分布式协同管理——Zookeeper

贝毅君

beiyj@zju.edu.cn

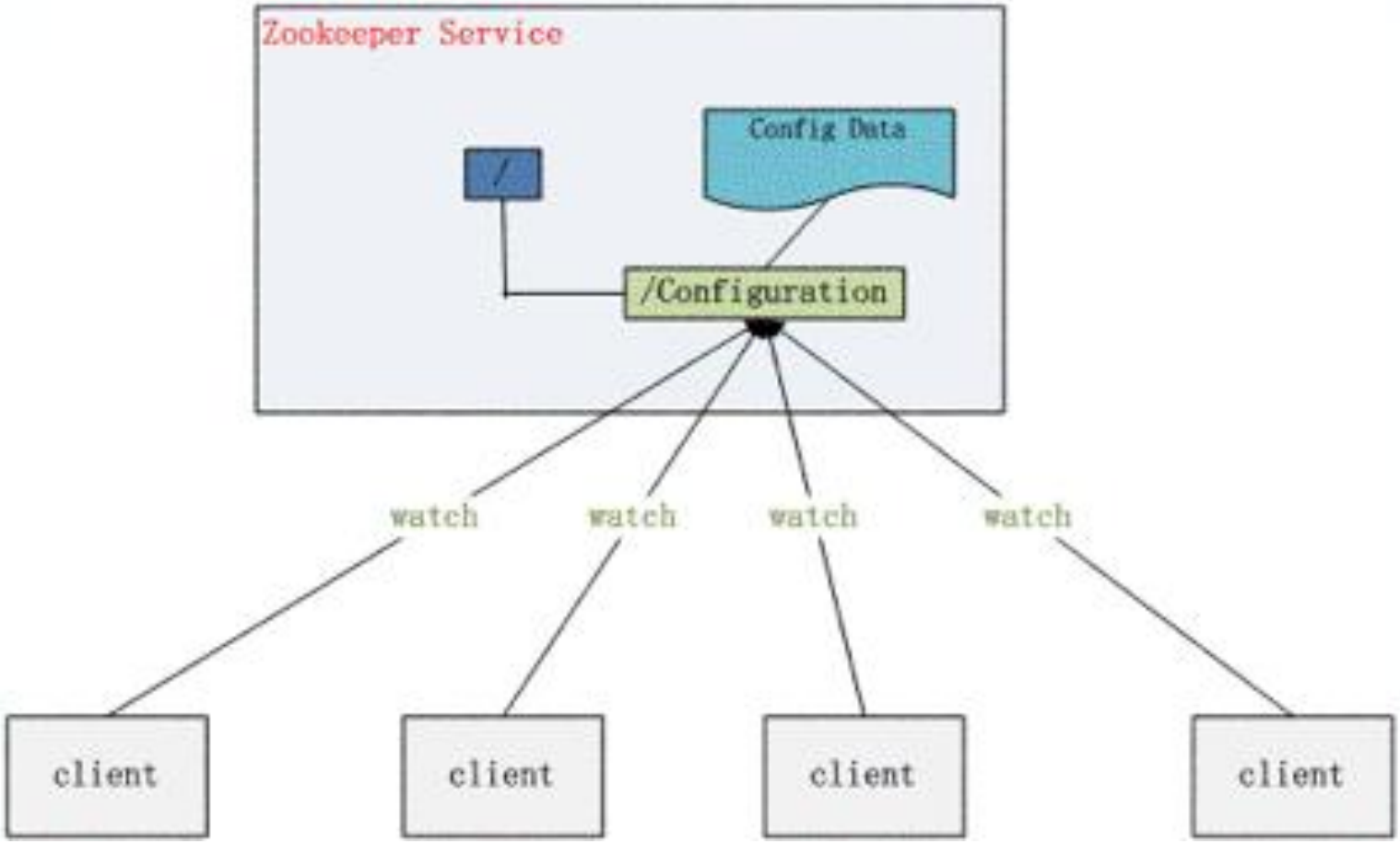
应用场景——统一命名服务

- Zookeeper的树形目标结构可以做为命名规则
- 分布式应用中，通常需要有一套完整的命名规则，既能够产生唯一的名称又便于人识别和记住，通常情况下用树形的名称结构是一个理想的选择，树形的名称结构是一个有层次的目录结构。
- Name Service 已经是 Zookeeper 内置的功能，你只要调用 Zookeeper 的 API 就能实现。如调用 create 接口就可以很容易创建一个目录节点。

应用场景——配置管理

- 配置的管理在分布式应用环境中很常见，例如同一个应用系统需要多台PC Server 运行，但是它们运行的应用系统的某些配置项是相同的。如果要修改这些相同的配置项，那么就必须同时修改每台运行这个应用系统的PC Server，这样非常麻烦而且容易出错。
- 将配置信息保存在 Zookeeper 的某个目录节点中，然后将所有需要修改的应用机器监控配置信息的状态，一旦配置信息发生变化，每台应用机器就会收到Zookeeper 的通知，然后从 Zookeeper 获取新的配置信息应用到系统中。

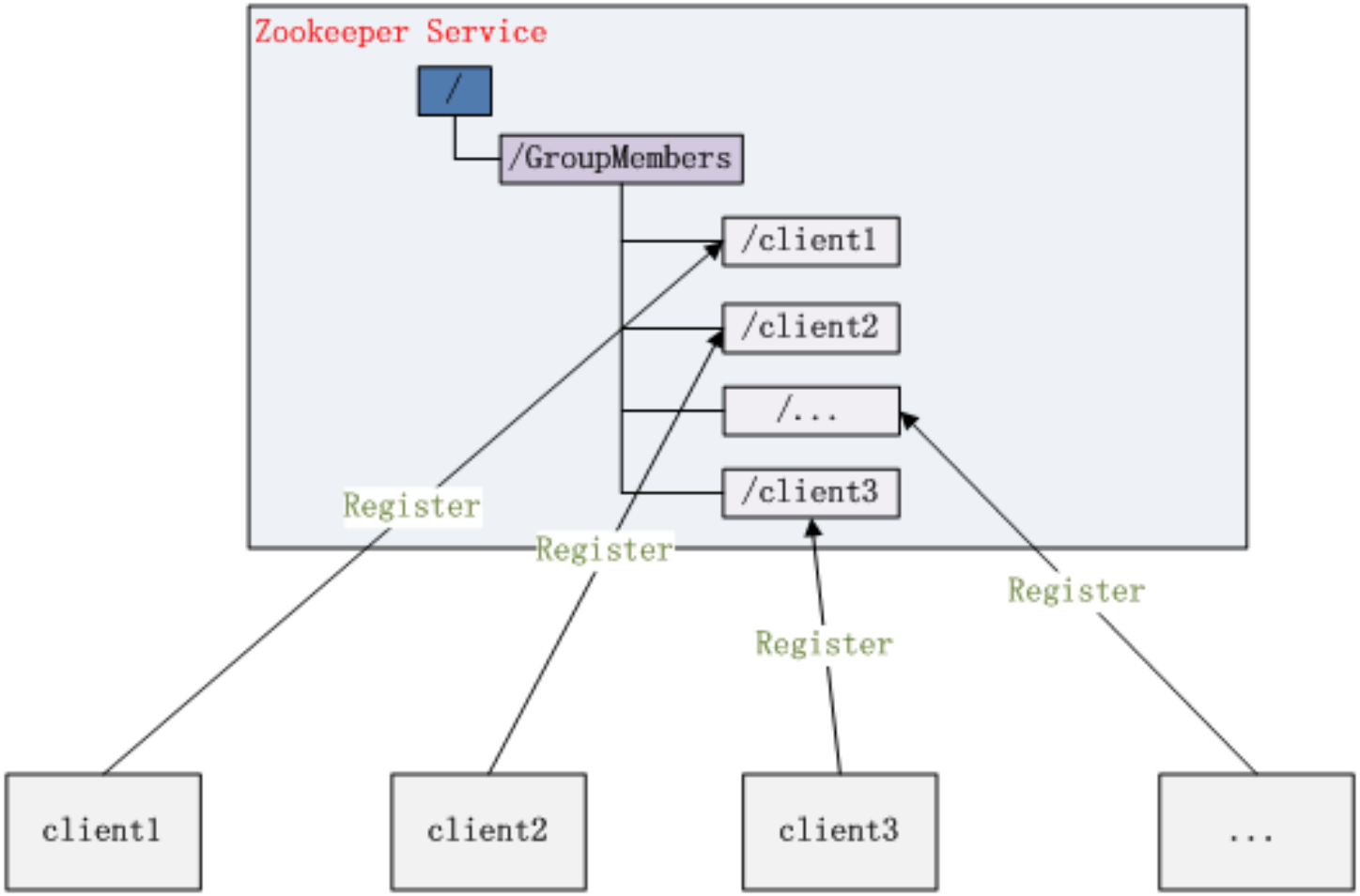
应用场景——配置管理(2)


Zookeeper Service
Config Data
/
/Configuration
watch
watch
watch
watch
client
client
client
client

应用场景——集群管理(1)

- Zookeeper 不仅能够维护当前的集群中机器的服务状态，而且能够选出一个“Master”，让这个总管来管理集群，这就是 Zookeeper 的另一个功能 Leader Election。
- 规定编号最小的为master，对SERVERS节点做监控的时候，得到服务器列表，只要所有集群机器逻辑认为最小编号节点为master，那么master就被选出。
- 这个master宕机的时候，相应的znode会消失，然后新的服务器列表就被推送到客户端，然后每个节点逻辑认为最小编号节点为master，这样就做到动态master选举。

应用场景——集群管理(2)


Zookeeper Service
/
/GroupMembers
/client1
/client2
/...
/client3
Register
Register
Register
Register
client1
client2
client3
...

应用场景——应用Leader选举

- 所有节点创建具有相同路径 /app/leader_election/guid_ 的顺序、临时节点。
- ZooKeeper集合将附加10位序列号到路径，创建的znode将是 /app/leader_election/guid_0000000001，/app/leader_election/guid_0000000002等。
- 对于给定的实例，在znode中创建最小数字的节点成为leader，而所有其他节点是follower。
- 每个follower节点监视下一个具有最小数字的znode。例如，创建znode/app/leader_election/guid_0000000008的节点将监视znode/app/leader_election/guid_0000000007，

应用场景——应用Leader选举

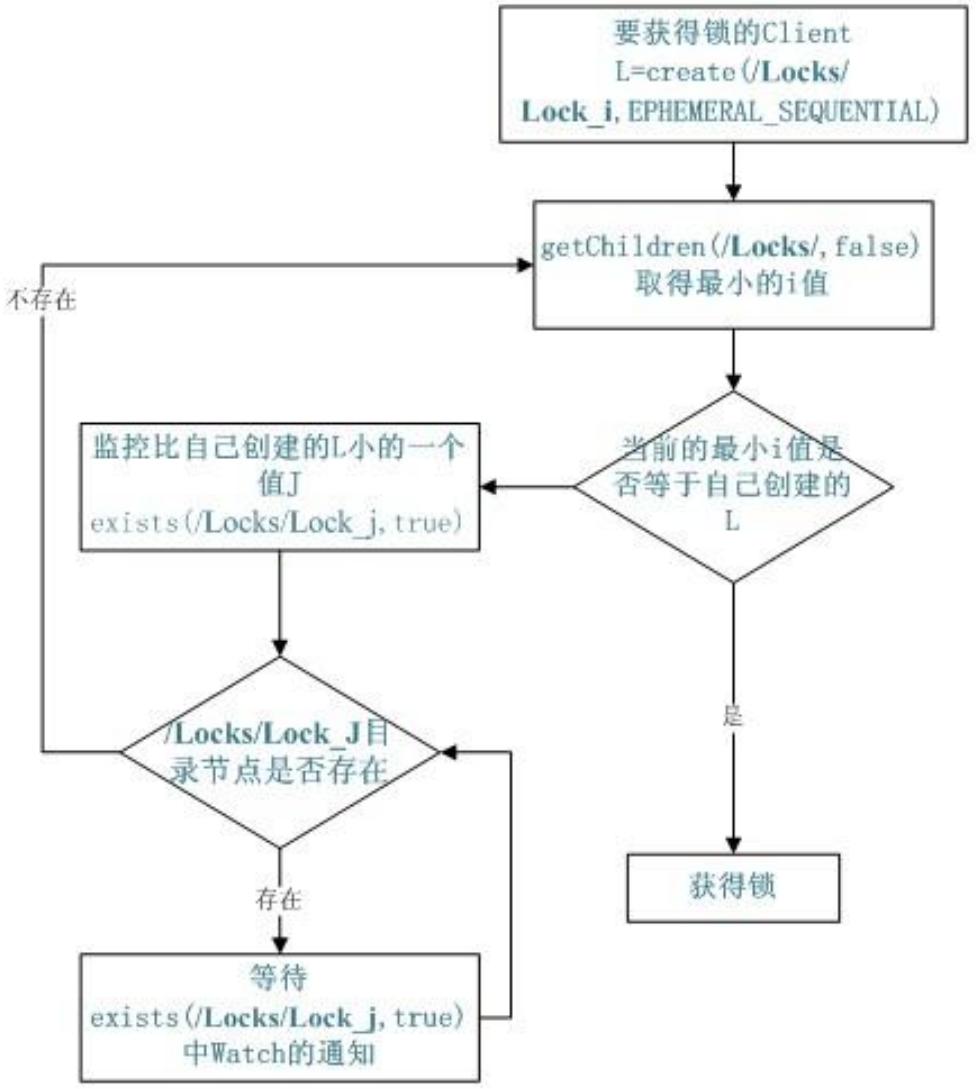
- 如果leader关闭，则其相应的znode/app/leader_electionN会被删除。
- 下一个在线follower节点将通过监视器获得关于leader移除的通知。
- 下一个在线follower节点将检查是否存在其他具有最小数字的znode。如果没有，那么它将承担leader的角色。否则，它找到的创建具有最小数字的znode的节点将作为leader。
- 类似地，所有其他follower节点选举创建具有最小数字的znode的节点作为leader。

应用场景——共享锁(1)

- 共享锁在同一个进程中很容易实现，但是在跨进程或者在不同 Server 之间就不好实现了。
- Zookeeper 却很容易实现这个功能，实现方式也是需要获得锁的 Server 创建一个 EPHEMERAL_SEQUENTIAL 目录节点，然后调用 getChildren方法获取当前的目录节点列表中最小的目录节点是不是就是自己创建的目录节点.
  - 如果正是自己创建的，那么它就获得了这个锁.
  - 如果不是那么它就调用 exists(String path, boolean watch) 方法并监控 Zookeeper 上目录节点列表的变化，一直到自己创建的节点是列表中最小编号的目录节点，从而获得锁.
  - 释放锁很简单，只要删除它自己所创建的目录节点。

应用场景——共享锁(2)

- Zookeeper实现Locks的流程图

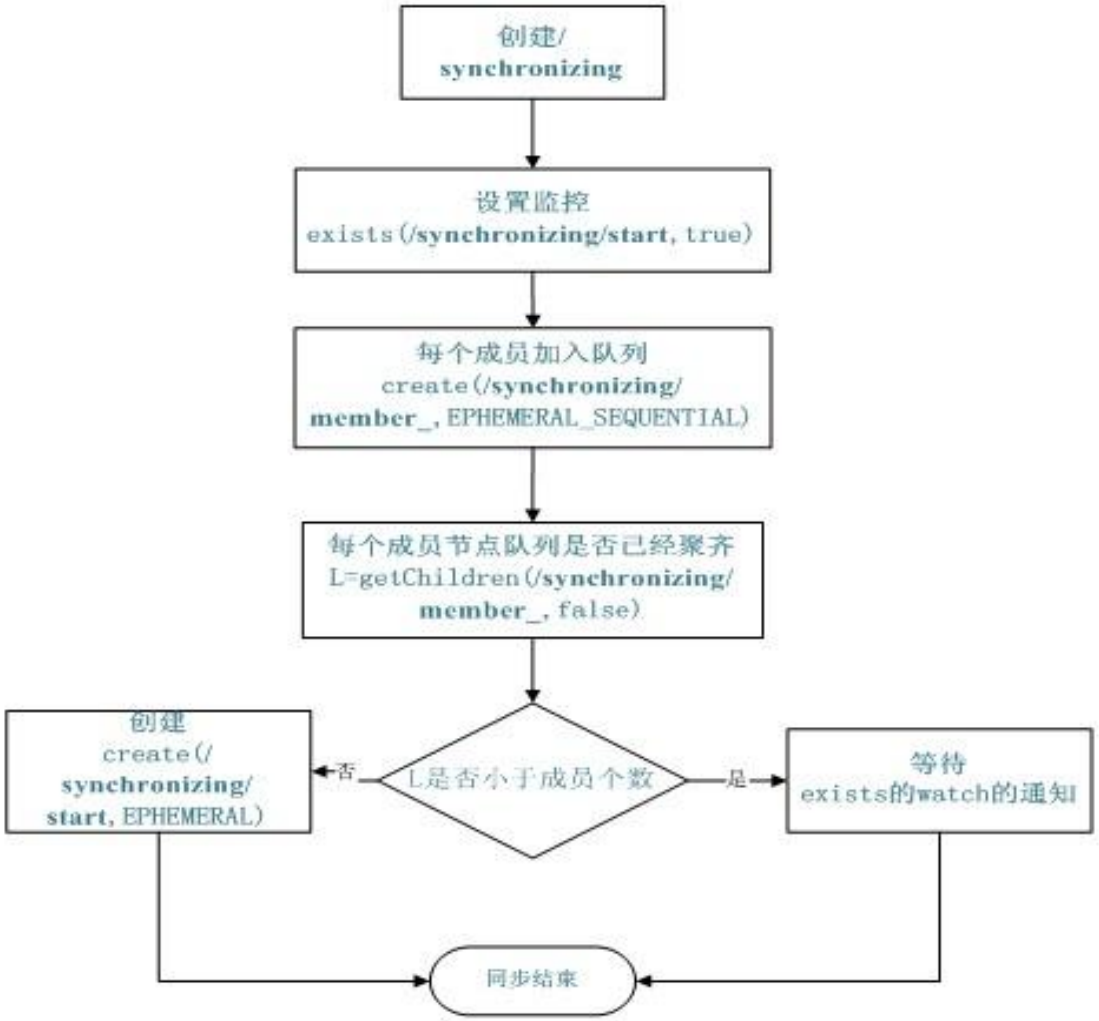


应用场景——队列管理(1)

- Zookeeper 可以处理两种类型的队列：
  - 当一个队列的成员都聚齐时，这个队列才可用，否则一直等待所有成员到达，这种是同步队列；
  - 队列按照 FIFO 方式进行入队和出队操作，例如实现生产者和消费者模型

应用场景——队列管理(2)

- 队列管理流程图



谢谢！