第 3 章 Zookeeper 内部原理

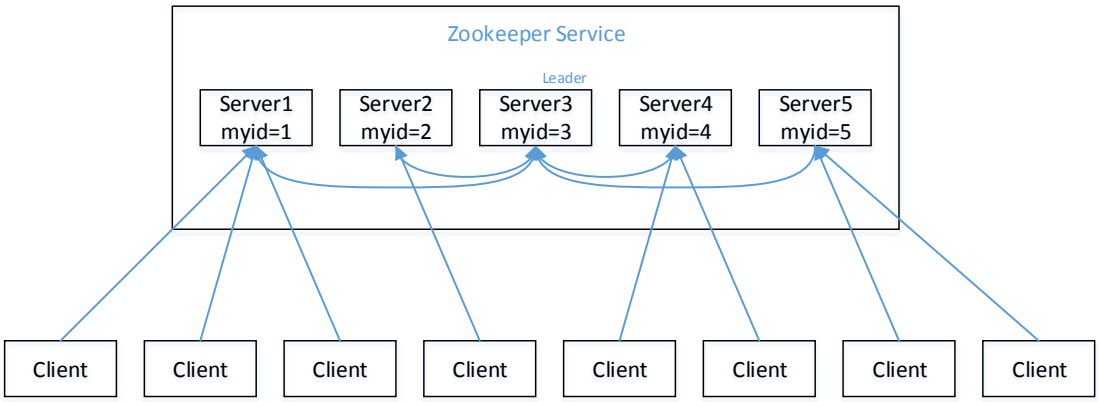
3.1 选举机制

1）半数机制（Paxos 协议）：集群中半数以上机器存活，集群可用。所以 zookeeper 适合装在奇数台机器上。

2）Zookeeper 虽然在配置文件中并没有指定 master 和 slave。但是，zookeeper 工作时，是有一个节点为 leader，其他则为 follower，Leader 是通过内部的选举机制临时产生的

3）以一个简单的例子来说明整个选举的过程。

假设有五台服务器组成的 zookeeper 集群，它们的 id 从 1-5，同时它们都是最新启动的，也就是没有历史数据，在存放数据量这一点上，都是一样的。假设这些服务器依序启动，来看看会发生什么。



（1）服务器 1 启动，此时只有它一台服务器启动了，它发出去的报没有任何响应，所以它的选举状态一直是 LOOKING 状态。

（2）服务器 2 启动，它与最开始启动的服务器 1 进行通信，互相交换自己的选举结果，由于两者都没有历史数据，所以 id 值较大的服务器 2 胜出，但是由于没有达到超过半数以上的服务器都同意选举它(这个例子中的半数以上是 3)，所以服务器 1、2 还是继续保持 LOOKING 状态。

（3）服务器 3 启动，根据前面的理论分析，服务器 3 成为服务器 1、2、3 中的老大，而与上面不同的是，此时有三台服务器选举了它，所以它成为了这次选举的 leader。

（4）服务器 4 启动，根据前面的分析，理论上服务器 4 应该是服务器 1、2、3、4 中最大的，但是由于前面已经有半数以上的服务器选举了服务器 3，所以它只能接收当小弟的命了。

（5）服务器 5 启动，同 4 一样当小弟。

3.2 节点类型

1）Znode 有两种类型：

短暂（ephemeral）：客户端和服务器端断开连接后，创建的节点自己删除

持久（persistent）：客户端和服务器端断开连接后，创建的节点不删除

2）Znode 有四种形式的目录节点（默认是 persistent ）

（1）持久化目录节点（PERSISTENT）

客户端与 zookeeper 断开连接后，该节点依旧存在

（2）持久化顺序编号目录节点（PERSISTENT_SEQUENTIAL）

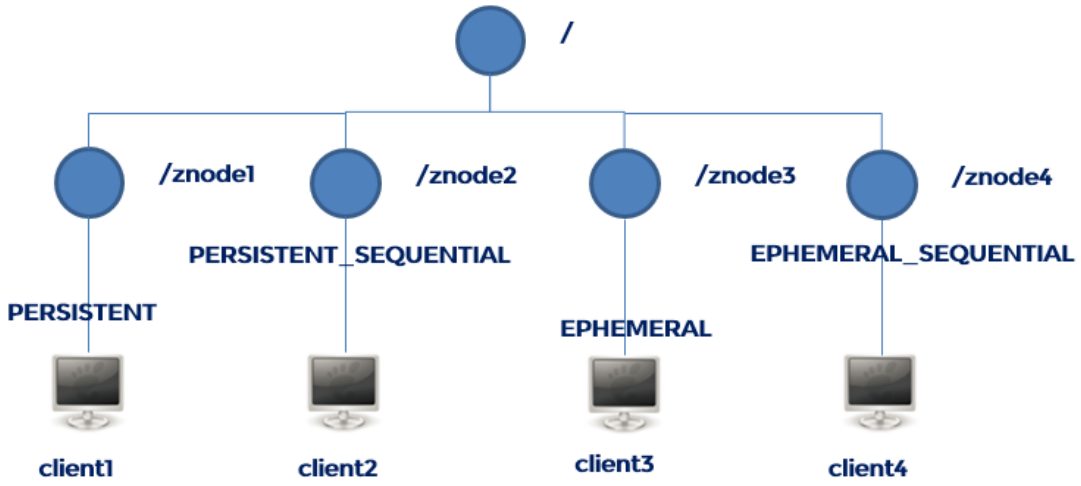
客户端与 zookeeper 断开连接后，该节点依旧存在，只是 Zookeeper 给该节点名称进行顺序编号

（3）临时目录节点（EPHEMERAL）

客户端与 zookeeper 断开连接后，该节点被删除

（4）临时顺序编号目录节点（EPHEMERAL_SEQUENTIAL）

客户端与 zookeeper 断开连接后，该节点被删除，只是 Zookeeper 给该节点名称进行顺序编号



3）创建 znode 时设置顺序标识，znode 名称后会附加一个值，顺序号是一个单调递增的计数器，由父节点维护

4）在分布式系统中，顺序号可以被用于为所有的事件进行全局排序，这样客户端可以通过顺序号推断事件的顺序

3.3 stat 结构体

1）czxid- 引起这个 znode 创建的 zxid，创建节点的事务的 zxid

每次修改 ZooKeeper 状态都会收到一个 zxid 形式的时间戳，也就是 ZooKeeper 事务 ID。

事务 ID 是 ZooKeeper 中所有修改总的次序。每个修改都有唯一的 zxid，如果 zxid1 小于 zxid2，那么 zxid1 在 zxid2 之前发生。

2）ctime - znode 被创建的毫秒数(从 1970 年开始)

3）mzxid - znode 最后更新的 zxid

4）mtime - znode 最后修改的毫秒数(从 1970 年开始)

5）pZxid-znode 最后更新的子节点 zxid

6）cversion - znode 子节点变化号，znode 子节点修改次数

7）dataversion - znode 数据变化号

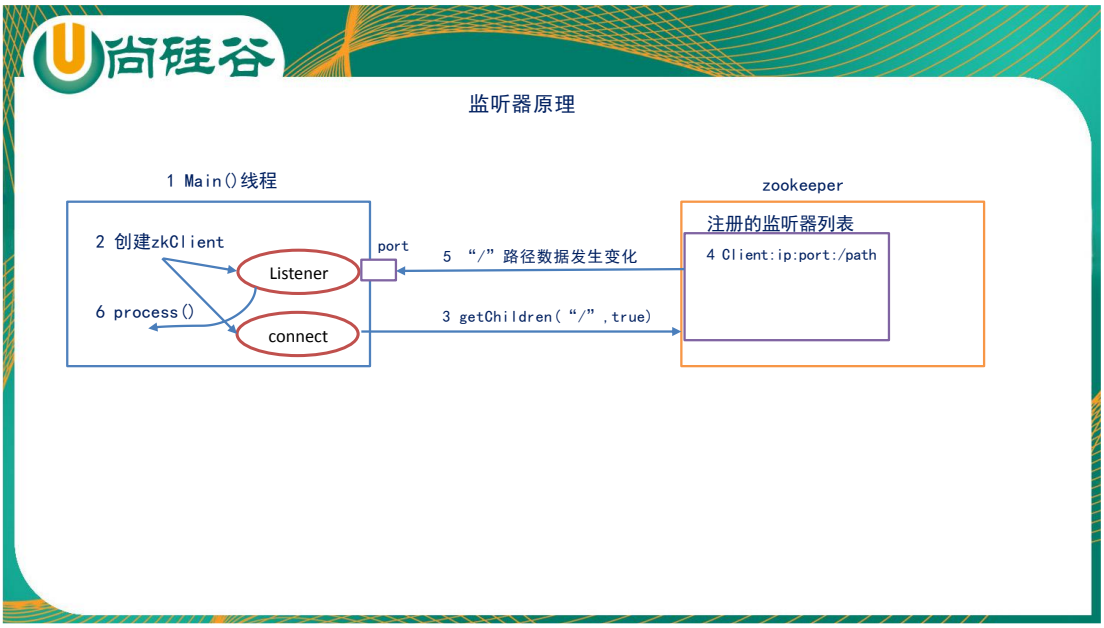
8）aclVersion - znode 访问控制列表的变化号

9）ephemeralOwner- 如果是临时节点，这个是 znode 拥有者的 session id。如果不是临时节点则是 0。

10）dataLength- znode 的数据长度

11）numChildren - znode 子节点数量

3.4 监听器原理



1）监听原理详解：

1）首先要有一个 main()线程

2）在 main 线程中创建 Zookeeper 客户端，这时就会创建两个线程，一个负责网络连接通信（connet），一个负责监听（listener）。

3）通过 connect 线程将注册的监听事件发送给 Zookeeper。

4）在 Zookeeper 的注册监听器列表中将注册的监听事件添加到列表中。

5）Zookeeper 监听到有数据或路径变化，就会将这个消息发送给 listener 线程。

6）listener 线程内部调用了 process（）方法。

2）常见的监听

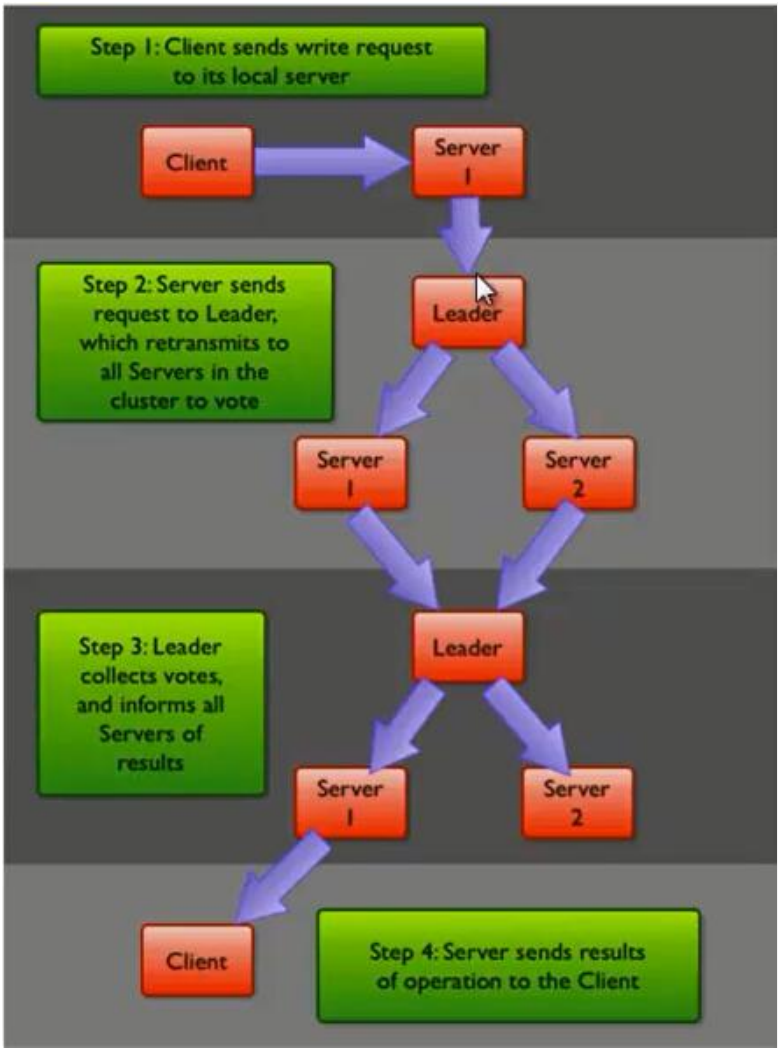
（1）监听节点数据的变化：

get path [watch]

（2）监听子节点增减的变化

ls path [watch]

3.5 写数据流程



ZooKeeper 的写数据流程主要分为以下几步：

1）比如 Client 向 ZooKeeper 的 Server1 上写数据，发送一个写请求。

2）如果 Server1 不是 Leader，那么 Server1 会把接受到的请求进一步转发给 Leader，因为每个 ZooKeeper 的 Server 里面有一个是 Leader。这个 Leader 会将写请求广播给各个 Server，比如 Server1 和 Server2， 各个 Server 写成功后就会通知 Leader。

3）当 Leader 收到大多数 Server 数据写成功了，那么就说明数据写成功了。如果这里三个节点的话，只要有两个节点数据写成功了，那么就认为数据写成功了。写成功之后，Leader 会告诉 Server1 数据写成功了。

4）Server1 会进一步通知 Client 数据写成功了，这时就认为整个写操作成功。ZooKeeper 整个写数据流程就是这样的。