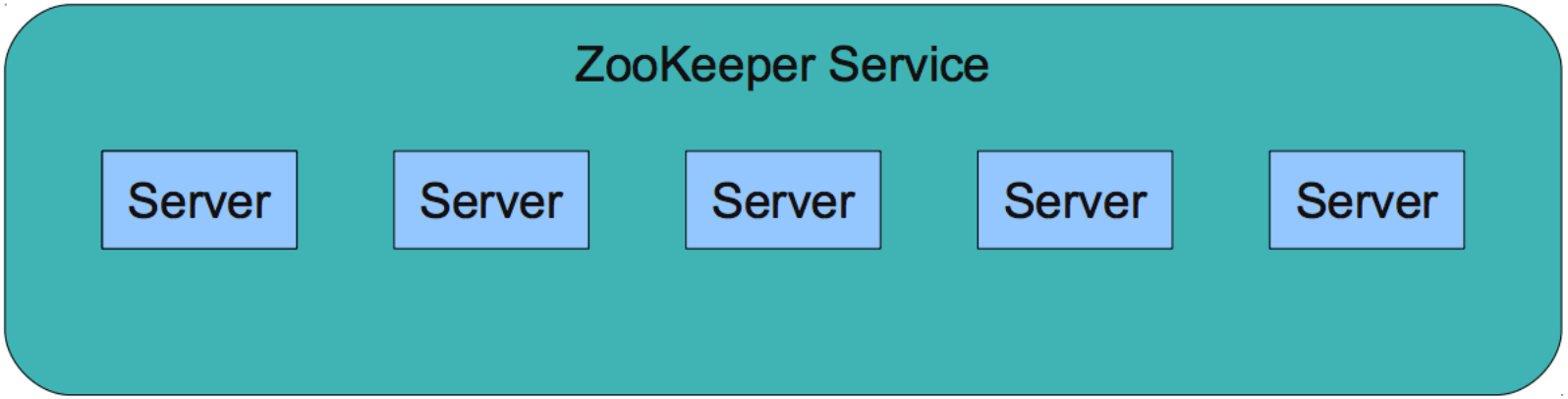
```
1 n = create(l + "/lock-", EPHEMERAL|SEQUENTIAL)
2 C = getChildren(l, false)
3 if n is lowest znode in C, exit
4 p = znode in C ordered just before n
5 if exists(p, true) wait for watch event
6 goto 2
```

Unlock

```
1 delete(n)
```



System Implementation



- All servers have a copy of the state in memory
- A leader is elected at startup
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change
  - We need 2f+1 machines to tolerate f failures

Summary

- Apache Zookeeper
  - Co-ordination in distributed systems
  - A distributed co-ordination kernel
  - Usage to build powerful primitives

- Resources:
  - Zookeeper [ATC'11]: https://zookeeper.apache.org/
  - Chubby [OSDI'06]: https://research.google.com/archive/chubby.html

Thanks!

http://homepages.inf.ed.ac.uk/pbhatoti/