1、（F93）那个zkclient的wather反复注册不太清楚，是怎么回事？

原生zk，wather注册只能监听znode最近一次操作的结果，需要反复注册才能达到不断监听znode的效果

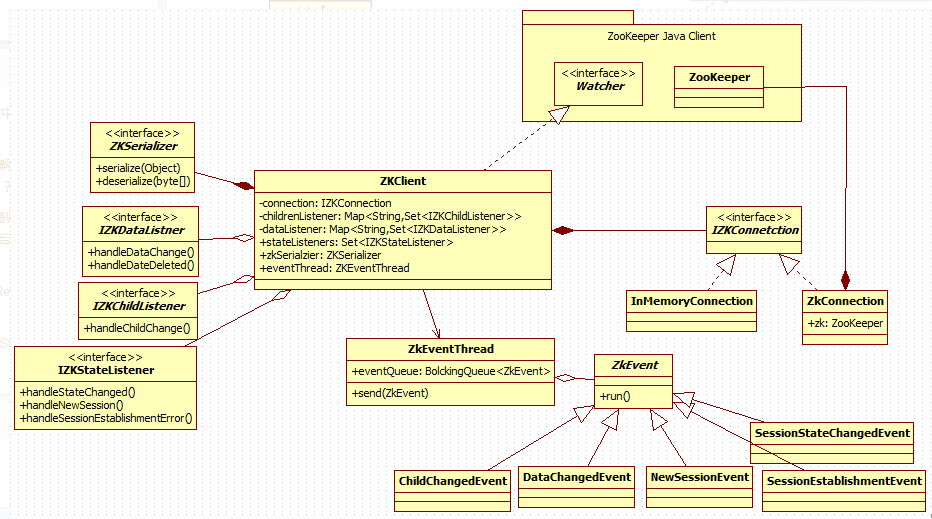
回复：原生zk，wather注册只能监听znode最近一次操作的结果，需要反复注册才能达到多次监听znode的效果。

zkclient 将wather和操作进行了分离，将事件重新定义分为了stateChanged、znodeChanged、dataChanged三种情况，用户可以注册这三种情况下的监听器（znodeChanged和dataChanged和路径有关），而不是注册Watcher。

ZKClient框架中ZooKeeper只注册了类ZkClient（实现了Watcher），由ZkClient统一处理WatchedEvent，根据WatchedEvent分发到三种情况的处理方法，处理方法在寻找到监听器后会将他send到ZkEventThread的BlockingQueue中，由ZkEventThread以线程的方式执行

zkClient将一次性watcher包装为持久watcher。后者的具体做法是简单的在watcher回调中，重新读取数据的同时再注册相同的watcher实例。

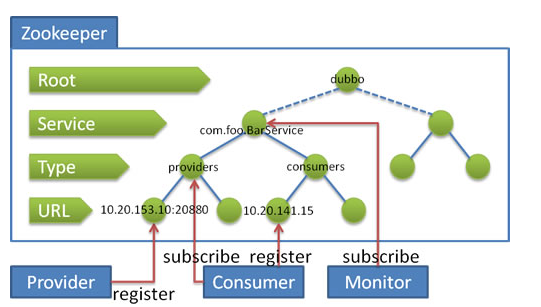
示例代码：https://blog.csdn.net/vbirdbest/article/details/82712453



2、（H36）zk 服务注册为什么是虚拟节点，用永久节点，服务关了，删掉不也行吗？

回复：zk的节点，有四种：永久、临时、永久有序、临时有序

场景一

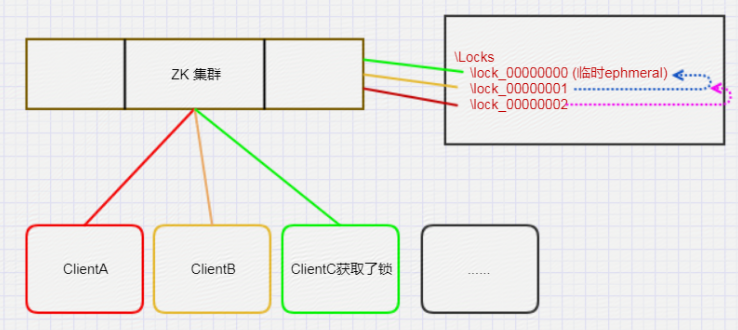


举个例子：当zk在dubbo框架中用做服务发现和注册时，会创建一个 /dubbo的永久节点，服务提供方会在/dubbo下需创建临时节点(服务名称、provider的ip、端口)，服务订阅方也会在/dubbo下需创建临时节点(consumer的ip、端口)。这样，当服务发布方和订阅方发生变化时，ZK会及时更新/dubbo下面的节点列表。

场景二 分布式锁

首先会有一个永久节点\Locks，然后每个客户端请求的时候会创建一个临时有序节点，在这时每个都是有序的，最小的节点就意味着获取了锁。

      在此图上显然ClientC获取了锁，其他的锁获取的节点不是最小的，但是他们之前会有一个链接，就是lock\_00000001在虽然没有获取锁，但是会需要监听lock\_00000000的，因为如果监听所有节点的话会浪费很多的资源。相应的大的节点都会watch比自己小的节点，当比自己节点小的节点释放之后然后就可以继续处理了。



3、（H100）如何比较consul 和 zk？

<https://zhidao.baidu.com/question/1963745821579705100.html>

4（I100）

负载均衡是谁实现呢？consumer，还是zookeeper？

答：昨天那个列子是consumer dubbo的话服务端和客户端都做了

1. （N43）

1、在用zookeeper实现分布式锁的时候，线程1加锁后，为什么不通过watcher机制通知其他线程锁已经释放，这样就不会造成阻塞了，

2、在用zookeeper实现服务注册时候，为什么不像配置中心那样使用本地缓存服务地址，然后用watcher进行监听和同步

回复：

1. 异步 很难保证顺序，和是否成功（网络）
2. 是可以的，毕竟是demo，dubbo没有注册中心也能调用原理也是缓存的。

6、（N25）

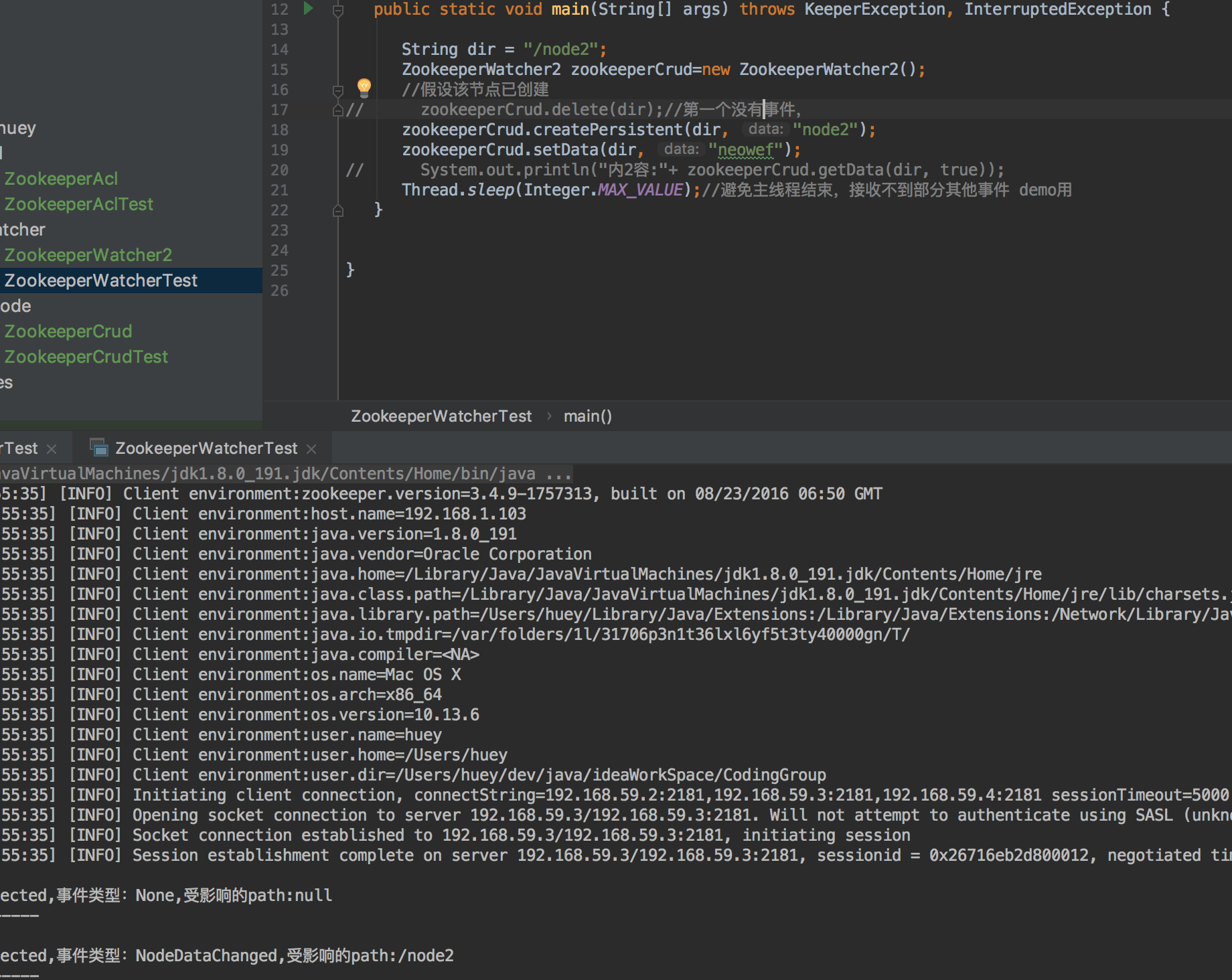
A：代码中的重试机制是？为什么要用到这个？

网络有不确定性，如果连不上应该有一定的连接次数

B:如何解决分布式事务性能太低的问题?

这个和zk没有太多关系呀。  
  
  
C：zookeeper能承受多少连接？几百个客户端订阅会不会造成性能问题？

这个没有具体测试、不过几百个问题不大的，zk读能力不错，写能力差点

1. 

（J66）

答：第一个事件不触发的问题

我在process方法中设置的是exits 因为第一行代码是delete 删除完成这个节点 exits就找不到这个节点了 自然就不能触发了，需要在delete之前增加wathcer机制（节点存在的情况）然后进行删除就ok了