# ZooKeeper简单应用

目录

[ZooKeeper简单应用 1](#_Toc505688814)

[ZooKeeper简介 1](#_Toc505688815)

[什么是Zookeeper 1](#_Toc505688816)

[Zookeeper和CAP的关系 1](#_Toc505688817)

[ZooKeeper应用场景 2](#_Toc505688818)

[分布式锁 2](#_Toc505688819)

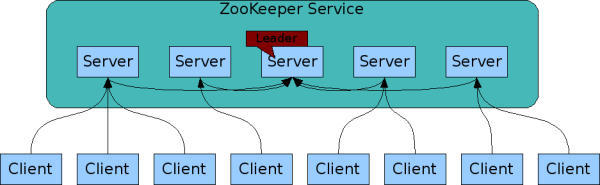
[分布式队列 2](#_Toc505688820)

## ZooKeeper简介

### 什么是Zookeeper

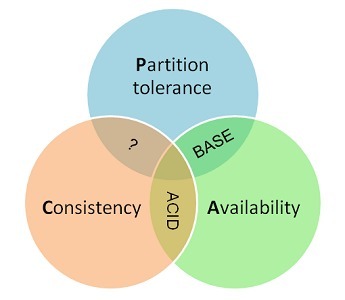
ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services。

这大概描述了Zookeeper主要是一个分布式服务协调框架，实现同步服务，配置维护和命名服务等分布式应用。是一个高性能的分布式数据一致性解决方案。



### Zookeeper和CAP[[1]](#footnote-1)的关系

作为一个分布式系统，分区容错性是一个必须要考虑的关键点。一个分布式系统一旦丧失了分区容错性，也就表示放弃了扩展性。因为在分布式系统中，网络故障是经常出现的，一旦出现在这种问题就会导致整个系统不可用是绝对不能容忍的。所以，大部分分布式系统都会在保证分区容错性的前提下在一致性和可用性之间做权衡。



ZooKeeper是个CP（一致性+分区容错性）的，即任何时刻对ZooKeeper的访问请求能得到一致的数据结果，同时系统对网络分割具备容错性；但是它不能保证每次服务请求的可用性。也就是在极端环境下，ZooKeeper可能会丢弃一些请求，消费者程序需要重新请求才能获得结果。

ZooKeeper是分布式协调服务，它的职责是保证数据在其管辖下的所有服务之间保持同步、一致；所以就不难理解为什么ZooKeeper被设计成CP而不是AP特性的了。而且， 作为ZooKeeper的核心实现算法Zab，就是解决了分布式系统下数据如何在多个服务之间保持同步问题的。

## ZooKeeper应用场景

配置中心

命名服务(Naming Service)

分布式锁

分布式队列

### 分布式锁

### 分布式队列

备注：

1、CAP一个分布式系统里面，节点组成的网络本来应该是连通的。然而可能因为一些故障，使得有些节点之间不连通了，整个网络就分成了几块区域。数据就散布在了这些不连通的区域中。这就叫分区。

当你一个数据项只在一个节点中保存，那么分区出现后，和这个节点不连通的部分就访问不到这个数据了。这时分区就是无法容忍的。

提高分区容忍性的办法就是一个数据项复制到多个节点上，那么出现分区之后，这一数据项就可能分布到各个区里。容忍性就提高了。

然而，要把数据复制到多个节点，就会带来一致性的问题，就是多个节点上面的数据可能是不一致的。要保证一致，每次写操作就都要等待全部节点写成功，而这等待又会带来可用性的问题。

总的来说就是，数据存在的节点越多，分区容忍性越高，但要复制更新的数据就越多，一致性就越难保证。为了保证一致性，更新所有节点数据所需要的时间就越长，可用性就会降低。

2、**Zookeeper工作原理**

Zab协议

Zookeeper的核心是广播，这个机制保证了各个Server之间的同步。实现这个机制的协议叫做Zab协议。

Zab（ZooKeeper Atomic Broadcast）原子消息广播协议作为数据一致性的核心算法，Zab协议是专为Zookeeper设计的支持崩溃恢复原子消息广播算法。

Zab协议核心如下：

所有的事务请求必须一个全局唯一的服务器（Leader）来协调处理，集群其余的服务器称为follower服务器。Leader服务器负责将一个客户端请求转化为事务提议（Proposal），并将该proposal分发给集群所有的follower服务器。之后Leader服务器需要等待所有的follower服务器的反馈，一旦超过了半数的follower服务器进行了正确反馈后，那么Leader服务器就会再次向所有的follower服务器分发commit消息，要求其将前一个proposal进行提交。

Zab模式

Zab协议包括两种基本的模式：崩溃恢复和消息广播。

当整个服务框架启动过程中或Leader服务器出现网络中断、崩溃退出与重启等异常情况时，Zab协议就会进入恢复模式并选举产生新的Leader服务器。

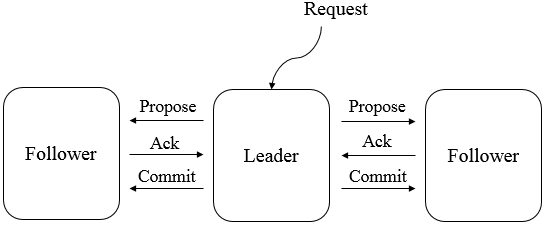
当选举产生了新的Leader服务器，同时集群中已经有过半的机器与该Leader服务器完成了状态同步之后，Zab协议就会退出恢复模式，状态同步是指数据同步，用来保证集群在过半的机器能够和Leader服务器的数据状态保持一致。

当集群中已经有过半的Follower服务器完成了和Leader服务器的状态同步，那么整个服务框架就可以进入消息广播模式。

当一台同样遵守Zab协议的服务器启动后加入到集群中，如果此时集群中已经存在一个Leader服务器在负责进行消息广播，那么加入的服务器就会自觉地进入数据恢复模式：找到Leader所在的服务器，并与其进行数据同步，然后一起参与到消息广播流程中去。

Zookeeper只允许唯一的一个Leader服务器来进行事务请求的处理，Leader服务器在接收到客户端的事务请求后，会生成对应的事务提议并发起一轮广播协议，而如果集群中的其他机器收到客户端的事务请求后，那么这些非Leader服务器会首先将这个事务请求转发给Leader服务器。

消息广播



Zab协议的消息广播过程使用是一个原子广播协议，类似一个2PC提交过程。具体的：

ZooKeeper使用单一主进程Leader用于处理客户端所有事务请求，并采用Zab的原子广播协议，将服务器数据状态变更以事务Proposal的形式广播Follower上，因此能很好的处理客户端的大量并发请求。

另一方面，由于事务间可能存在着依赖关系，Zab协议保证Leader广播的变更序列被顺序的处理，有些状态的变更必须依赖于比它早生成的那些状态变更。

最后，考虑到主进程Leader在任何时候可能崩溃或者异常退出， 因此Zab协议还要Leader进程崩溃的时候可以重新选出Leader并且保证数据的完整性；Follower收到Proposal后，写到磁盘，返回Ack。Leader收到大多数ACK后，广播Commit消息，自己也提交该消息。Follower收到Commit之后，提交该消息。

Zab协议简化了2PC事务提交：

去除中断逻辑移除，follower要么ack，要么抛弃Leader。

Leader不需要所有的Follower都响应成功，只要一个多数派Ack即可。

崩溃恢复

上面我们讲了Zab协议在正常情况下的消息广播过程，那么一旦Leader服务器出现崩溃或者与过半的follower服务器失去联系，就进入崩溃恢复模式。

恢复模式需要重新选举出一个新的Leader，让所有的Server都恢复到一个正确的状态。

1. C代表一致性，A代表可用性（在一定时间内，用户的请求都会得到应答），P代表分区容错。 [↑](#footnote-ref-1)