# Zookeeper知识点

## Zookeeper是什么框架

分布式的、开源的分布式应用程序协调服务，原本是Hadoop、HBase的一个重要组件。它为分布式应用提供一致性服务的软件，包括：配置维护、域名服务、分布式同步、组服务等。

## 应用场景

Zookeeper的功能很强大，应用场景很多，结合我实际工作中使用Dubbo框架的情况，Zookeeper主要是做注册中心用。基于Dubbo框架开发的提供者、消费者都向Zookeeper注册自己的URL，消费者还能拿到并订阅提供者的注册URL，以便在后续程序的执行中去调用提供者。而提供者发生了变动，也会通过Zookeeper向订阅的消费者发送通知。

## Paxos算法& Zookeeper使用协议

Paxos算法是分布式选举算法，Zookeeper使用的 ZAB协议（Zookeeper原子广播），二者有相同的地方，比如都有一个Leader，用来协调N个Follower的运行；Leader要等待超半数的Follower做出正确反馈之后才进行提案；二者都有一个值来代表Leader的周期。

不同的地方在于：

ZAB用来构建高可用的分布式数据主备系统（Zookeeper），Paxos是用来构建分布式一致性状态机系统。

Paxos算法、ZAB协议要想讲清楚可不是一时半会的事儿，自1990年莱斯利·兰伯特提出Paxos算法以来，因为晦涩难懂并没有受到重视。后续几年，兰伯特通过好几篇论文对其进行更进一步地解释，也直到06年谷歌发表了三篇论文，选择Paxos作为chubby cell的一致性算法，Paxos才真正流行起来。

对于普通开发者来说，尤其是学习使用Zookeeper的开发者明确一点就好：分布式Zookeeper选举Leader服务器的算法与Paxos有很深的关系。

## 选举算法和流程

ZooKeeper采用Fast Leader Election算法（ 可以理解为Paxos的一个简化版，一个变种）。

ZooKeeper服务器共有4个状态：

LOOKING：寻找Leader状态。

LEADING：领导者。

FOLLOWING：跟随者。

OBSERVING：观察者。

大致的过程就是：

1. 服务启动之后，先选择自己作为Leader，广播通知其他服务器，并等待接收其他服务器的发送的选票。
2. 不停地接收其他服务器发送的选票，根据ZXID和服务器的ID更新自己的选票。
3. 直到某服务器的选票过半数，设置为Leader，其他都是Follower。
4. 当Leader已经确定的情况下，新加入的服务器也只能是Follower。

ZXID：ZooKeeper状态的每一次改变（每一次请求）, 都对应着一个递增的Transaction id, 该id称为zxid. 由于zxid的递增性质, 如果zxid1小于zxid2, 那么zxid1肯定先于zxid2发生. 创建任意节点, 或者更新任意节点的数据, 或者删除任意节点, 都会导致Zookeeper状态发生改变, 从而导致zxid的值增加。

初始状态，大家的ZXID都一致，那就就选举ID（myid里面配置的数字）最大的。

假设有5台服务器：

|  |
| --- |
| 1)服务器1启动，此时只有它一台服务器启动了，它发出去的消息没有任何响应，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(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先选自己为Leader，服务器1、2分别也选它为Leader（ID最大），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，所以它只能做Follower。  5)服务器5启动，同4一样，当Follower。 |

## Zookeeper有哪几种节点类型

持久：创建之后一直存在，除非有删除操作，创建节点的客户端会话失效也不影响此节点。

持久顺序：跟持久一样，就是父节点在创建下一级子节点的时候，记录每个子节点创建的先后顺序，会给每个子节点名加上一个数字后缀。

临时：创建客户端会话失效（注意是会话失效，不是连接断了），节点也就没了。不能建子节点。

临时顺序：不用解释了吧。

## Zookeeper对节点的watch监听通知是永久的吗？

不是。官方声明：一个Watch事件是一个一次性的触发器，当被设置了Watch的数据发生了改变的时候，则服务器将这个改变发送给设置了Watch的客户端，以便通知它们。

为什么不是永久的，举个例子，如果服务端变动频繁，而监听的客户端很多情况下，每次变动都要通知到所有的客户端，这太消耗性能了。

一般是客户端执行getData(“/节点A”,true)，如果节点A发生了变更或删除，客户端会得到它的watch事件，但是在之后节点A又发生了变更，而客户端又没有设置watch事件，就不再给客户端发送。

在实际应用中，很多情况下，我们的客户端不需要知道服务端的每一次变动，我只要最新的数据即可。

## 部署方式？集群中的机器角色都有哪些？集群最少要几台机器

单机，集群。Leader、Follower。集群最低3（2N+1）台，保证奇数，主要是为了选举算法。

## 集群如果有3台机器，挂掉一台集群还能工作吗？挂掉两台呢？

一个原则：过半存活即可用。

## 集群支持动态添加机器吗？

其实就是水平扩容了，Zookeeper在这方面不太好。两种方式：

全部重启：关闭所有Zookeeper服务，修改配置之后启动。不影响之前客户端的会话。

逐个重启：顾名思义。这是比较常用的方式。

## Zookeeper是如何监控集群服务健康的

参看《Zookeeper有哪几种节点类型》，我们知道主要有两大类：永久和临时。

所有的服务，在某个永久节点（假设为pnode）下创建临时子节点（假设为cnode1、cnode2……cnodeN），并与ZK保持心跳，以免Session失效。

其他服务通过方法getChildren(父节点地址,watch=true)对父节点下的子节点设置watch。

而一旦出现服务停止，网络中断等异常情况，心跳没有了，session也超时了，那对应的临时子节点也会被删除。

临时子节点一被删除，就等于产生了变化，触发watch，其他服务器就能得到变化后新信息。

同样的，新增服务时，子节点也是发生了变化，同样触发watch。