第三节-zookeeper客户端与服务端交互流程源码解析

鲁班学院-周瑜

曾参与大型电商平台、互联网金融产品等多家互联网公司的开发，曾就职于大众点评，任项目经理等职位，参与并主导千万级并发电商网站与系统架构搭建

> 学海无涯，我们一起勉力前行
>
> 课程讲师：**周瑜老师QQ：3413298904**
>
> 往期课程资料：**木兰老师QQ: 2746251334**
>
> VIP课程咨询：**安其拉老师QQ: 3164703201**

源码构建

1. 安装ant
2. github:https://github.com/boomblog/zookeeper
3. 切到brach-3.4.13-vip分支
4. 在项目根目录下运行ant eclipse
5. 用IDEA导入eclipse工程
6. 运行QuorumPeerMain，program arguments=conf/zoo.cfg，并且复制conf/log4j.properties到src/java/main路径下

NIO中的几个概念

Channel（通道），Buffer（缓冲区），Selector（选择器

Channel

可以将NIO 中的Channel同传统IO中的Stream来类比，但是要注意，传统IO中，Stream是单向的，比如InputStream只能进行读取操作，OutputStream只能进行写操作。而Channel是双向的，既可用来进行读操作，又可用来进行写操作

Buffer

在NIO中所有数据的读和写都离不开Buffer，读取的数据只能放在Buffer中，写入数据也是先写入到Buffer中

Selector

将Channel和Selector配合使用，必须将channel注册到selector上。通过SelectableChannel.register()方法来实现。

```
channel.configureBlocking(false);
SelectionKey key = channel.register(selector, Selectionkey.OP_READ);
```

四种不同类型的事件：

1. Connect
2. Accept
3. Read
4. Write

通道触发了一个事件意思是该事件已经就绪。所以，某个channel成功连接到另一个服务器称为“连接就绪”。一个server socket channel准备好接收新进入的连接称为“接收就绪”。一个有数据可读的通道可以说是“读就绪”。等待写数据的通道可以说是“写就绪”。

ZookeeperMain-客户端启动类

1. ZookeeperMain接收客户端命令
2. ZookeeperMain将客户端命令转化为Request

Zookeeper

Zookeeper调用ClientCnxn.submitRequest方法将Request包装成Packet并添加到outgoingQueue队列中

ClientCnxn

SendThread

1. 连接到服务端并且进行重试
2. 发送ping
3. doIO

发送数据

1. 从outgoingQueue中取出数据并发送给服务端
2. 将需要等待结果的Packet加入到pendingQueue中

读取数据

1. 接收watcher事件通知，并且把通知加入到waitingEvents队列中去
2. 将pendingQueue中的Packet读取出来并且使用服务端返回的结果进行装配
3. 如果是同步请求则唤醒线程
4. 如果是异步请求则将Packet加入到waitingEvents队列中

EventThread

1. 从waitingEvents队列中获取数据
2. 如果是watcher事件通知，出发绑定的watcher逻辑
3. 如果异步请求，则调用对应的异步回调函数

QuorumPeerMain-服务端启动类

1. 解析配置
2. 根据配置进行单机或集群模式的启动（判断条件就是配置文件中的servers的数量）

QuorumPeerConfig-配置类

ZooKeeperServerMain-单机模式启动类

1. 初始化ZooKeeperServer
2. 初始化FileTxnSnapLog
3. 初始化NIOServerCnxnFactory
4. 启动NIOServerCnxnFactory
5. 启动ZooKeeperServer

ServerConfig

单机模式下的配置类， 配置属性比集群模式下的要少一点

ZooKeeperServer

zk服务器

1. 初始化ZKDatabase
2. 初始化DataTree
3. 从SnapShot从还原DataTree
4. 开启session检查器
5. 设置请求处理器RequestProcessor

FileTxnSnapLog

事务日志和快照持久化工具类

1. TxnLog-事务日志
2. SnapShot-快照日志

ServerCnxnFactory

服务器上下文工厂类，负责创建服务器上下文，默认为NIOServerCnxnFactory，可配

1. 开启ServerSocketChannel

NIOServerCnxnFactory

一个线程类

1. 启动ZooKeeperThread，这个类开启的其实就是NIOServerCnxnFactory自己
2. 接收到客户端的连接事件，就初始化出来一个NIOServerCnxn
3. 接收数据或写出数据，NIOServerCnxn.doIO

读数据

1. 读取的是ConnectRequest，服务端新建一个session，并生成一个新的sessionId，并生成一个Request调用submitRequest方法
2. 读取的正常操作请求，也会生成一个Request调用submitRequest方法

提交请求submitRequest方法

RequestProcessor

单机模式下:

1. firstProcessor = PrepRequestProcessor(线程)
2. PrepRequestProcessor.next = SyncRequestProcessor(线程)
3. SyncRequestProcessor.next = FinalRequestProcessor

PrepRequestProcessor-接受客户端请求生成txn事务以及节点修改记录

1. processRequest方法将Request添加到submittedRequests队列中
2. 线程不停的从submittedRequests获取请求

3. 根据请求的类型进入不同的处理，我们以create为例
4. 获取父节点信息
5. 校验ACL
6. 临时节点与顺序节点逻辑
7. 生成txn事务
8. 生成父节点修改记录
9. 生成新增节点修改记录
10. 将修改记录加入到outstandingChanges队列中
11. 调用nextProcessor.processRequest(request);

SyncRequestProcessor

1. processRequest方法将Request添加到queuedRequests队列中
2. 负责从queuedRequests队列中获取Request
3. 负责将txn同步到磁盘，并且进行快照
4. 如果同步完成了就会调用nextProcessor.processRequest(si);

FinalRequestProcessor

1. 从outstandingChanges队列中获取Request
2. 更新DataTree
3. 触发watcher
4. 构造Response
5. 通过NIOServerCnxn中的sendResponse把Response转化成ByteBuffer返回给客户端

只读模式

当服务器集群一半以上的节点挂掉之后，不能进行写操作，但是可以做读操作，这时可以开启只读模式

1. 服务器开启只读模式：-Dreadonlymode.enabled=true
2. 客户端调用开启只读模式：zkServer.sh start -r 或者 new ZooKeeper(..., canBeReadOnly=true)
3. 以上两步都开启了才是支持只读模式