Programação em Sistemas Distribuídos

MEI-MI-MSI

2018/19

Apache ZooKeeper

Prof. António Casimiro

What is ZooKeeper?

Ciências ULisboa

- A highly available coordination service, open-source, scalable and distributed, offering a set of useful functions to build distributed applications:
  - configuration, synchronization, consensus, group membership, leader election, naming
- Overcomes difficulty of implementing these kinds of services reliably

Zookeeper Guarantees

Ciências ULisboa

- Clients will never detect old data
- Clients will get notified of a change to data they are watching within a bounded period of time
- All requests from a client will be processed in order
- All results received by a client will be consistent with results received by all other clients

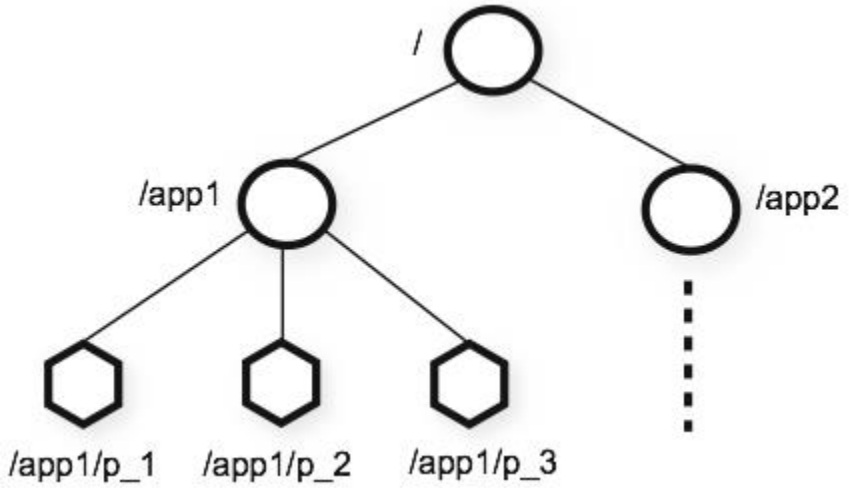
Zookeeper consistency properties

Ciências ULisboa

- Linearizable writes
  - All requests that update the state of ZooKeeper are serializable and respect real time order
- FIFO client order
  - All requests are in order that they were sent by client

Zookeeper Data Model (1)

- Znode
  - In-memory data node
  - Hierarchical namespace, much like a standard file system
  - Znode can have data associated with it, and children
  - UNIX-like notation for path
- Types of Znode
  - Regular
  - Ephemeral – exists as long as session that created it is active
- Flags of Znode
  - Sequential flag



Zookeeper Data Model (2)

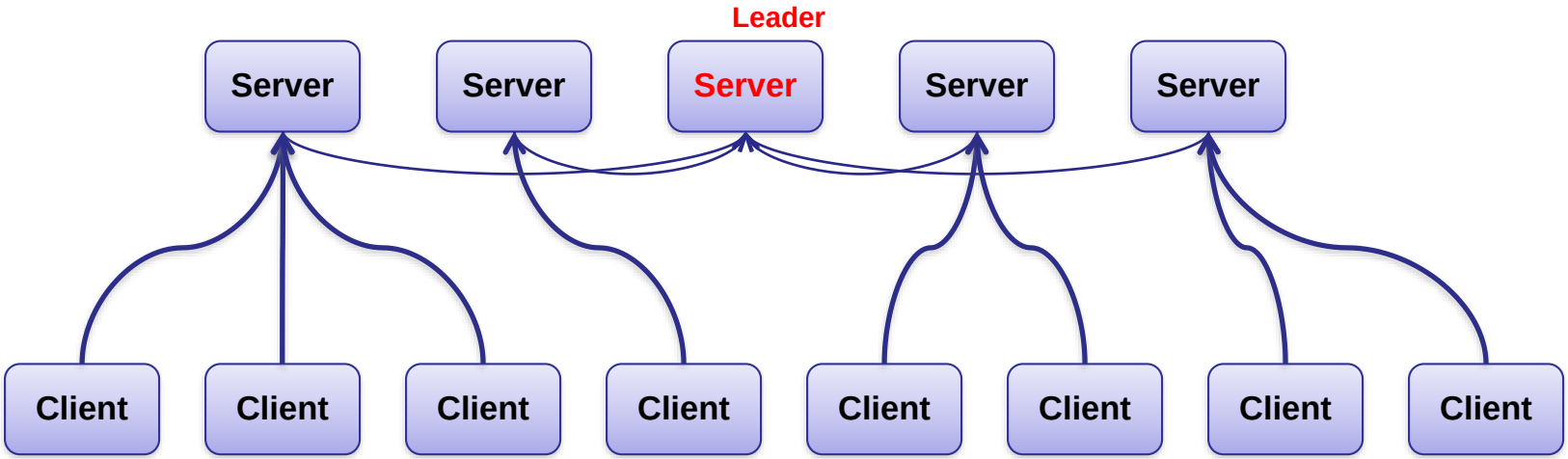
Ciências ULisboa

- Watch Mechanism
  - Get change notifications
- Other properties of Znode
  - Data is read/written in its entirety
  - Znode was NOT designed for data storage, instead it stores meta-data or configuration data, akin to coordination functions
  - information like: timestamps, versions, sequences, membership, etc.
- Session
  - A connection to server from client is a session
  - Timeout mechanism

Client Interface

- Two versions: synchronous and asynchronous
  - *create(path, data, acl, flags)*
  - *delete(path, version)*
  - *exist(path, watch)*
  - *getData(path, watch)*
  - *setData(path, data, version)*
  - *getChildren(path, watch)*
- Only asynchronous
  - *sync(path)*

ZooKeeper Service



- All servers store a copy of the data (in memory)
- A leader is elected at start up
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers has made change persistent

Examples of primitives

Ciências ULisboa

- Configuration Management
  - For dynamic configuration purpose
  - Simplest way is to make up a Znode ***c*** for saving configuration
  - Other processes set the watch flag on ***c***
  - The notification just indicates that there was an update, but other updates might have followed
  - It is up to the client to check the new state (possibly holding several updates, which occurred between the moment when the notification was sent and the moment when the client is checking)

Examples of primitives

- Rendezvous
  - Creates a Znode ***r*** to hold configuration data
  - When master arrives, fills Znode ***r*** with its configuration data (e.g., own IP address)
  - Slave nodes watch Znode ***r*** to become aware of changes (e.g., master is available)

- Group Membership
  - Create a Znode ***g***
  - Each process creates a Znode under ***g*** in ephemeral mode
  - Watch ***g*** for group information

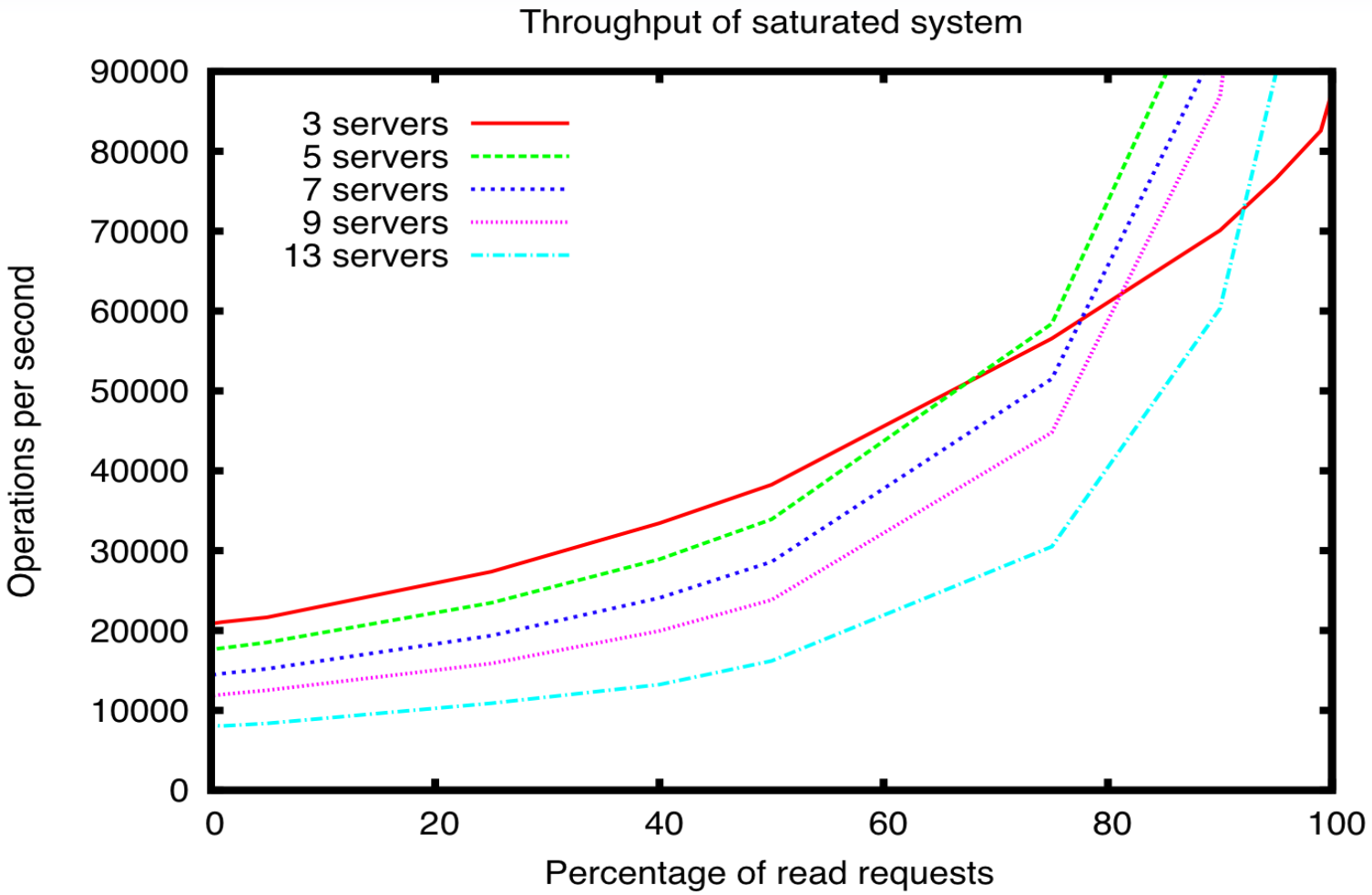
Examples of primitives

Ciências ULisboa

- Simple Lock
  - Create a Znode ***l*** for locking
  - If one gets to create ***l*** he gets the lock
  - Others who fail to create, watch ***l***
  - Problems: herd effect

- Double Barrier
  - To synchronize the beginning and the end of computation
  - Create a Znode ***b***, and every process needs to register on it, by adding a Znode under ***b***
  - When number of nodes reach threshold, start the computation
  - Remove Znode under ***b*** when computation is complete
  - Leave when no Znodes remain under ***b***

Performance



Who uses Zookeeper?

Ciências ULisboa

- **Hadoop MapReduce** – The next generation of Hadoop MapReduce (called "Yarn") uses ZooKeeper.
- **Yahoo!** – ZK is used for a myriad of services inside Yahoo! for doing leader election, configuration management, cluster management, load balancing, sharding, locking, work queues, group membership, etc.
- **Deepdyve** - Does search for research and provides access to high quality content using advanced search technologies. ZK is used to manage server state, control index deployment and a myriad of other tasks
- **Katta** - Katta serves distributed Lucene indexes in a grid environment. ZK is used for node, master and index management in the grid
- **Hbase** – HBase is an open-source distributed column-oriented database on Hadoop. Uses ZK for master election, server lease management, bootstrapping, and coordination between servers
- **Rackspace** – Email & Apps team uses ZK to co-ordinate sharding, handling responsibility changes, and distributed locking
- And many more…

Zookeeper deployment

Ciências ULisboa

- Multi-tenant
  - Provides mechanisms (quotas, connection management, chroot) for multi-user environments
- Recipes
  - Reusable code libraries
- Bindings
  - Java, C, Perl, Python (and others)

Ciências
ULisboa

Design elements

"As is", culled from several sources on manuals and the Internet, with aim of giving design and programming examples

Essential internals

- Replication. No SPOF
  - Leader + Followers, 2f+1 nodes can tolerate failure of f nodes
  - If the leader fails, a new one is elected
- All replicas can accept requests
  - It's a system designed for few writes and many reads
- Consistency model: completely ordered history of updates
  - All updates go through the leader
  - Consistency using consensus – well known ways are Paxos algorithm, State Machine Replication, etc.
- Uses ZooKeeper Atomic Broadcast protocol (ZAB)
  - ZAB – very similar to multi-Paxos, but the differences are real
  - The implementation builds upon the FIFO property of TCP streams

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(path)*

Ephemeral Nodes, Watches

Ciências ULisboa

- Ephemeral nodes
  - Present as long as the session that created it is active
  - Cannot have child nodes
- Watches
  - Tell me when something changes. E.g., configuration data
  - One time trigger. Must be reset by the client if interested in future notifications
  - Not a full fledged notification system. Client should verify what changed in the node contents after receiving the watch event
  - Ordering guarantee: a client will never see a change for which it has set a watch until it first sees the watch event
  - Default watcher – notified of state changes in the client (connection loss, session expiry, …)

Zookeeper Session

Ciências ULisboa

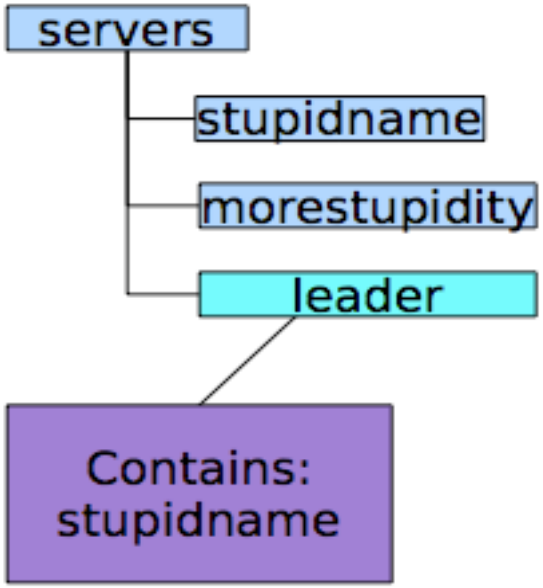
- ZK client establishes connection to ZK service, using a language binding (Java, C, Perl, Python)
- If a list of servers is provided when connecting, retry connection with other servers until it is (re)established
- When a client gets a handle to the ZK service, ZK creates a ZK session, represented as a 64-bit number
- If reconnected to a different server within the session timeout, session remains the same
- Session is kept alive by periodic PING requests from the client library

Use cases

- Leader Election
- Group Membership
- Work Queues
- Configuration Management
- Cluster Management
- Load Balancing
- Sharding

Leader Election

1. getdata("/servers/leader", true)
2. if successful follow the leader described in the data and exit
3. create("/servers/leader", hostname, EPHEMERAL)
4. if successful lead and exit
5. goto step 1



Leader Election in Java
Connect to server

Ciências ULisboa

```
package ZKElection;

import java.io.IOException;
import java.nio.ByteBuffer;

import org.apache.zookeeper.CreateMode;
import org.apache.zookeeper.KeeperException;
import org.apache.zookeeper.KeeperException.Code;
import org.apache.zookeeper.Watcher;
import org.apache.zookeeper.ZooDefs.Ids;
import org.apache.zookeeper.ZooKeeper;
import org.apache.zookeeper.data.Stat;

public class ZKClient {
    private ZooKeeper zk;

    public ZKClient(Integer port) throws IOException, KeeperException, InterruptedException{
        zk = new ZooKeeper("127.0.0.1", port, (Watcher) this);
        leaderElection(port);
    }
```

Leader Election in Java
Elect leader

Ciências ULisboa

```
public void leaderElection(Integer id) throws KeeperException, InterruptedException {

    byte[] leaderID = ByteBuffer.allocate(4).putInt(id).array();

    Stat stat = zk.exists("/leader", false);  //Don't care about setting watch
    if(stat != null){
        // There's already a leader – behave as follower
        leaderID = zk.getData("/leader", false, stat);
    } else {
        try{
            String node = zk.create("/leader", leaderID, Ids.OPEN_ACL_UNSAFE,
                    CreateMode.EPHEMERAL);
            // There was no leader, I'm the leader – do something as leader
        } catch (KeeperException e) {
            if (e.code() == Code.NODEEXISTS) {
                // There's already a leader – behave as follower
                leaderID = zk.getData("/leader", false, stat);}
        }
    }
}
```

Leader Election in Java

Main

```
    public static void main (String[] args){
        try {
            new ZKClient(Integer.parseInt(args[0]));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Leader Election in Perl

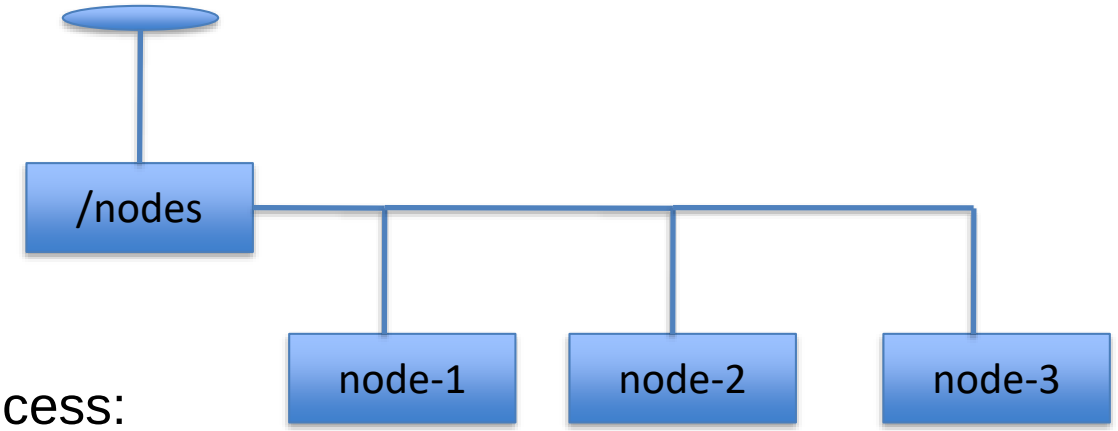
Ciências ULisboa

```
my $zkh = Net::ZooKeeper->new('localhost:7000');
my $req_path = "/app/leader";
$path = $zkh->get($req_path, 'stat'=> $stat, 'watch'=>$watch);
if (defined $path) {
    #someone else is the leader
     #parse the string path that contains the leader address
} else {
    $path = $zkh->create($req_path, "hostname:info", 'flags' => ZOO_EPHEMERAL,
                'acl' => ZOO_OPEN_ACL_UNSAFE) ;
    if (defined $path) {
       #we are the leader, continue leading
    } else {
        $path = $zkh->get($req_path, 'stat'=> $stat, 'watch'=>$watch);
        #someone  else is the leader
        # parse the string path that contains the leader address
    }
}
```

Leader Election in Python

```
handle = zookeeper.init("localhost:2181", my_connection_watcher, 10000, 0)
(data, stat) = zookeeper.get(handle, “/app/leader”, True);
if (stat == None)
        path = zookeeper.create(handle, “/app/leader”,  hostname:info,
[ZOO_OPEN_ACL_UNSAFE], zookeeper.EPHEMERAL)
        if (path == None)
            (data, stat) = zookeeper.get(handle, “/app/leader”, True)
            #someone  else is the leader
            # parse the string path that contains the leader address
        else
            # we are the leader continue leading
else
    #someone else is the leader
    #parse the string path that contains the leader address
```

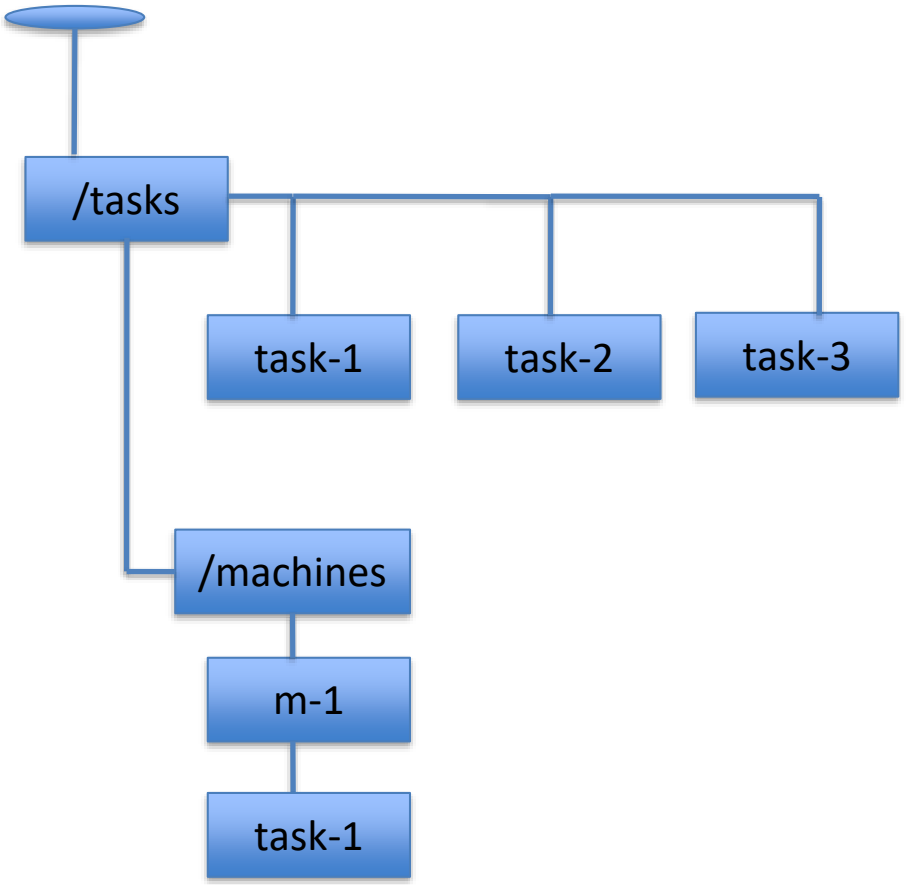
Cluster Management



- Monitoring process:
  1. Watch on /nodes
  2. On watch trigger do
     - getChildren(/nodes, true)
  3. Track which nodes have gone away
- Each Node:
  1. Create /nodes/node-${i} as ephemeral nodes
  2. Keep updating /nodes/node-${i} periodically for node status changes (status updates could be load/iostat/cpu/others)

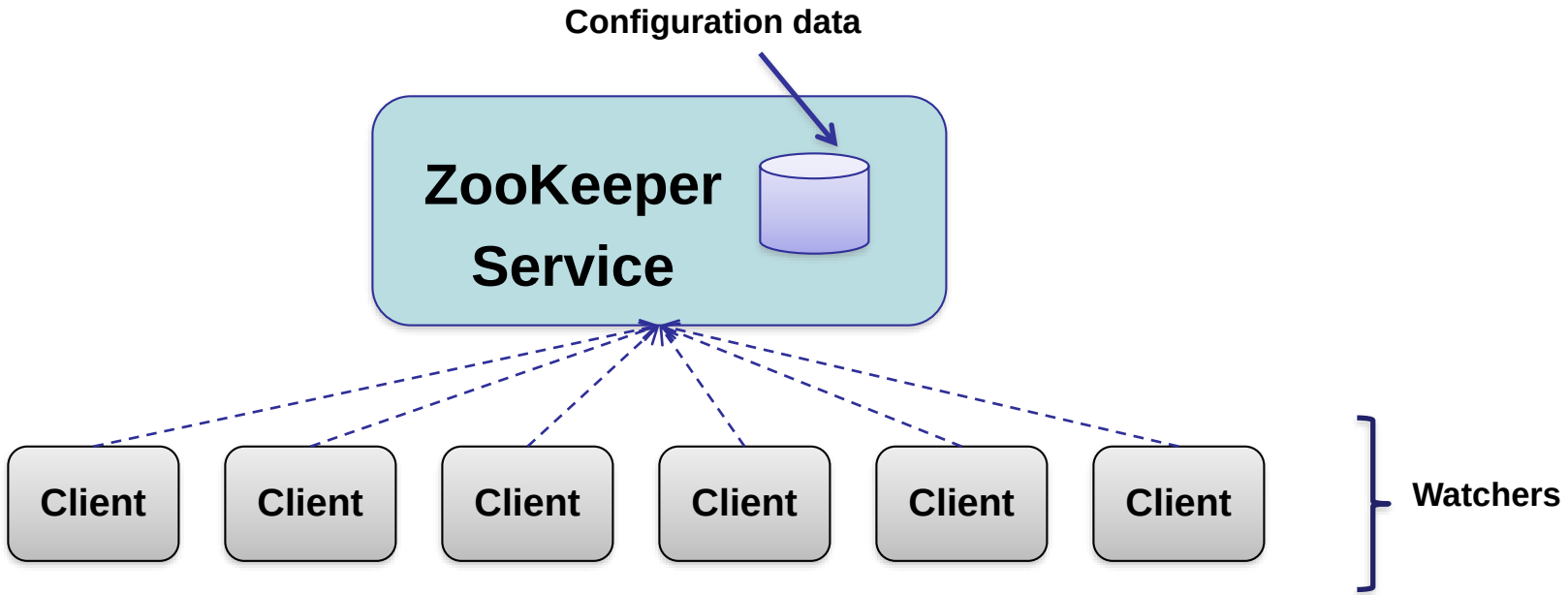
Work Queues

- Monitoring process:
  1. Watch /tasks for published tasks
  2. Pick tasks on watch trigger from /tasks
  3. Assign it to a machine specific queue by creating create(/machines/m-${i}/task-${j})
  4. Watch for deletion of tasks (maps to task completion)
- Machine process:
  1. Machines watch for /(/machines/m-${i}) for any creation of tasks
  2. After executing task-${i} delete task-${i} from /tasks and /m-${i}



Configuration management

- Configuration data stored in Znodes
- Clients set watchers
- Clients are notified when the configuration is updated
- Clients reset the watch, read the latest configuration and take appropriate action



Simple Lock without herd effect

Ciências ULisboa

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)
```

Read/Write Lock

Ciências ULisboa

Write Lock

```
1  n = create(l + "/write-", 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 event
6  goto 2
```

Read Lock

```
1  n = create(l + "/read-", EPHEMERAL|SEQUENTIAL)
2  C = getChildren(l, false)
3  if no write znodes lower than n in C, exit
4  p = write znode in C ordered just before n
5  if exists(p, true) wait for event
6  goto 3
```

Setup ZooKeeper in the labs (Linux)

Ciências ULisboa

- Check ZooKeeper info on:

  http://zookeeper.apache.org/

- Look for ZooKeeper (v3.4.8) in the lab PCs
- If not available, download newest version (v3.4.9) from:

  http://mirrors.fe.up.pt/pub/apache/zookeeper/

- Install in your working area
- Follow getting started instructions for standalone server on:

  http://zookeeper.apache.org/doc/r3.4.9/