18-zookeeper-原理介绍

----------成都尚学堂-mr-zeng----------

zookeeper概念

zookeeper 是 Google 的 Chubby一个开源的实现，是 Hadoop 体系的分布式协调服务。帮助管理分布式架构里的服务器(为集群做一些打酱油的事情，高可用，统一配置，分布式锁...)。**ps-》zookeeper本身也是分布式架构**（他自己解决了分布式高可用问题-，解决了分布式数据存储一致性问题）。

zookeeper核心功能

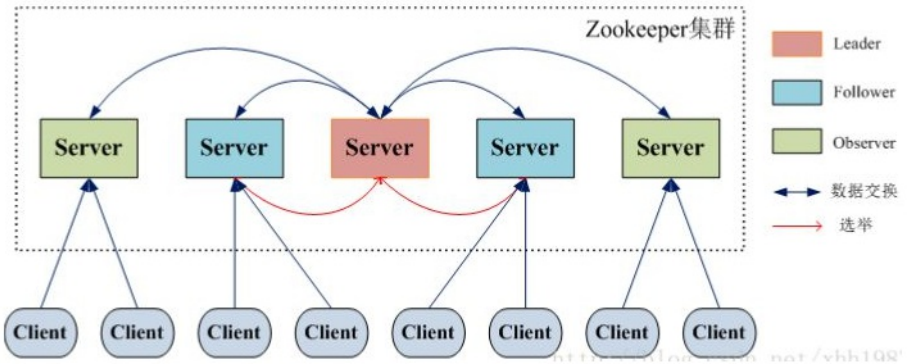
a）提供的小型的**分布式文件系统**（**内存中-访问速度快，存储量不大-也会写在硬盘中**）。

--》各服务器统一存储数据-（最终数据一致）

b）提供了对文件系统里**各数据操作的监听事件（Watcher）**。

--》观察者模式

zookeeper架构



服务端集群（三个角色：Leader，follower，observer（了解即可（主要是扩展读性能））-管理里整个分布式文件系统。

Leader：总统，整个服务器里老大。

功能：查询数据，增删该数据，选举提意见。

follower：议员。

功能：查询数据，选举提意见。

observer：记者。

功能：查询数据。

客户端

client：屁民，可以使用zookeeper服务端来管理数据。

zookeeper的应用

1）分布式系统统一配置
2）分布式系统--单点服务器-高可用
3）分布式统一命名
4）分布式锁
...

使用了zookeeper技术的框架（很多分布式服务框架都会使用zookeeper）

1）hdfs-namenode的高可用
2）yarn-resourceManger的高可用
3）dubobo 远程调用服务管理
4）solr
5）storm
...

----------Zookeeper服务端集群工作原理----------怎么达到（分布式服务里的数据一致性）----------

先说Paxos，它是一个基于消息传递的【分布式系统数据】一致性算法，Leslie Lamport在1990年提出，近几年被广泛应用于分布式计算中，Google的Chubby，Apache的Zookeeper都是基于它的理论来实现的，Paxos还被认为是到目前为止唯一的分【分布式系统数据】算法，其它的算法都是Paxos的改进或简化。有个问题要提一下，Paxos有一个前提：没有拜占庭将军问题。就是说Paxos只有在一个可信的计算环境【安全的

计算环境】中才能成立，这个环境是不会被入侵所破坏的。

关于Paxos的具体算法-》比较抽象-以下是通俗版

有一个小岛住了一批居民，岛上面所有的【法规-数据】由一些特殊的人决定，他们叫做**议员，**议员里有一个老大-->**总统。议员总数**是确定的（建议奇数），不能更改。

岛上屁民

a）提议【法规-数据-更新】请求某一个议员，所有议员和总统就会开会讨论，每个提议都有一个编号(PID)（ **）**），这个编号是一直增长的（**0开始**），不能倒退。每个**提议**都需要**超过半数**((议员总数)/2 +1)的**议员**同意才能生效。

b）【法规-数据-查询】请求某一个议员，直接从议员那里获取【法规-数据】，如果需要最新的（则需要同步（总统）获取最新数据）。

提议时-开会-议员做的事情

每个议员**只会同意大于当前编号的提议**，包括已生效的和未生效的。**如果议员收到小于等于当前编号的提议**，他会拒绝，并告知对方：你的提议已经有人提过了。这里的当前编号是每个议员在自己记事本上面记录的编号，他不断更新这个编号。整个议会不能保证所有议员记事本上的编号总是相同的。现在议会有一个目标：**保证所有的议员对于提议都能达成一致的看法（所有议员查询的数据都一致（不会出现：一个议员认为电费1元，另一个认为2元））。**

编号PID（Zxid）：防止冲突，每个议员只会同意大于当前**编号**的提议。

超过半数：防止两个冲突提议都生效。如两个同时屁民（都说电费问题（1元和2元））谁先过半就生效，另一个作废。

议员总数：建议奇数，容错性（可以死掉小半的议员）

议员总数 可以死掉的数量

5 2

6 2

7 3

8 3

模拟一次会议开始的（这里模拟电费问题-》最开始电费5角（编号0））

有一个议员发了一个提议：将电费设定为1元/度。他首先看了一下记事本，嗯，当前提议编号是0，那么我的这个提议的编号就是1，先投了自己一票，于是他给所有议员发消息：1号提议，设定电费1元/度。其他议员收到消息以后查了一下记事本，哦，当前电费-提议编号是0，这个提议可接受，于是他记录下这个提议并回复：我接受你的1号提议，同时他在记事本上记录：当前提议编号为1。**发起提议的议员收到了超过半数的回复，**立即给所有人**发通知：1号提议生效**！收到的议员会修改他的记事本，将1好提议由记录改成正式的法令，当有人问他电费为多少时，他会查看法令并告诉对方：1元/度。

冲突的解决（多个议员都同时提议）：假设总共有三个议员：周杰伦，蔡依林，志林姐姐

提议：闯红灯-罚款100元（编号0）

两个议员同时提议

1）**周杰伦**：提出"闯红灯-罚款250"，编号1 -》周杰伦自己同意 -》发给其他人

1）**蔡依林**：提出"闯红灯-罚款500"，编号1 -》蔡依林自己同意 -》发给其他人

2）**志林姐姐**

a）先收到了**周杰伦**的提议，他首先看目前自己的提议（闯红灯-罚款100元（编号0）），收到的是"闯红灯-罚款250"（编号1），编号更大则同意该提议。

b）紧接着收到了**蔡依林**的提议，他首先看目前自己的提议（闯红灯-罚款250"（编号1））），收到的是"闯红灯-罚款500"（编号1），编号一样则拒绝该提议。

于是

周杰伦的（"闯红灯-罚款250"，编号1 ）超过半数--》作为生效的法规执行。

蔡依林的（"闯红灯-罚款500"，编号1 ）没有超过半数--》没有生效。，如果蔡依林发现自己的1号没有生效那么还可以发起2号，获取也可以直接去同步使用生效的1号 。

Zookeeper集群和小岛例子映射

小岛(Island)——ZooKeeper Server 集群

议员(Senator)——ZooKeeper Server

提议(Proposal)——ZNode Change(Create/Delete/SetData...)

提议编号(PID)——Zxid(ZooKeeper Transaction Id)

正式法规——ZNode里正式生效的数据

貌似关键的概念都能一一对应上，但是等一下，Paxos岛上的议员应该是人人平等的吧，而ZK Server好像有一个Leader的概念。没错，其实Leader的概念也应该属于Paxos范畴的。如果议员人人平等，在某种情况下会由于提议的冲突而产生一个“活锁”。解决方案——在所有议员中设立一个总统，**只有总统有权发出提议**，如果议员有自己的提议，必须发给总统并由总统来提出。好，我们又多了一个角色：总统。

活锁：例如a，b，c三个议员，同时发网费（2，3，4元），自己都同意自己的，然后收到被人的（编号小）都不同意。这样就一直没有提议完成！！。
选总统---》统一的发提议不会有同时发的情况，总统按顺序发，防止冲突。

总统——ZK Server Leader
总统怎么选出来的？
总统挂掉--或---刚开始运作：只有一个议员一产生，马上就发自荐信给所有人，如果有(议员总数)/2 +1的人通过了，那么总统就产生了（选举和提议类似！！）

----------**ZK Server工作情景**----------

情况一：（屁民想要查看法规-读取数据）

屁民甲(Client)到某个议员(ZK Server)那里询问(Get)某条法令的情况(ZNode的数据)，议员毫不犹豫的拿出他的记事本(local storage)，查阅法令并告诉他结果，同时声明：我的数据不一定是最新的。你想要最新的数据？没问题，等着，等我找总统Sync一下再告诉你。

情况二：（屁民想要提交法规建议-更新数据（增，删，改））

屁民乙(Client)到某个议员(ZK Server)那里希望执行一个新的提议（电费下调到1角一度），议员让他在办公室等着，自己将提议反映给了总统，总统发出提议，询问所有议员的意见，**过半议员表示同意，于是总统发表声明**，（电费下调到1角一度）生效，让所有的议员**更新电费价格。**

情况三：（选举-总统挂了-选举期间不提供服务）

总统突然挂了，没有老大，会议不能正常工作，议员接二连三的发现联系不上总统，于是各自发表声明，推选新的总统，总统大选期间政府停业，拒绝屁民的请求。

情况四：（容错性-议员挂了）
议员挂了，会议还是可以正常工作，但是挂掉的人数不能过半---（因为选举-提议-都需要过半的人来投票！）。