ZooKeeper

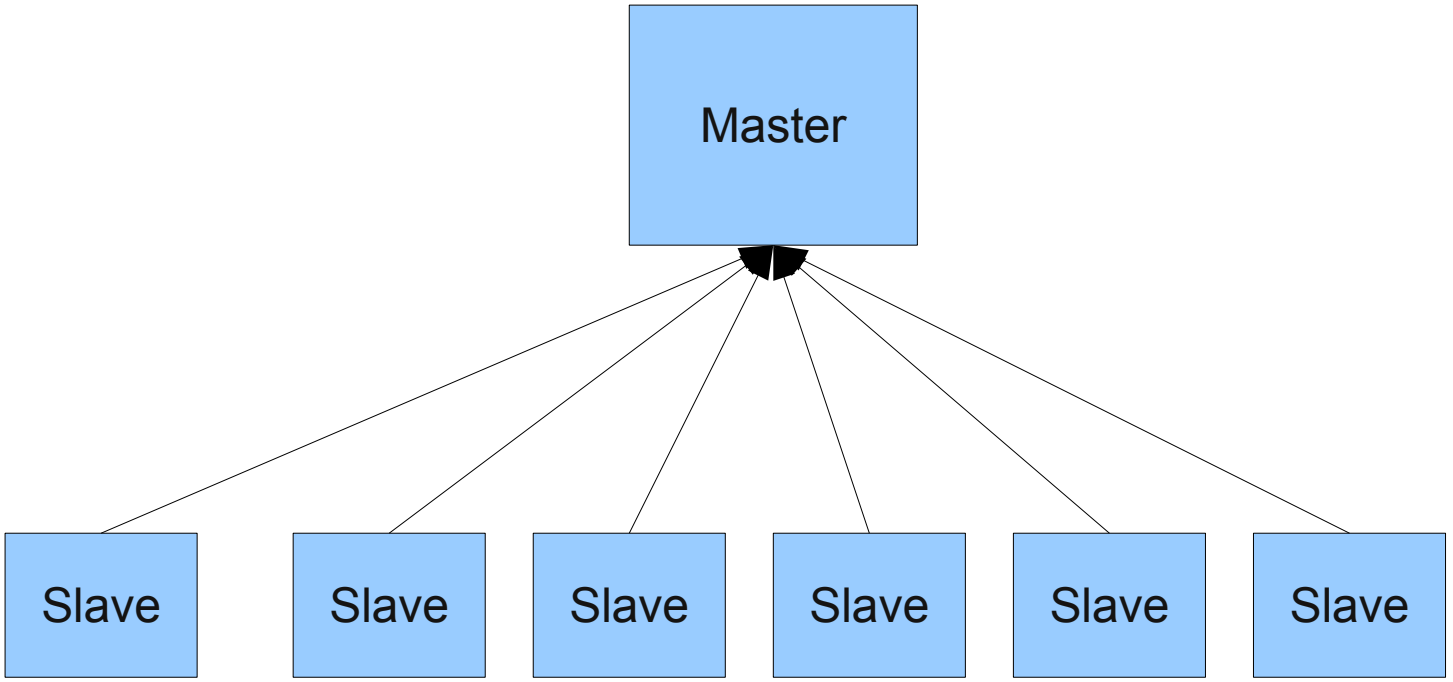
Wait-free coordination for Internet-scale systems

Patrick Hunt and Mahadev (Yahoo! Grid)
Flavio Junqueira and **Benjamin Reed** (Yahoo! Research)

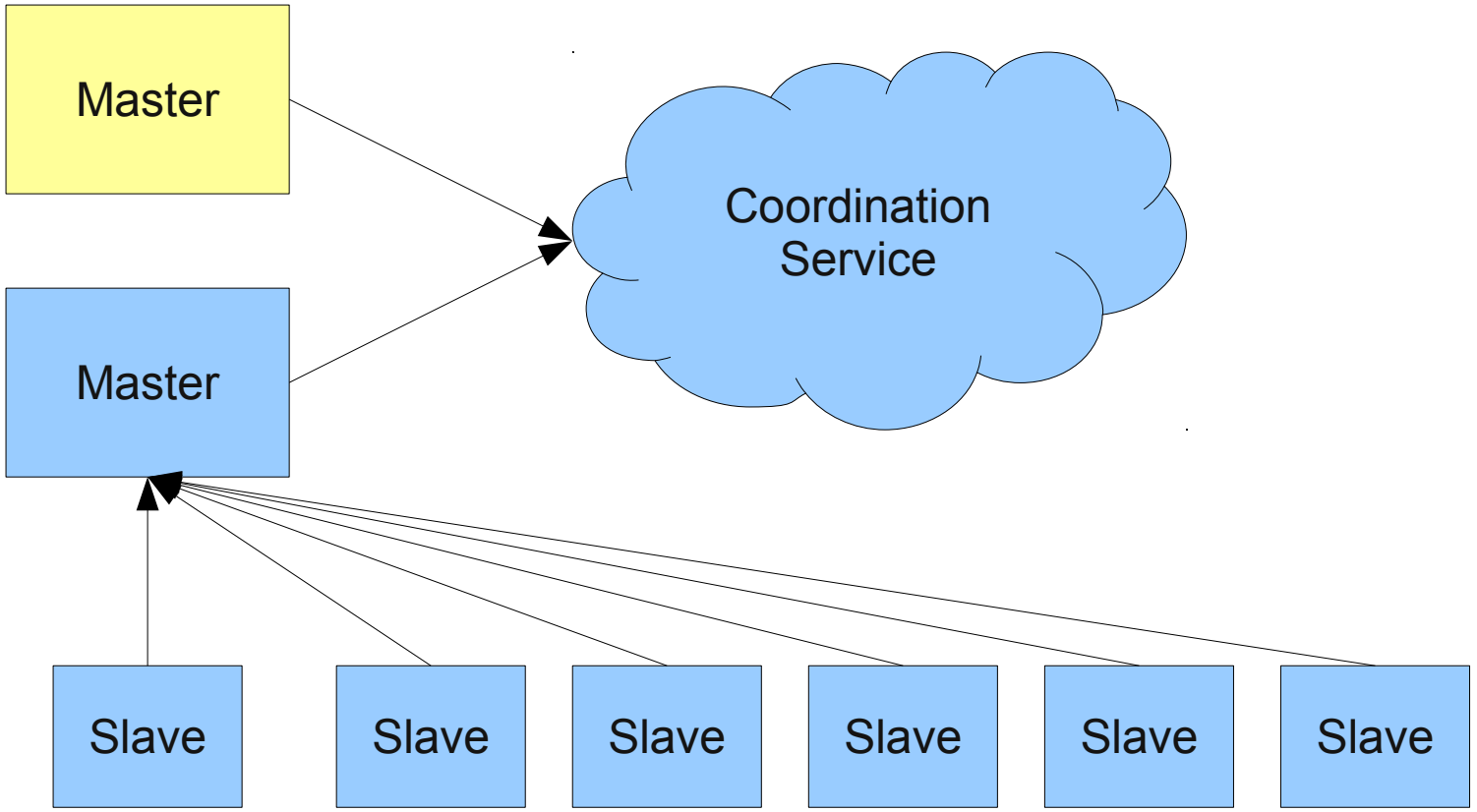
Internet-scale Challenges

- Lots of servers, users, data
- FLP, CAP
- Mere mortal programmers

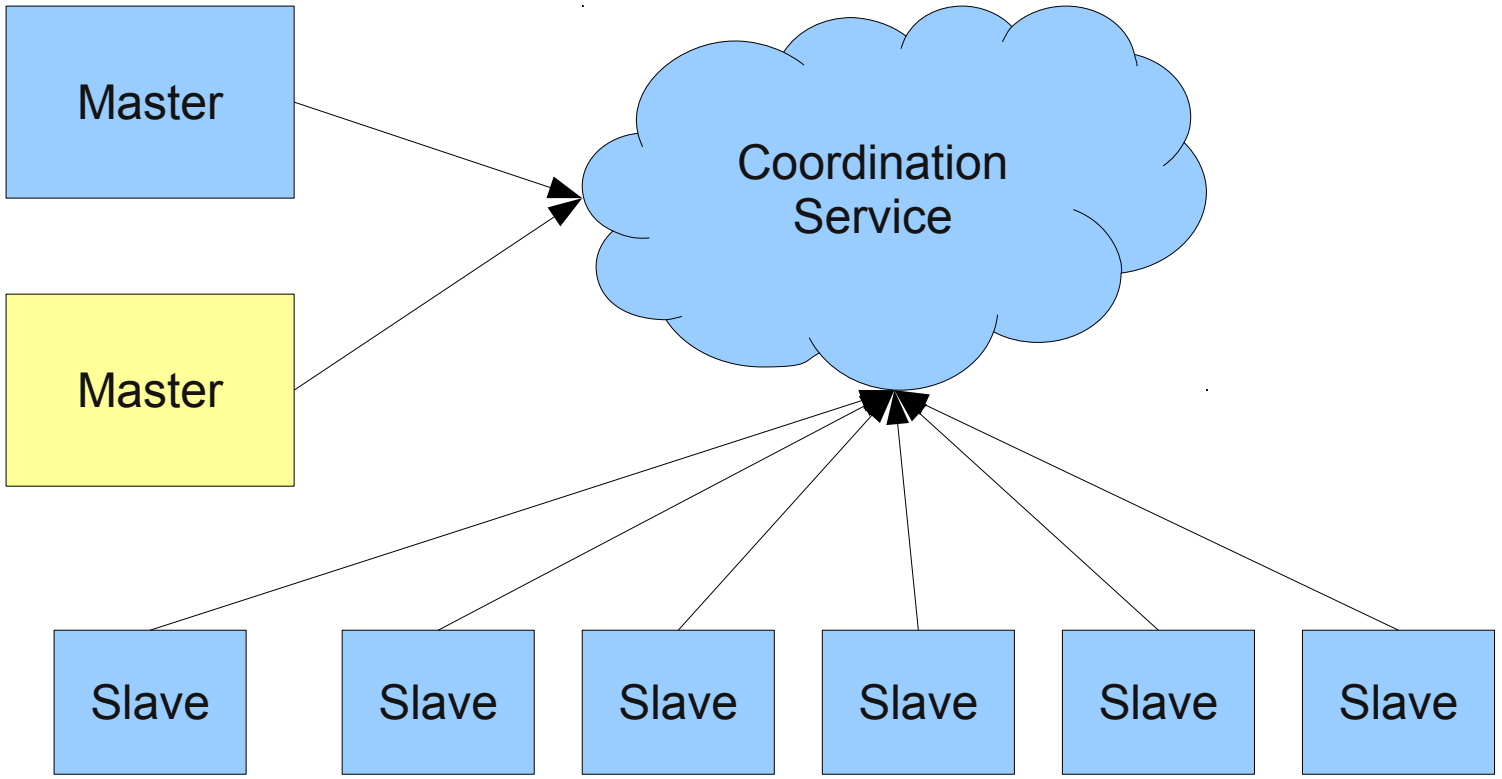
Classic Distributed System



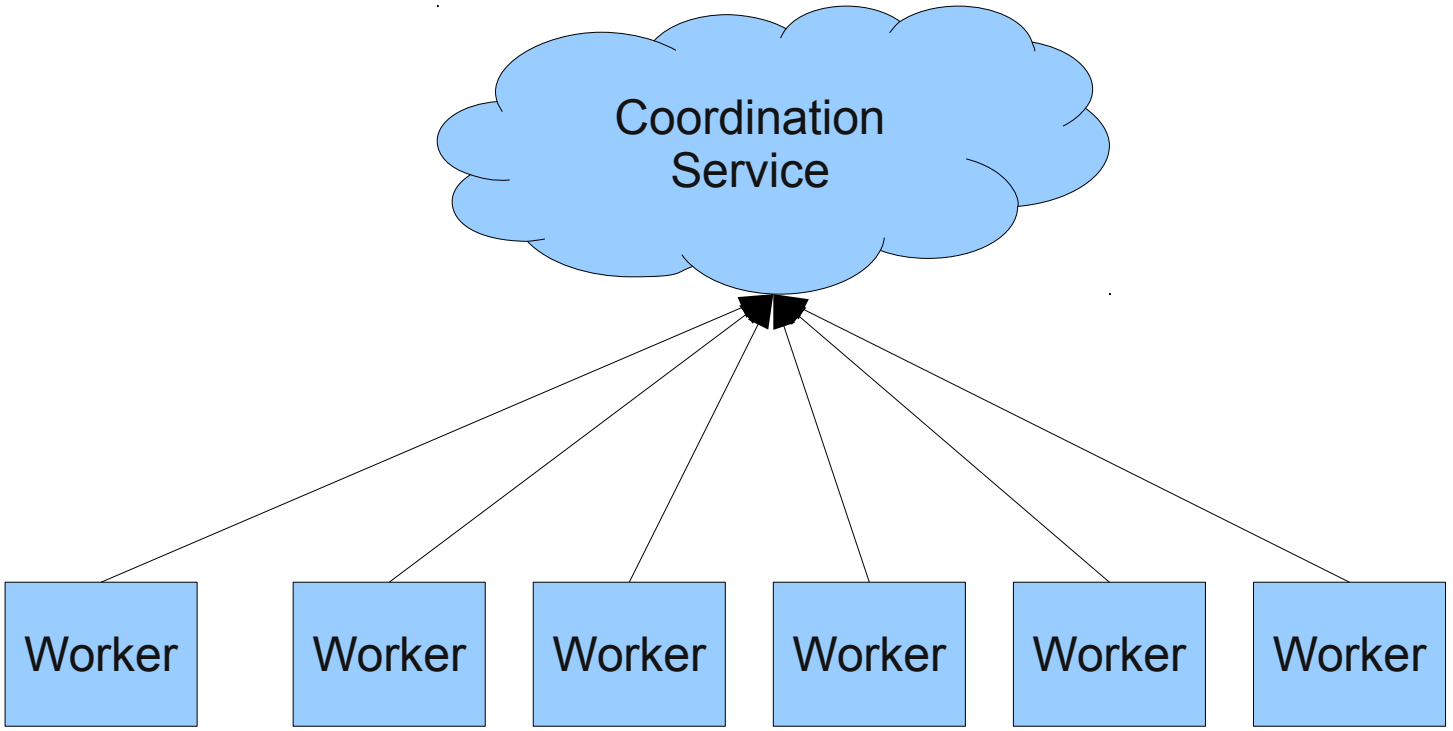
Fault Tolerant Distributed System



Fault Tolerant Distributed System



Fully Distributed System



YAHOO!

What is coordination?

- Group membership
- Leader election
- Dynamic Configuration
- Status monitoring
- Queuing
- Barriers
- Critical sections

Goals

- Been done in the past
  - ISIS, distributed locks (Chubby, VMS)
- High Performance
  - Multiple outstanding ops
  - Read dominant
- General (Coordination Kernel)
- Reliable
- Easy to use

wait-free

- Pros
  - Slow processes cannot slow down fast ones
  - No deadlocks
  - No blocking in the implementations
- Cons
  - Some coordination primitives are blocking
  - Need to be able to efficiently wait for conditions

Serializable vs Linearizability

- Linearizable writes
- Serializable read (may be stale)
- Client FIFO ordering

Change Events

- Clients request change notifications
- Service does timely notifications
- Do not block write requests
- Clients get notification of a change before they see the result of a change

Solution

Order + wait-free + change events = coordination

ZooKeeper API

String create(path, data, acl, flags)

void delete(path, expectedVersion)

Stat setData(path, data, expectedVersion)

(data, Stat) getData(path, watch)

Stat exists(path, watch)

String[] getChildren(path, watch)

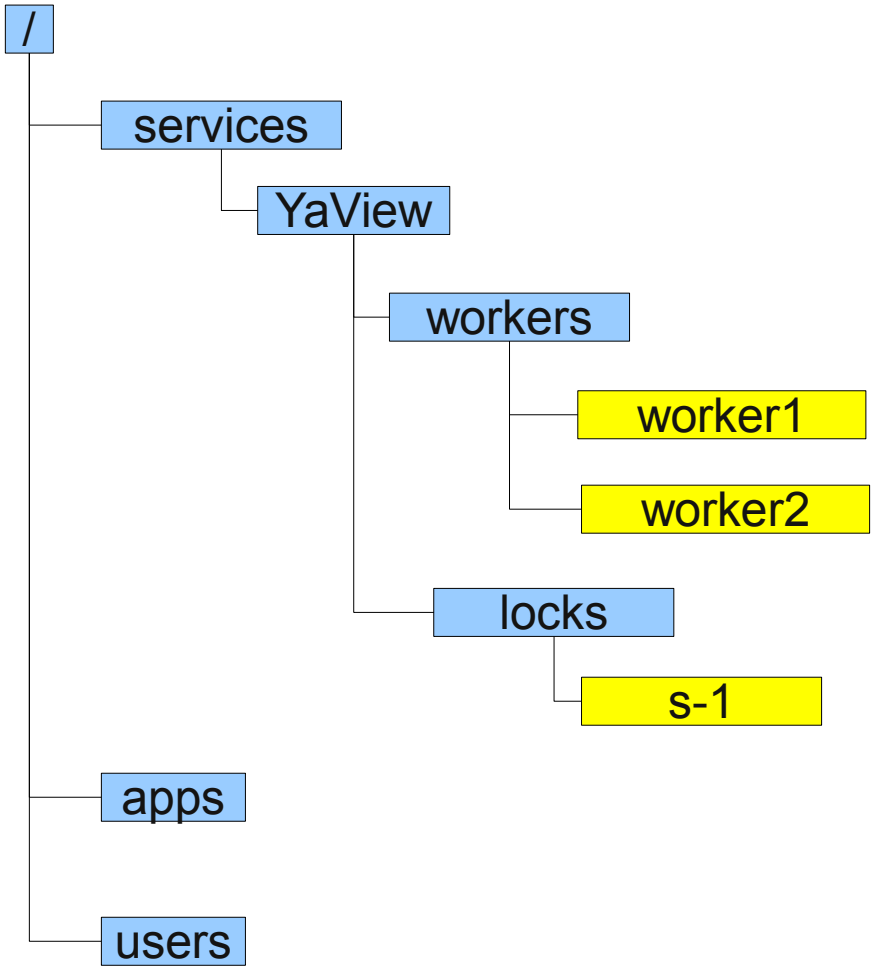
void sync()

Stat setACL(path, acl, expectedVersion)

(acl, Stat) getACL(path)

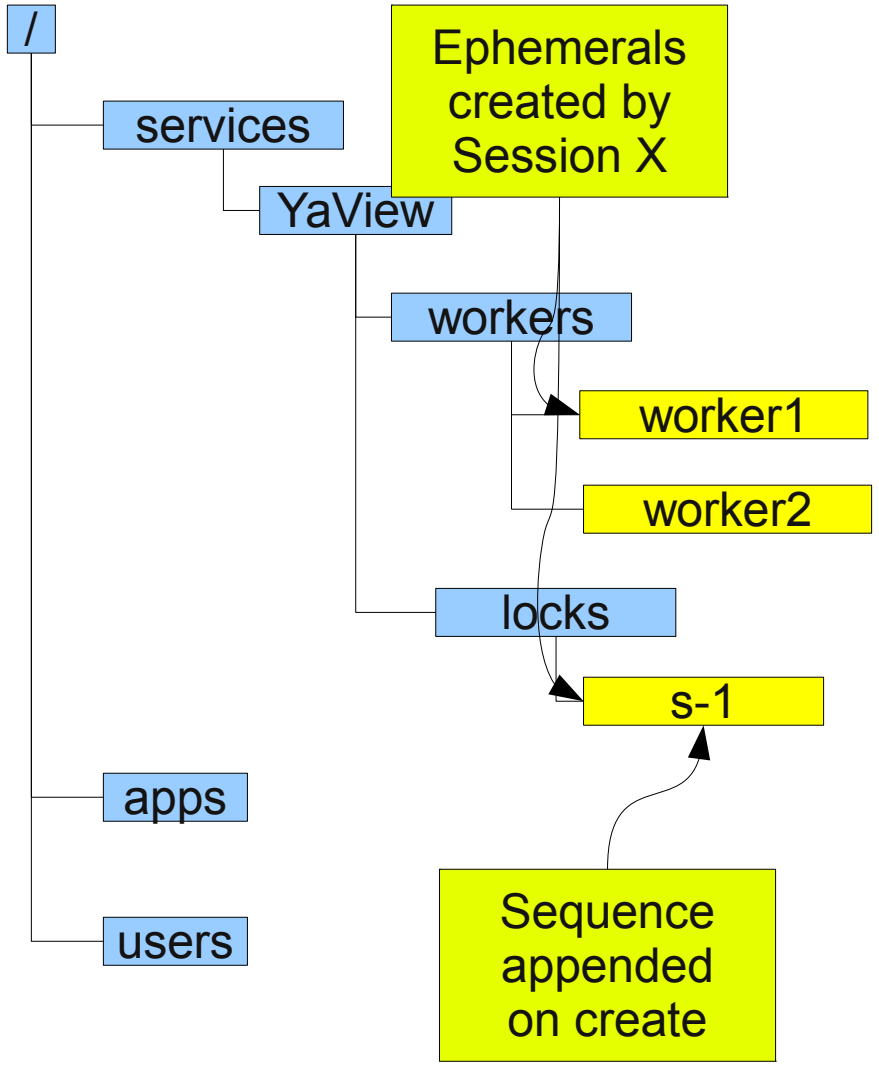
Data Model

- Hierarchal namespace (like a file system)
- Each znode has data and children
- data is read and written in its entirety

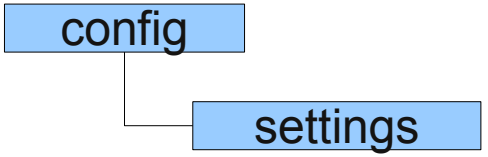


Create Flags

- Ephemeral: znode deleted when creator fails or explicitly deleted
- Sequence: append a monotonically increasing counter



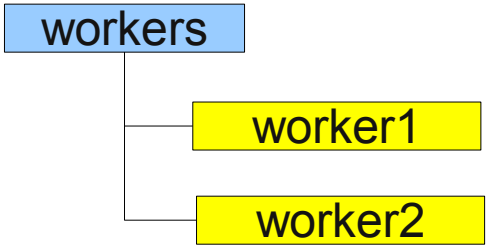
Configuration



- Workers get configuration
  - getData(“.../config/settings”, true)
- Administrators change the configuration
  - setData(“.../config/settings”, newConf, -1)
- Workers notified of change and get the new settings
  - getData(“.../config/settings”, true)

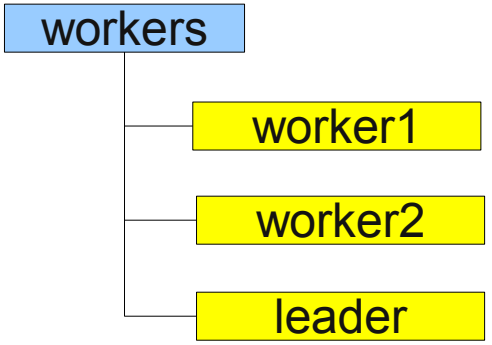
Group Membership

- Register serverName in group
  - create(“.../workers/workerName”, hostInfo, EPHEMERAL)
- List group members
  - listChildren(“.../workers”, true)



Leader Election

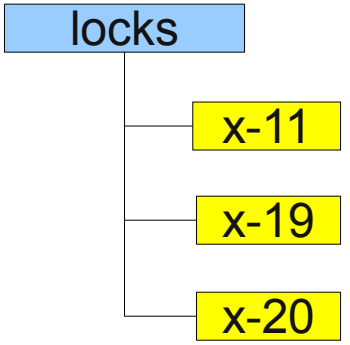
- getData(“.../workers/leader”, true)
- if successful follow the leader described in the data and exit
- create(“.../workers/leader”, hostname, EPHEMERAL)
- if successful lead and exit
- goto step 1



If a watch is triggered for “.../workers/leader”, followers will restart the leader election process

Locks

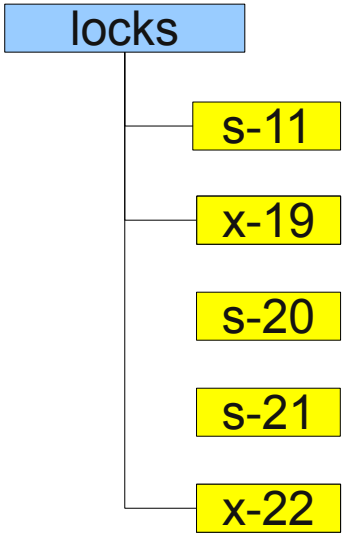
- id = create(“.../locks/x-”, SEQUENCE|EPHEMERAL)
- getChildren(“.../locks”/, false)
- if id is the 1st child, exit
- exists(name of last child before id, true)
- if does not exist, goto 2)
- wait for event
- goto 2)



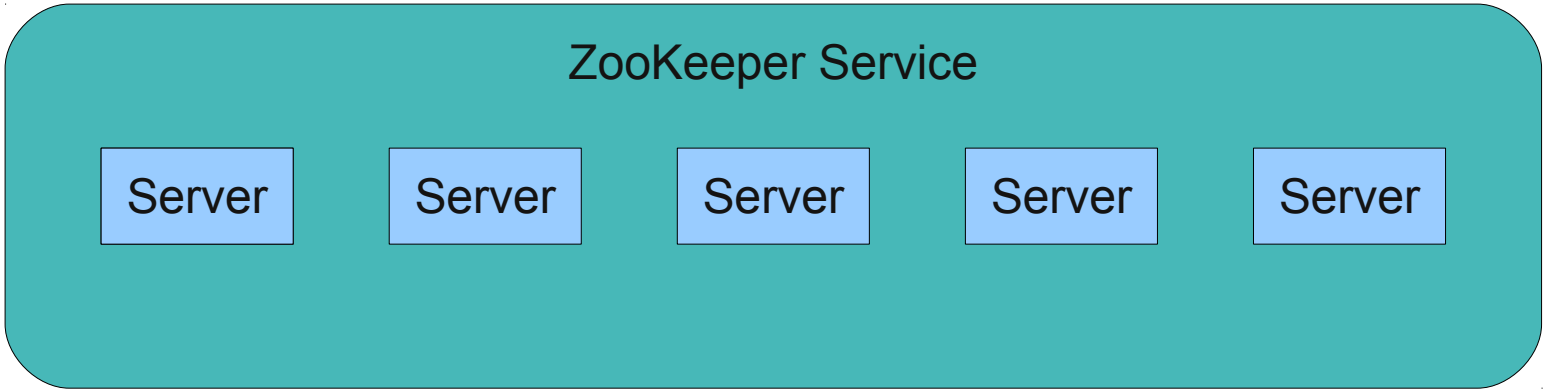
Each znode watches one other.
No herd effect.

Shared Locks

- id = create(“.../locks/s-”, SEQUENCE|EPHEMERAL)
- getChildren(“.../locks”/, false)
- if no children that start with x- before id, exit
- exists(name of the last x- before id, true)
- if does not exist, goto 2)
- wait for event
- goto 2)



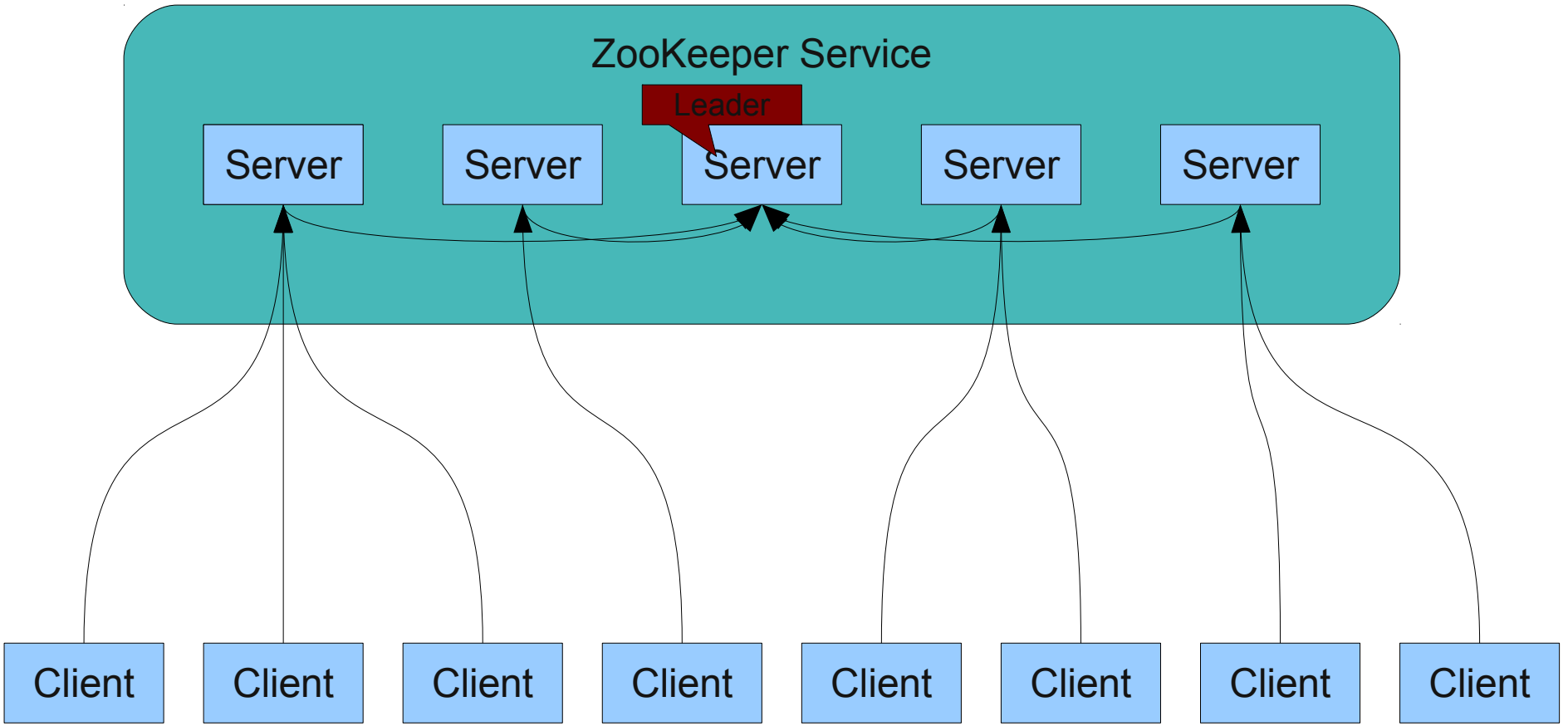
ZooKeeper Servers



- 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

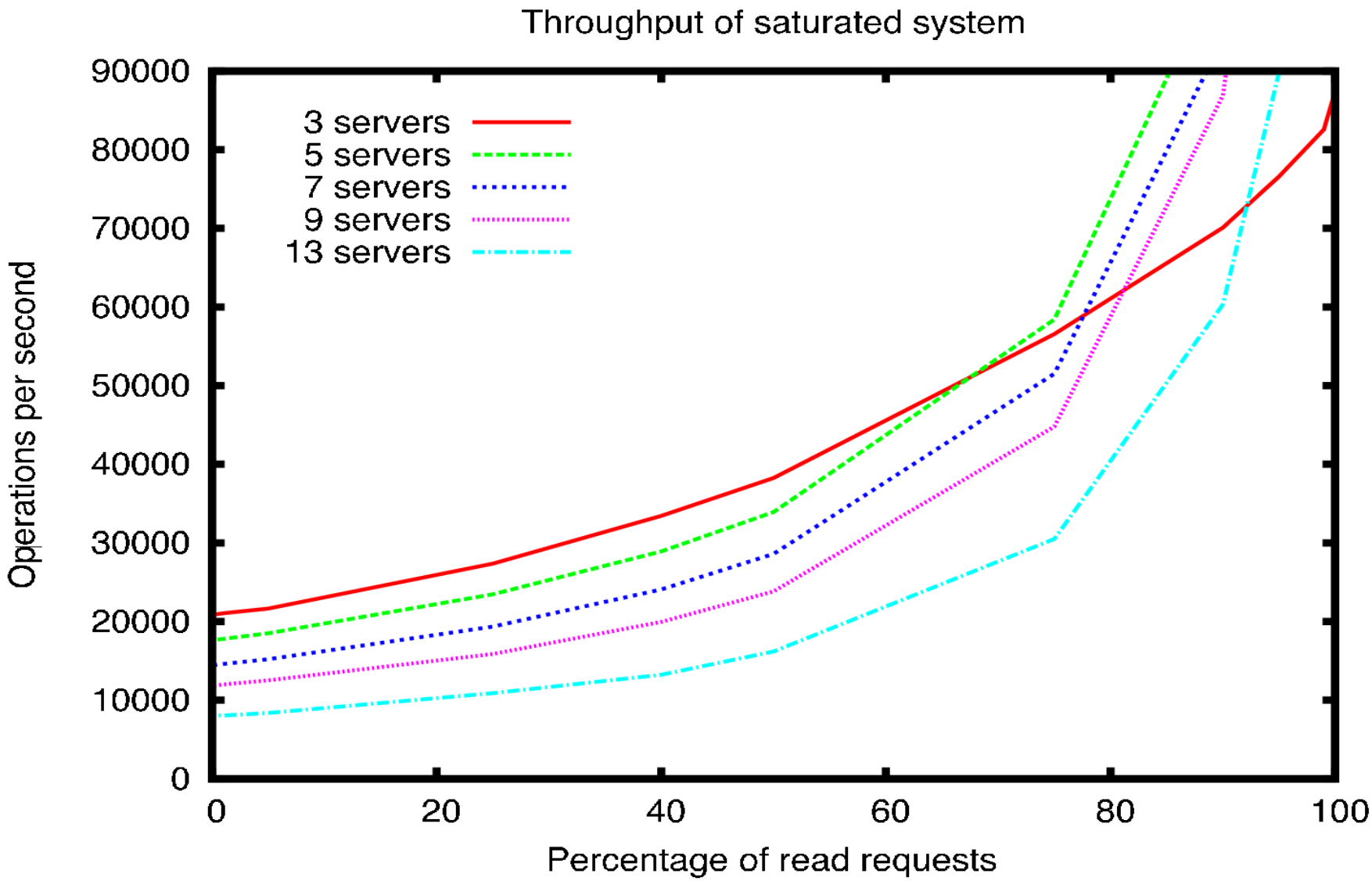
  We need 2f+1 machines to tolerate f failures

ZooKeeper Servers



YAHOO!

Current Performance



Summary

- Easy to use
- High Performance
- General
- Reliable
- Release 3.3 on Apache
  - See http://hadoop.apache.org/zookeeper
  - Committers from Yahoo! and Cloudera