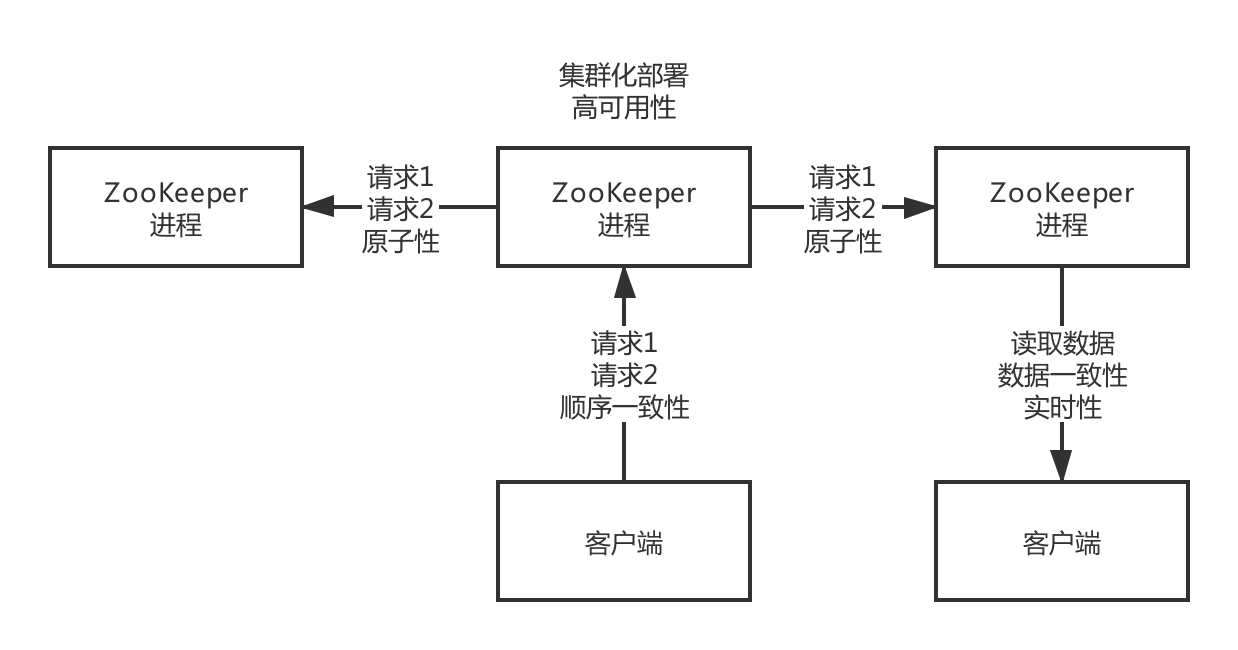
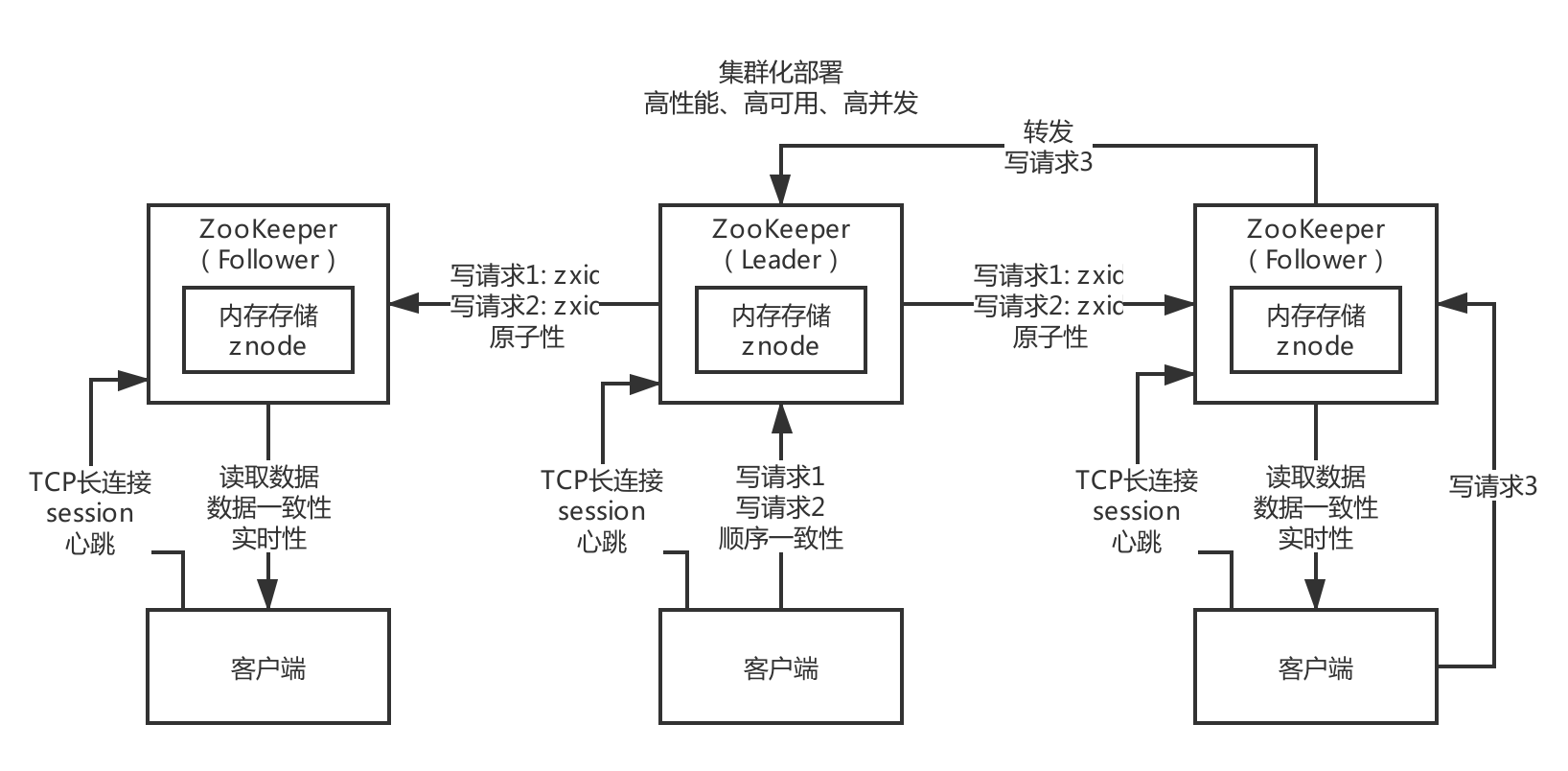
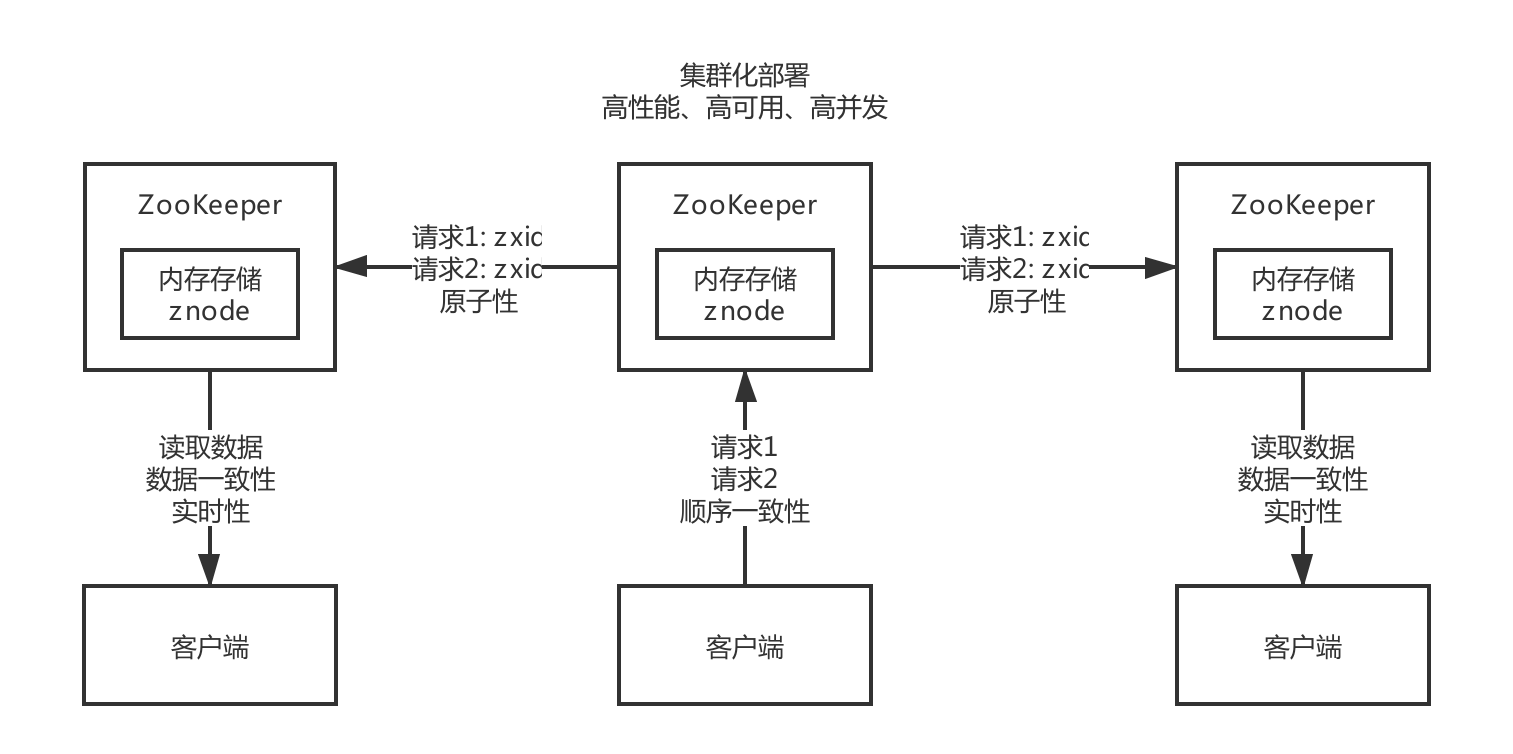
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/004\_ZooKeeper为了满足分布式系统的需求要有哪些特性？/01\_ZooKeeper架构原理.png**



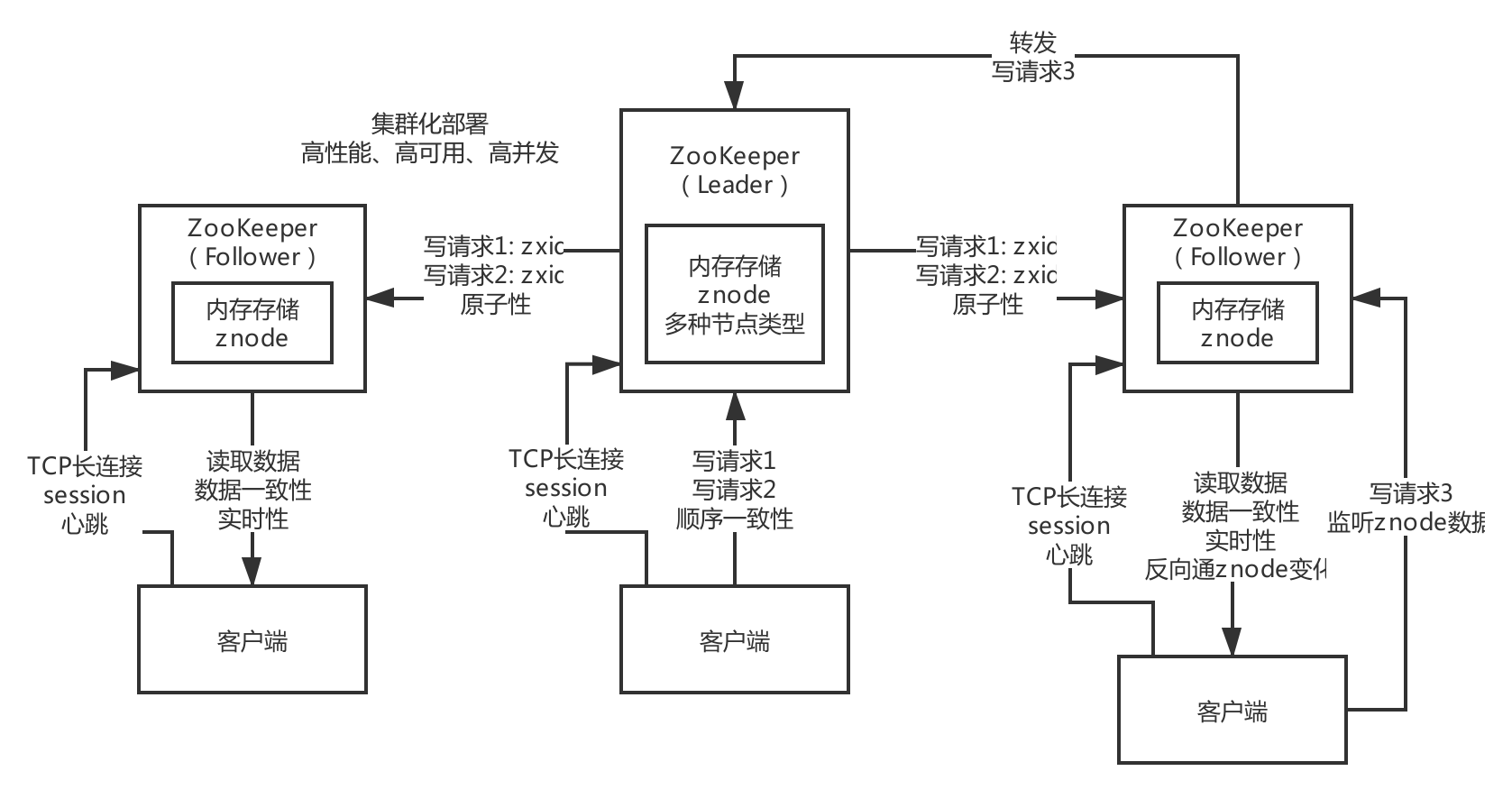
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/007\_客户端与ZooKeeper之间的长连接和会话是什么？/01\_ZooKeeper架构原理.png**



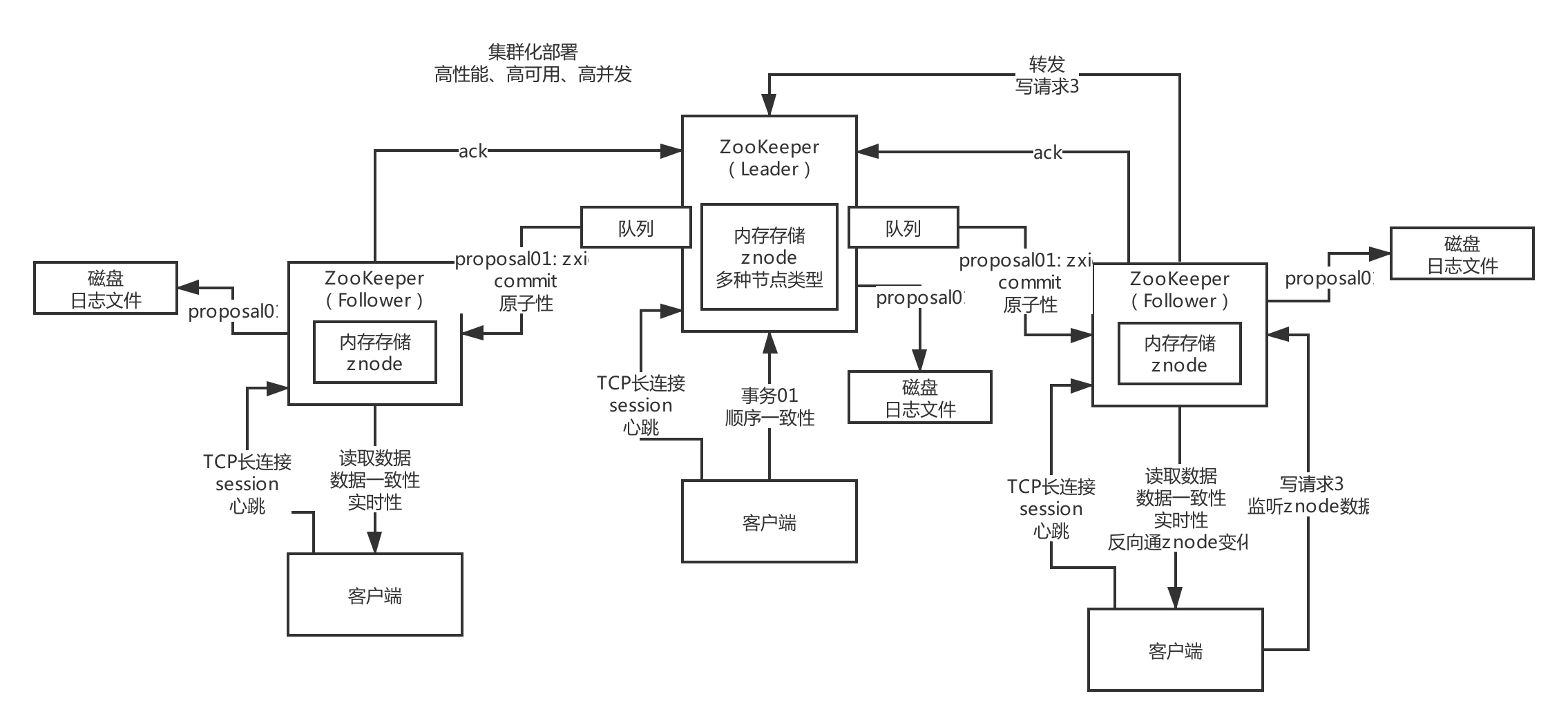
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/005\_为了满足分布式系统的需求，ZooKeeper的架构设计有哪些特点？/01\_ZooKeeper架构原理.png**



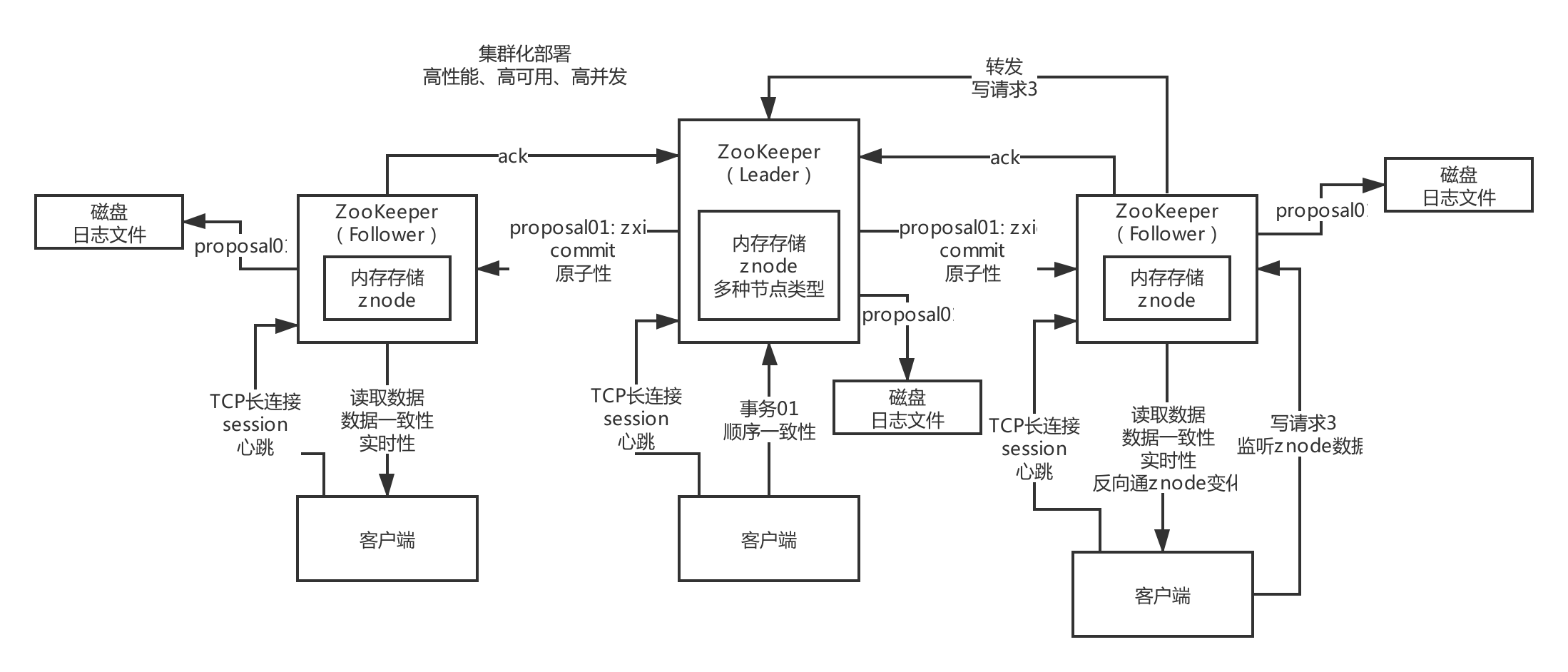
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/009\_ZooKeeper最核心的一个机制：Watcher监听回调/01\_ZooKeeper架构原理.png**



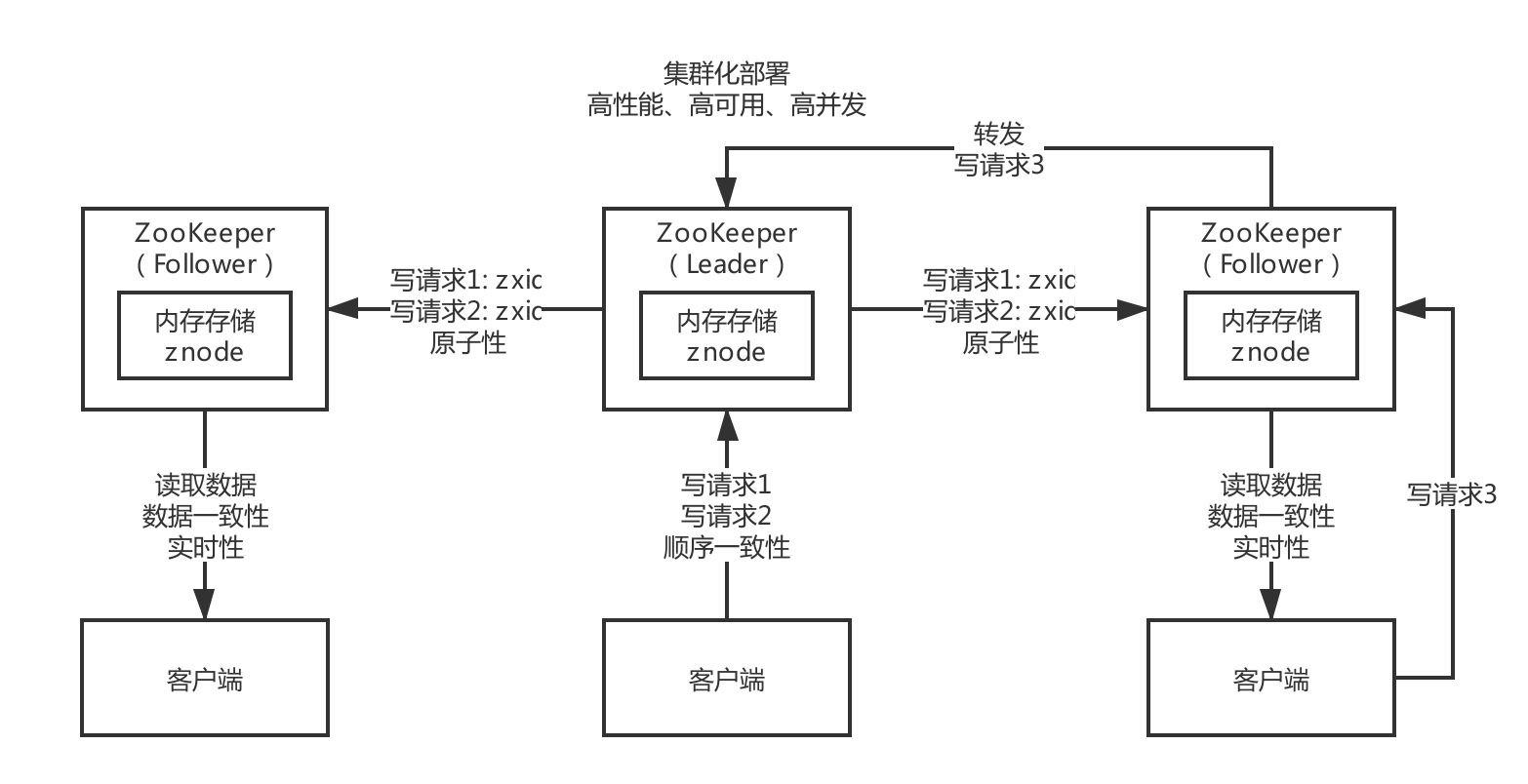
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/013\_采用了2PC两阶段提交思想的ZAB消息广播流程/01\_ZooKeeper架构原理.png**



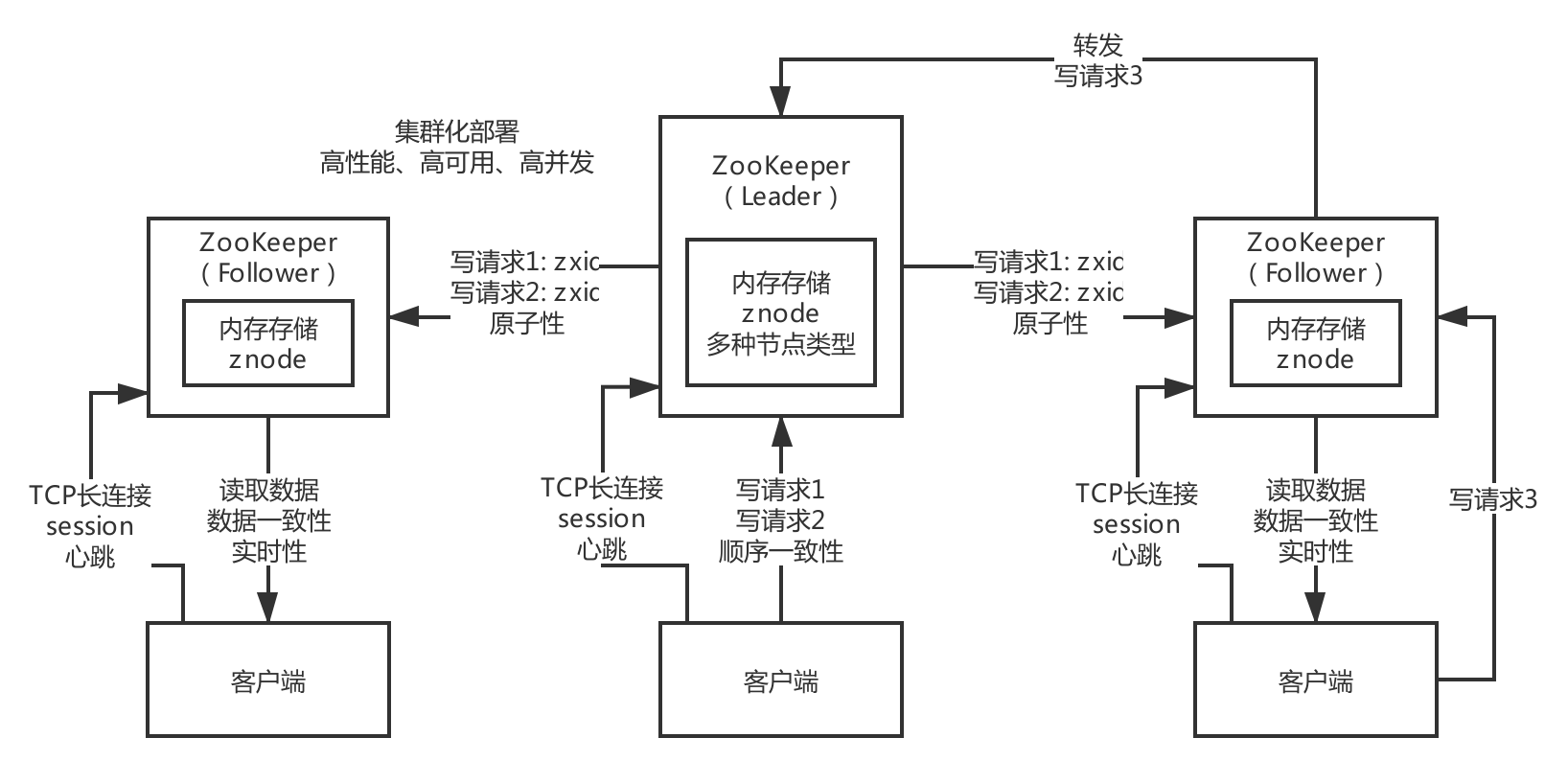
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/011\_ZAB的核心思想介绍：主从同步机制和崩溃恢复机制/01\_ZooKeeper架构原理.png**



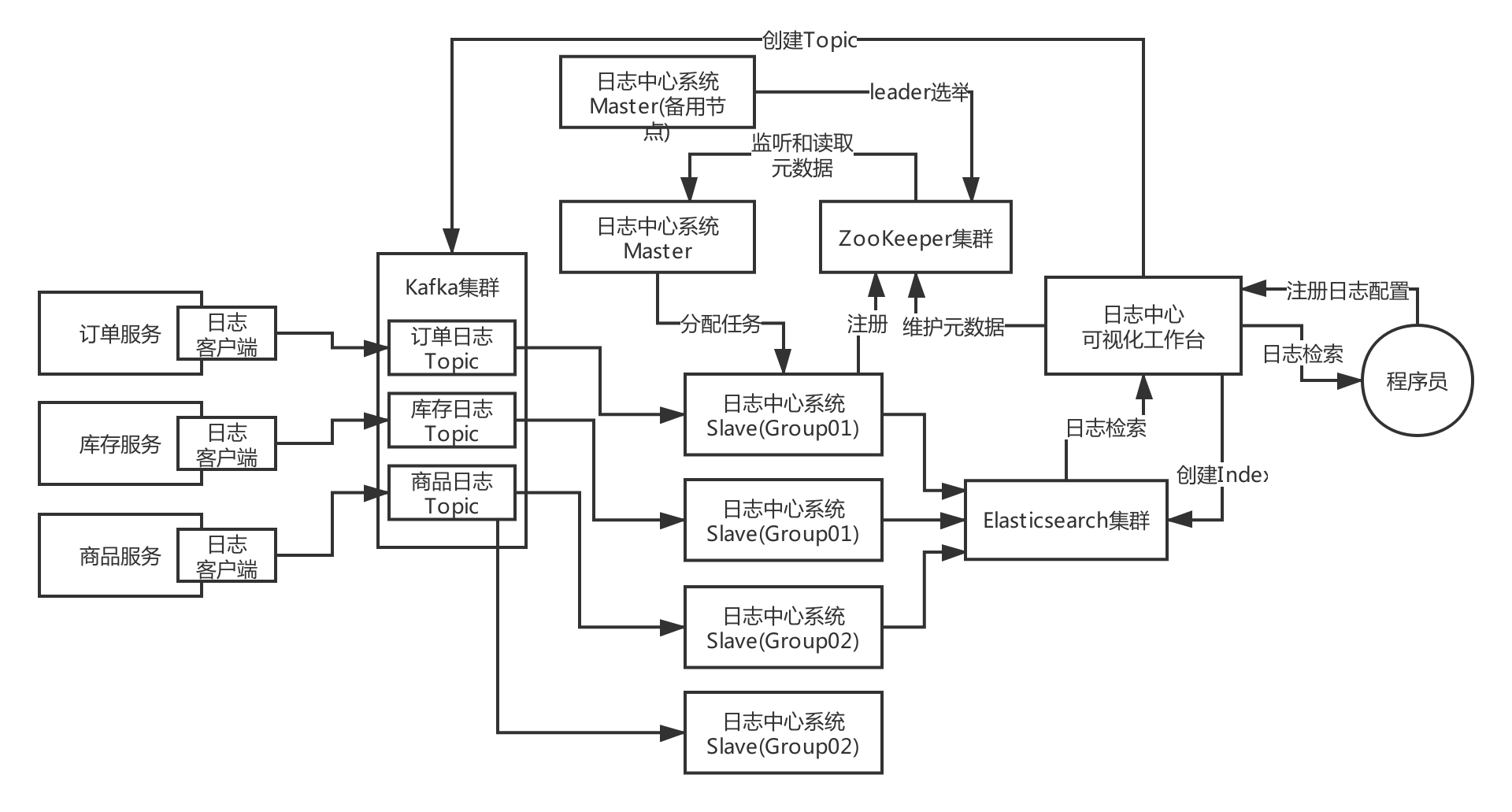
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/006\_ZooKeeper集群的三种角色：Leader、Follower、Observer/01\_ZooKeeper架构原理.png**



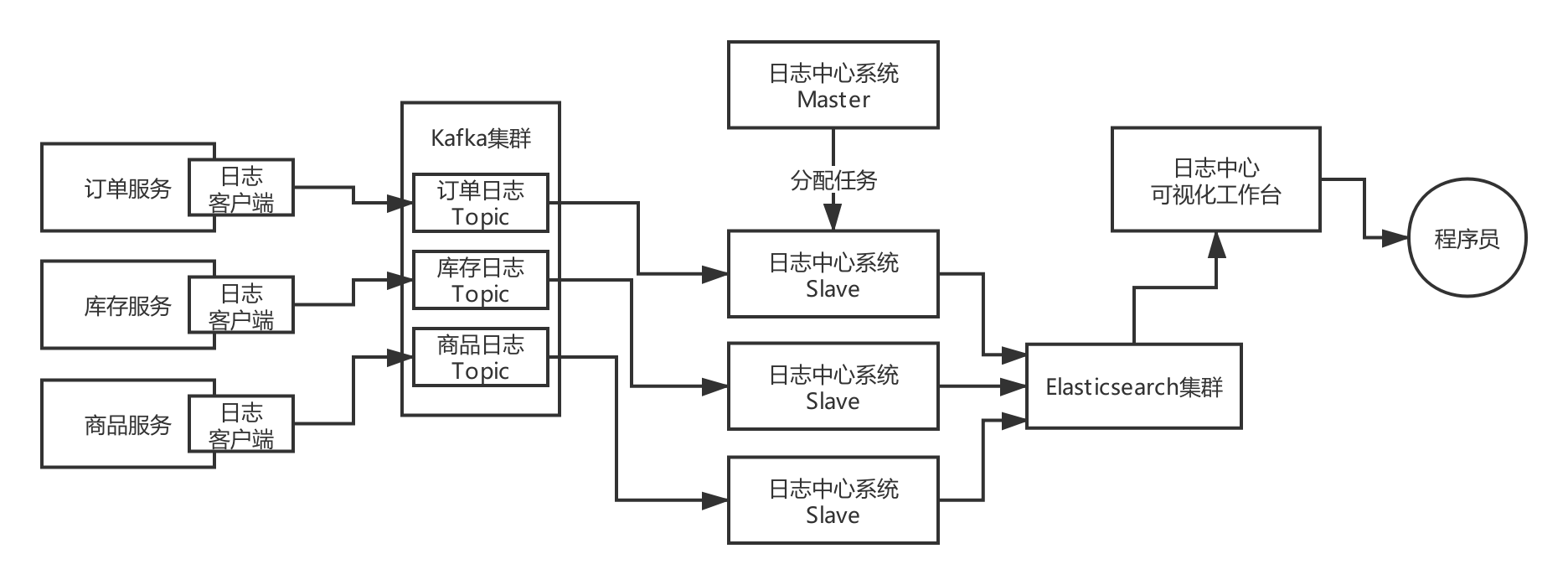
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/008\_ZooKeeper的数据模型：znode和节点类型/01\_ZooKeeper架构原理.png**



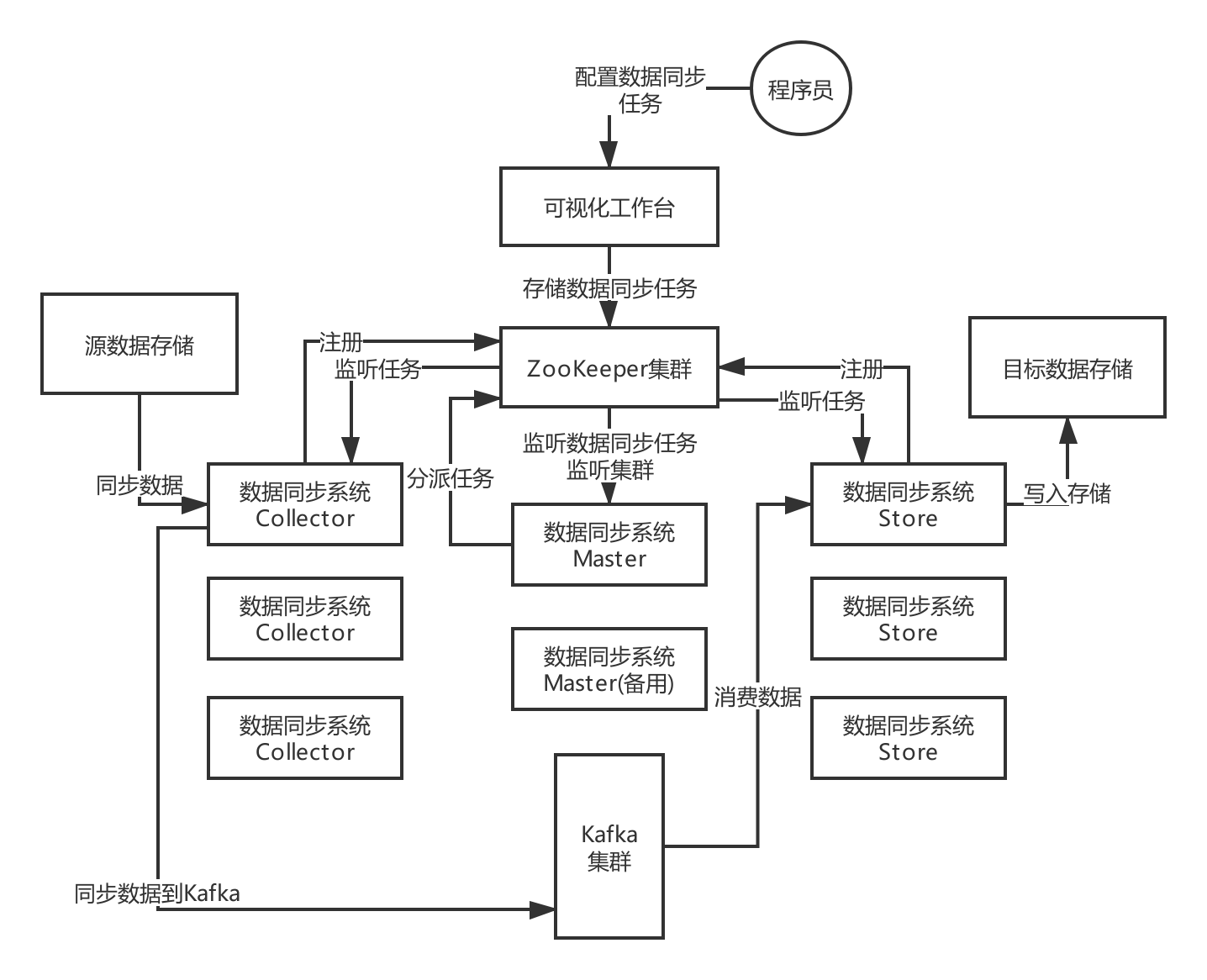
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/095\_基于ZooKeeper与Elasticsearch的日志中心课程的预告/04\_基于ES的日志中心架构.png**



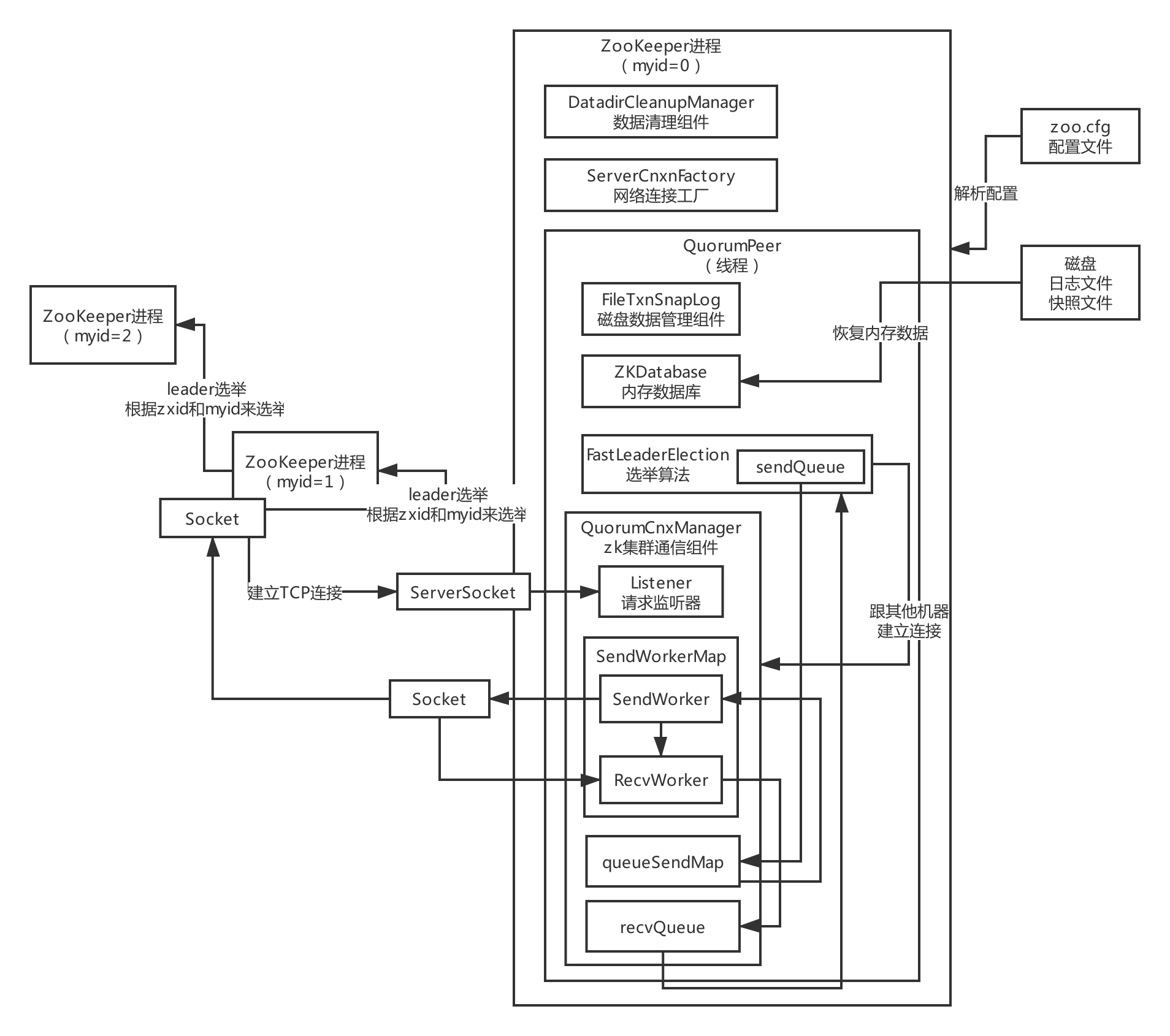
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/089\_生产案例：基于Elasticsearch的日志中心架构设计/04\_基于ES的日志中心架构.png**



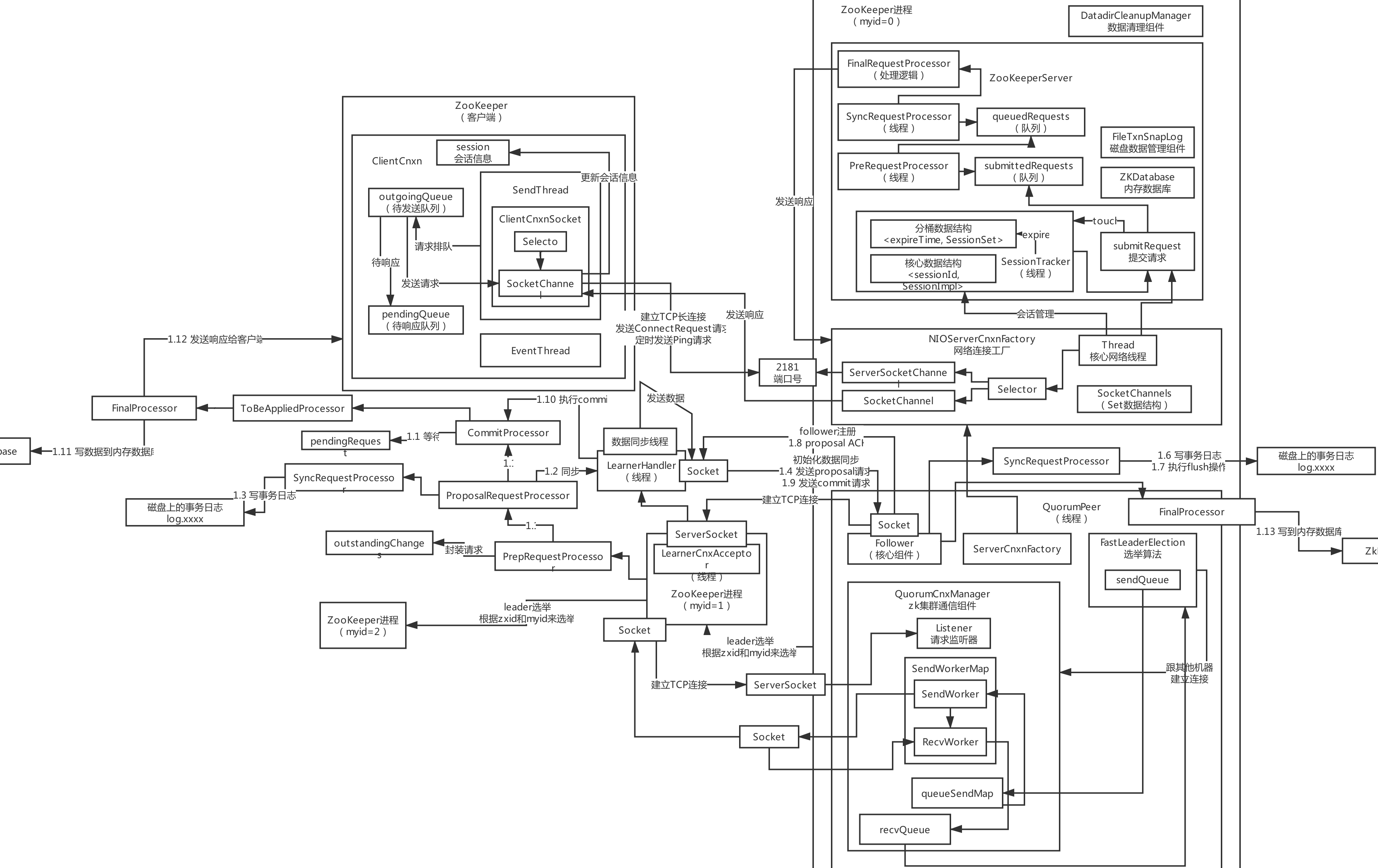
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/098\_基于ZooKeeper实现一个通用数据同步系统的课程预告/05\_通用数据同步平台的架构设计.png**



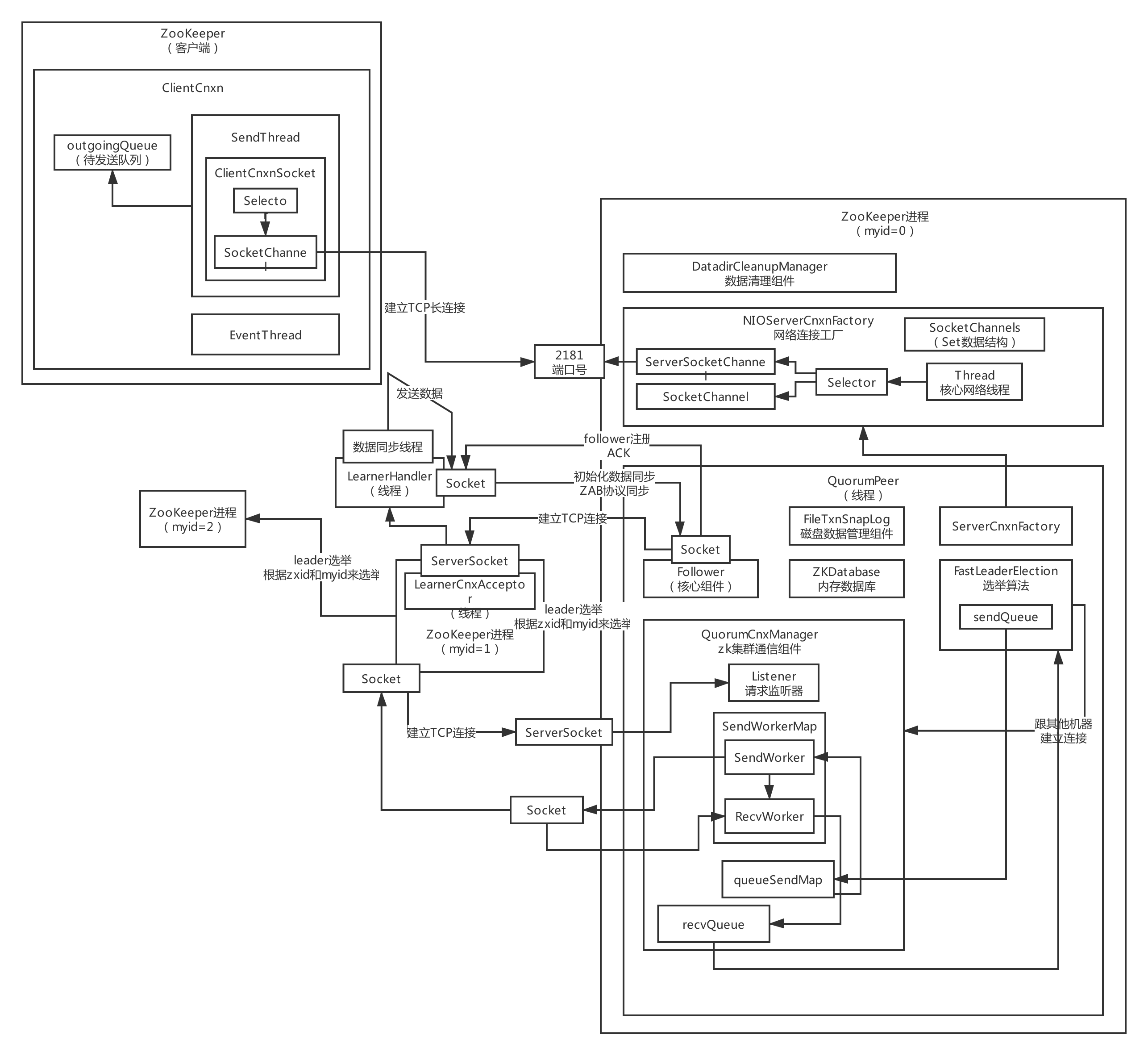
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/120\_如何根据选举结果更新自己的角色状态？/06\_ZooKeeper源码级的架构原理.png**



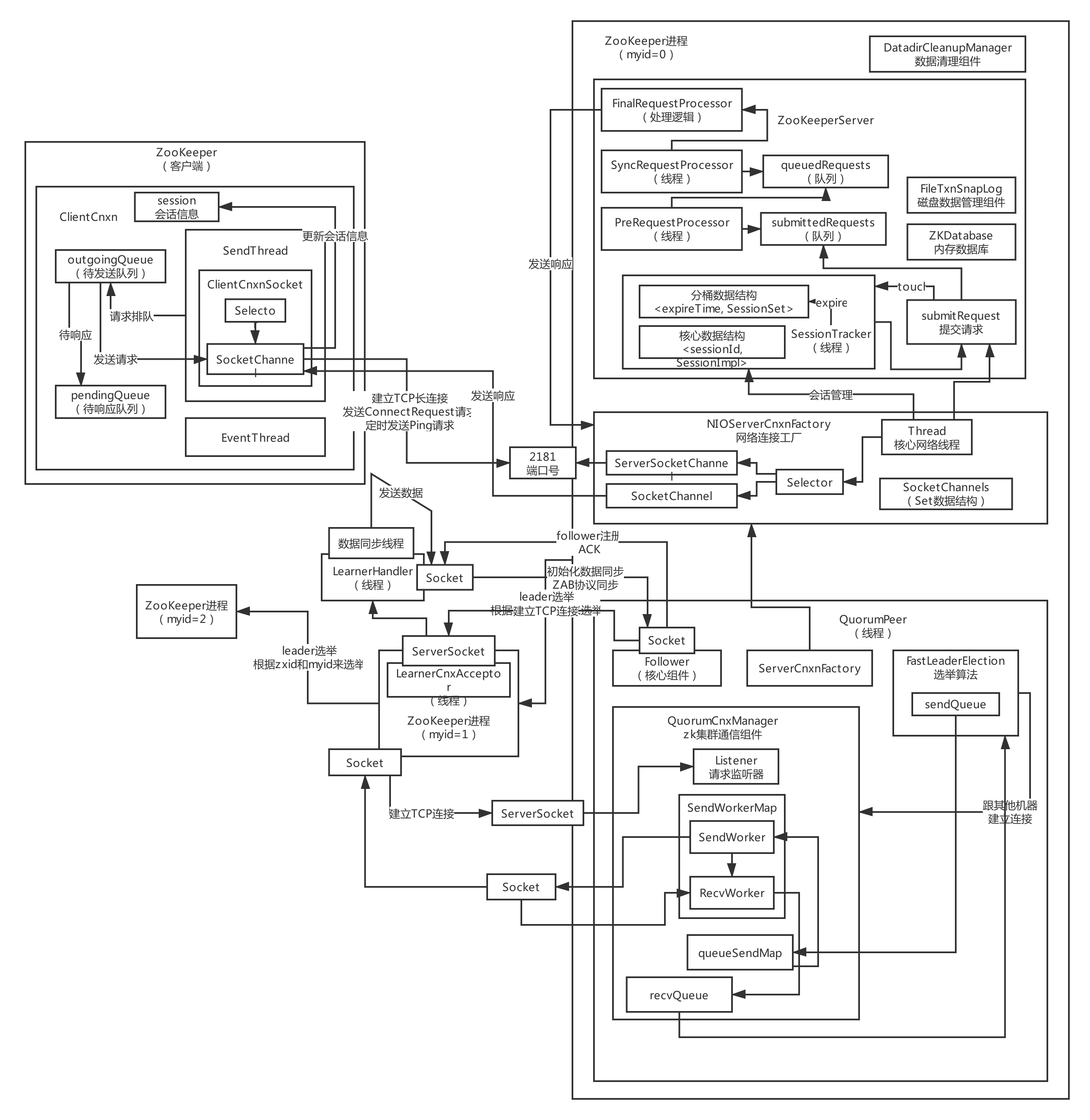
# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/179\_本周源码剖析内容的一个总结/06\_ZooKeeper源码级的架构原理.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/139\_下周课程预告：Session的建立以及工业级的Session管理机制/06\_ZooKeeper源码级的架构原理.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/159\_如果客户端崩溃或者重启，此时对会话是如何处理的？/06\_ZooKeeper源码级的架构原理.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/004\_ZooKeeper为了满足分布式系统的需求要有哪些特性？/笔记.docx**

《004\_ZooKeeper为了满足分布式系统的需求要有哪些特点？》

ZooKeeper肯定是一套系统，这个系统可以存储元数据，支持Master选举，可以进行分布式协调和通知

集群部署：不可能单机版本

顺序一致性：所有请求全部有序

原子性：要么全部机器都成功，要么全部机器都别成功

数据一致性：无论连接到哪台ZK上去，看到的都是一样的数据，不能有数据不一致

高可用：如果某台机器宕机，要保证数据绝对不能丢失

实时性：一旦数据发生变更，其他人要实时感知到

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/019\_ZooKeeper为什么只能是小集群部署？为什么适合读多写少场景？/笔记.docx**

《019\_ZooKeeper为什么只能是小集群部署？为什么适合读多写少场景？》

大数据的同学，Java架构，分布式架构，Eureka源码解析

Eureka，peer-to-peer架构，master-slave

小集群部署，每个节点收到的注册、心跳所有的信息，都必须向其他节点都进行同步，有很大的问题，他在进行同步的时候，采取的是完全的一个异步同步的机制，不管什么2PC，异步慢慢同步就可以了

时效性是很差的，eureka，这个技术不适合大公司，大厂的场景去使用

现在第二个问题，为什么zk的leader和follower只能是三五台机器，小集群部署？因为你想，假设你有1个leader + 20个follower，21台机器，你觉得靠谱吗？不靠谱，因为follower要参与到ZAB的写请求过半ack里去

如果你有20个follower，一个写请求出去，要起码等待10台以上的Follower返回ack，才能发送commit，才能告诉你写请求成功了，性能是极差的

所以zk的这个ZAB协议就决定了一般其实就是1个leader + 2个follower的小集群就够了，写请求是无法扩展的，读请求如果量大，可以加observer机器，最终就是适合读多写少的场景

主要就是用于分布式系统的一些协调工作

这也就让大家知道了，很多互联网公司里，不少系统乱用zk，以为zk可以承载高并发写，结果每秒几万写请求下去，zk的leader机器直接可能就挂掉了，扛不住那么大的请求量，zk一旦挂掉，连带的kafka等系统会全部挂掉

zk适合读多写少的，zk集群挂掉了

leader写入压力过大， 最终导致集群挂掉了，对一个公司的技术平台是有重大打击的，hbase、kafka之类的一些技术都是强依赖zk的，dubbo + zk去做服务框架的话，有上万甚至几十瓦的服务实例的时候

大量的服务的上线、注册、心跳的压力，达到了每秒几万，甚至上十万，zk的单个leader写入是扛不住那么大的压力的

一般适合写比较少

读比较多，observer节点去线性扩展他的高并发读的能力

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/007\_客户端与ZooKeeper之间的长连接和会话是什么？/笔记.docx**

《007\_客户端与ZooKeeper之间的长连接和会话是什么？》

zk集群启动之后，自己分配好角色，然后客户端就会跟zk建立连接，是TCP长连接

把我们的Java架构课程里的网络那块的东西，自研的分布式海量小文件存储系统的项目，我们手写了大量的底层的网络通信的代码

也就建立了一个会话，就是session，可以通过心跳感知到会话是否存在，有一个sessionTimeout，意思就是如果连接断开了，只要客户端在指定时间内重新连接zk一台机器，就能继续保持session，否则session就超时了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/001\_我们一般到底用ZooKeeper来干什么事儿？/笔记.docx**

ZooKeeper顶尖高手课程：从实战到源码

Kafka里面大量使用了ZooKeeper进行元数据管理、Master选举、分布式协调，Canal也是一样，ZooKeeper进行元数据管理，Master选举实现HA主备切换

HDFS，HA也是基于ZK来做的

6周的，zk核心原理，zk集群部署、运维和管理，zk实战开发，zk在hdfs、kafka、canal源码中的运用的分析，两周的时间研究zk的核心的内核源码和底层原理

《001\_我们一般到底用ZooKeeper来干什么事儿？》

Java架构的课，分布式架构中，分布式锁，Redis分布式锁，ZooKeeper分布式锁

分布式锁：运用于分布式的Java业务系统中

元数据管理：Kafka、Canal，本身都是分布式架构，分布式集群在运行，本身他需要一个地方集中式的存储和管理分布式集群的核心元数据，所以他们都选择把核心元数据放在zookeeper中的

分布式协调：如果有人对zk中的数据做了变更，然后zk会反过来去通知其他监听这个数据的人，告诉别人这个数据变更了，kafka有多个broker，多个broker会竞争成为一个controller的角色

如果作为controller的broker挂掉了，此时他在zk里注册的一个节点会消失，其他broker瞬间会被zk反向通知这个事情，继续竞争成为新的controller

这个就是非常经典的一个分布式协调的场景，有一个数据，一个broker注册了一个数据，其他broker监听这个数据

Master选举 -> HA架构

HDFS，NameNode HA架构，部署主备两个NameNode，只有一个人可以通过zk选举成为Master，另外一个backup

Canal，HA

ZooKeeper，分布式协调系统，封装了分布式架构中所有核心和主流的需求和功能，分布式锁、分布式集群的集中式元数据存储、Master选举、分布式协调和通知

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/010\_一个关键的问题：ZooKeeper到底通过什么协议在集群间进行数据一致性同步？/笔记.docx**

《010\_一个关键的问题：ZooKeeper到底通过什么协议在集群间进行数据一致性同步？》

在整个zk的架构和工作原理中，有一个非常关键的环节，就是zk集群的数据同步是用什么协议做的？其实用的是特别设计的ZAB协议，ZooKeeper Atomic Broadcast，就是ZooKeeper原子广播协议

原子性

通过这个协议来进行zk集群间的数据同步，保证数据的强一致性

zk，就说他的一致性的问题，很多同学可能不理解zk的一致性

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/016\_崩溃恢复时选举出来的Leader是如何跟其他Follower进行同步的？/笔记.docx**

《016\_崩溃恢复时选举出来的Leader是如何跟其他Follower进行同步的？》

新选举出来一个leader之后，本身人家会挑选已经收到的事务zxid里最大的那个follower作为新的leader。

5个机器，1leader + 4个follower

1个leader把proposal发送给4个follower，其中3个folower（过半）都收到了proposal返回ack了，第四个follower没收到proposal

此时leader执行commit之后自己挂了，commit没法送给其他的follower，commit刚发送给一个follower

剩余的4个follower，只要3个人投票一个人当leader，就是leader

假设那3个收到proposal的follower都投票第四台没有收到proposal的follower当心的leader？这条数据一定永久性丢失了

选择一个拥有事务zxid最大的机器作为新Leader

其他的follower就会跟他进行同步，他给每个follower准备一个队列，然后把所有的proposal都发送给follower，只要过半follower都ack了，就会发送commit给那个follower

所谓的commit操作，就是把这条数据加入内存中的znode树形数据结构里去，然后就对外可以看到了，也会去通知一些监听这个znode的人

如果一个follower跟leader完全同步了，就会加入leader的同步follower列表中去，然后过半follower都同步完毕了，就可以对外继续提供服务了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/020\_一清二楚：再次回头对ZooKeeper特性的总结/笔记.docx**

核心原理，比kafka、hdfs、hbase之类的要简单很多，最最底层的技术，很多人都依赖他

《020\_一清二楚：再次回头对ZooKeeper特性的总结》

集群模式部署

一般奇数节点，因为你5台机器可以挂2台，6台机器也是挂2台，不能超过一半的机器挂掉，所以5台和6台效果一致，那奇数节点可以减少机器开销，小集群部署，读多写少

主从架构：Leader、Follower、Observer（一般刚开始没必要用）

内存数据模型：znode，多种节点类型

客户端跟zk进行长连接，TCP，心跳，维持session

zxid，高32位，低32位

ZAB协议，2PC，过半ack + 磁盘日志写，commit + 写内存数据结构

支持Watcher机制，监听回调通知

顺序一致性：消息按顺序同步，但是最终才会一致，不是强一致

高性能，2PC中的过半写机制，纯内存的数据结构，znode

高可用，follower宕机没影响，leader宕机有数据不一致问题，新选举的leader会自动处理，正常运行，但是在恢复模式期间，可能有一小段时间是没法写入zk的

高并发，单机leader写，Observer可以线性扩展读QPS

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/005\_为了满足分布式系统的需求，ZooKeeper的架构设计有哪些特点？/笔记.docx**

《005\_为了满足分布式系统的需求，ZooKeeper的架构设计有哪些特点？》

为了实现需要的一些特性，ZooKeeper的架构设计需要有哪些特点？

集群化部署：3~5台机器组成一个集群，每台机器都在内存保存了zk的全部数据，机器之间互相通信同步数据，客户端连接任何一台机器都可以

树形结构的数据模型：znode，树形结构，数据模型简单，纯内存保存

数据结构就跟我们的文件系统是类似的，是有层级关系的树形的文件系统的数据结构

znode可以认为是一个节点而已

create /usr/local/uid

create /usr/local/test\_file

uid：可以写入一些数据的值，比如说hello world

test\_file：也可以写入一些数据的值

顺序写：集群中只有一台机器可以写，所有机器都可以读，所有写请求都会分配一个zk集群全局的唯一递增编号，zxid，保证各种客户端发起的写请求都是有顺序的

数据一致性：任何一台zk机器收到了写请求之后都会同步给其他机器，保证数据的强一致，你连接到任何一台zk机器看到的数据都是一致的

高性能：每台zk机器都在内存维护数据，所以zk集群绝对是高并发高性能的，如果你让zk部署在高配置物理机上，一个3台机器的zk集群抗下每秒几万请求没有问题

高可用：哪怕集群中挂掉不超过一半的机器，都能保证可用，数据不会丢失，3台机器可以挂1台，5台机器可以挂2台

高并发：高性能决定的，只要基于纯内存数据结构来处理，并发能力是很高的，只有一台机器进行写，但是高配置的物理机，比如16核32G，写入几万QPS，读，所有机器都可以读，3台机器的话，起码可以支撑十几万QPS

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/009\_ZooKeeper最核心的一个机制：Watcher监听回调/笔记.docx**

《009\_ZooKeeper最核心的一个机制：Watcher监听回调》

ZooKeeper最核心的机制，就是你一个客户端可以对znode进行Watcher监听，然后znode改变的时候回调通知你的这个客户端，这个是非常有用的一个功能，在分布式系统的协调中是很有必要的

支持写和查：只能实现元数据存储，Master选举，部分功能

分布式系统的协调需求：分布式架构中的系统A监听一个数据的变化，如果分布式架构中的系统B更新了那个数据/节点，zk反过来通知系统A这个数据的变化

/usr/local/uid

使用zk很简单，内存数据模型（不同节点类型）；写数据，主动读取数据；监听数据变化，更新数据，反向通知数据变化

实现分布式集群的集中式的元数据存储、分布式锁、Master选举、分布式协调监听

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/013\_采用了2PC两阶段提交思想的ZAB消息广播流程/笔记.docx**

《013\_采用了2PC两阶段提交思想的ZAB消息广播流程》

每一个消息广播的时候，都是2PC思想走的，先是发起事务Proposal的广播，就是事务提议，仅仅只是个提议而已，各个follower返回ack，过半follower都ack了，就直接发起commit消息到全部follower上去，让大家提交

发起一个事务proposal之前，leader会分配一个全局唯一递增的事务id，zxid，通过这个可以严格保证顺序

leader会为每个follower创建一个队列，里面放入要发送给follower的事务proposal，这是保证了一个同步的顺序性

每个follower收到一个事务proposal之后，就需要立即写入本地磁盘日志中，写入成功之后就可以保证数据不会丢失了，然后返回一个ack给leader，然后过半follower都返回了ack，leader推送commit消息给全部follower

leader自己也会进行commit操作

commit之后，就意味这个数据可以被读取到了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/011\_ZAB的核心思想介绍：主从同步机制和崩溃恢复机制/笔记.docx**

《011\_ZAB的核心思想介绍：主从同步机制和崩溃恢复机制》

协议的本质而言，划分集群角色，主从架构，Leader和Follower两种角色

只有Leader可以接受写操作，Leader和Follower都可以读，Leader收到事务请求，转换为事务Proposal（提议）同步给所有的Follower，超过半数的Follower都说收到事务proposal了，Leader再给所有的Follower发一个Commit消息，让所有Follower提交一个事务

而且如果Leader崩溃了，要重新选举Leader保证继续运行

角色划分，2PC（两阶段），过半写机制

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/006\_ZooKeeper集群的三种角色：Leader、Follower、Observer/笔记.docx**

《006\_ZooKeeper集群的三种角色：Leader、Follower、Observer》

通常来说ZooKeeper集群里有三种角色的机器

集群启动自动选举一个Leader出来，只有Leader是可以写的，Follower是只能同步数据和提供数据的读取，Leader挂了，Follower可以继续选举出来Leader，Observer也只能读但是Observer不参与选举

Observer这个东西，再往后讲点，分析清楚了别的东西以后，Observer我们会回过头来讲解的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/002\_有哪些开源的分布式系统中使用了ZooKeeper？/笔记.docx**

《002\_有哪些开源的分布式系统中使用了ZooKeeper？》

Canal、Kafka、HDFS，学习过的这些技术都用了ZooKeeper，元数据管理，Master选举

ZooKeeper，他主要是提供哪些功能，满足哪些需求，使用在哪些场景下，最后一句话总结，ZooKeeper到底是为什么而生的，定位是什么？

三类系统

第一类：分布式Java业务系统，分布式电商平台，大部分的Java开发的互联网平台，或者是传统架构系统，都是分布式Java业务系统，Dubbo、Spring Cloud把系统拆分成很多的服务或者是子系统，大家协调工作，完成最终的功能

ZooKeeper，用的比较少，分布式锁的功能，而且很多人会选择用Redis分布式锁

第二类：开源的分布式系统

Dubbo，HBase，HDFS，Kafka，Canal，Storm，Solr

分布式集群的集中式元数据存储、Master选举实现HA架构、分布式协调和通知

Dubbo：ZooKeeper作为注册中心，分布式集群的集中式元数据存储

HBase：分布式集群的集中式元数据存储

HDFS：Master选举实现HA架构

Kafka：分布式集群的集中式元数据存储，分布式协调和通知

Canal：分布式集群的集中式元数据存储，Master选举实现HA架构

第三类：自研的分布式系统

HDFS，面向的超大文件，切割成一个一个的小块儿，分布式存储在一个大的集群里

分布式海量小文件系统：NameNode的HA架构，仿照HDFS的NameNode的HA架构，做主备两个NameNode，进行数据同步，然后自动基于zk进行热切换

在很多，如果你自己研发类似的一些分布式系统，都可以考虑，你是否需要一个地方集中式存储分布式集群的元数据？是否需要一个东西辅助你进行Master选举实现HA架构？进行分布式协调通知？

如果你在自研分布式系统的时候，有类似的需求，那么就可以考虑引入ZooKeeper来满足你的需求

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/012\_从zk集群启动到数据同步再到崩溃恢复的ZAB协议流程/笔记.docx**

《012\_从zk集群启动到数据同步再到崩溃恢复的ZAB协议流程》

zk集群启动的时候，进入恢复模式，选举一个leader出来，然后leader等待集群中过半的follower跟他进行数据同步，只要过半follower完成数据同步，接着就退出恢复模式，可以对外提供服务了

**只要有超过一半的机器，认可你是leader，你就可以被选举为leader**

3台机器组成了一个zk集群，启动的时候，只要有2台机器认可一个人是Leader，那么他就可以成为leader了

3台可以容忍不超过一半的机器宕机，1台

剩余的2台机器，只要2台机器都认可其中某台机器时leader，2台 > 一半，就可以选举出来一个leader了

zk的leader选举算法，我们可以在后面的zk核心源码剖析的时候

1台机器时没有办法自己选举自己的

5台机器，3台机器认可某个人是leader；可以允许2台机器宕机，3台机器，leader选举，只要是5台机器，一半2.5，3台机器都认可某个人是leader，此时3 > 2.5，过半，leader是可以选举出来的

2台机器，小于一半，没有办法选举新的leader出来了

当然还没完成同步的follower会自己去跟leader进行数据同步的

此时会进入消息广播模式

只有leader可以接受写请求，但是客户端可以随便连接leader或者follower，如果客户端连接到follower，follower会把写请求转发给leader

leader收到写请求，就把请求同步给所有的follower，过半follower都说收到了，就再发commit给所有的follower，让大家提交这个请求事务

如果突然leader宕机了，会进入恢复模式，重新选举一个leader，只要过半的机器都承认你是leader，就可以选举出来一个leader，所以zk很重要的一点是主要宕机的机器数量小于一半，他就可以正常工作

因为主要有过半的机器存活下来，就可以选举新的leader

新leader重新等待过半follower跟他同步，完了重新进入消息广播模式

2PC是什么，两阶段提交，Java架构里的分布式事务的课程，好好去看一下

集群启动：恢复模式，leader选举（过半机器选举机制） + 数据同步

消息写入：消息广播模式，leader采用2PC模式的过半写机制，给follower进行同步

崩溃恢复：恢复模式，leader/follower宕机，只要剩余机器超过一半，集群宕机不超过一半的机器，就可以选举新的leader，数据同步

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/015\_ZAB协议下两种可能存在的数据一致性问题/笔记.docx**

《015\_ZAB协议下一种可能存在的数据一致性问题》

Leader收到了过半的follower的ack，接着leader自己commit了，还没来得及发送commit给所有follower自己就挂了，这个时候相当于leader的数据跟所有follower是不一致的，你得保证全部follower最终都得commit

另外一个，leader可能会自己收到了一个请求，结果没来得及发送proposal给所有follower之前就宕机了，此时这个Leader上的请求应该是要被丢弃掉的

所以在leader崩溃的时候，就会选举一个拥有事务id最大的机器作为leader，他得检查事务日志，如果发现自己磁盘日志里有一个proposal，但是还没提交，说明肯定是之前的leader没来得及发送commit就挂了

此时他就得作为leader为这个proposal发送commit到其他所有的follower中去，这个就保证了之前老leader提交的事务已经会最终同步提交到所有follower里去

然后对于第二种情况，如果老leader自己磁盘日志里有一个事务proposal，他启动之后跟新leader进行同步，发现这个事务proposal其实是不应该存在的，就直接丢弃掉就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/014\_停一下脚步：ZooKeeper到底是强一致性还是最终一致性？/笔记.docx**

《014\_停一下脚步：ZooKeeper到底是强一致性还是最终一致性？》

强一致性：只要写入一条数据，立马无论从zk哪台机器上都可以立马读到这条数据，强一致性，你的写入操作卡住，直到leader和全部follower都进行了commit之后，才能让写入操作返回，认为写入成功了

此时只要写入成功，无论你从哪个zk机器查询，都是能查到的，强一致性

明显，ZAB协议机制，zk一定不是强一致性

最终一致性：写入一条数据，方法返回，告诉你写入成功了，此时有可能你立马去其他zk机器上查是查不到的，短暂时间是不一致的，但是过一会儿，最终一定会让其他机器同步这条数据，最终一定是可以查到的

研究了ZooKeeper的ZAB协议之后，你会发现，其实过半follower对事务proposal返回ack，就会发送commit给所有follower了，只要follower或者leader进行了commit，这个数据就会被客户端读取到了

那么有没有可能，此时有的follower已经commit了，但是有的follower还没有commit？绝对会的，所以有可能其实某个客户端连接到follower01，可以读取到刚commit的数据，但是有的客户端连接到follower02在这个时间还没法读取到

所以zk不是强一致的，不是说leader必须保证一条数据被全部follower都commit了才会让你读取到数据，而是过程中可能你会在不同的follower上读取到不一致的数据，但是最终一定会全部commit后一致，让你读到一致的数据的

zk官方给自己的定义：顺序一致性

因此zk是最终一致性的，但是其实他比最终一致性更好一点，出去要说是顺序一致性的，因为leader一定会保证所有的proposal同步到follower上都是按照顺序来走的，起码顺序不会乱

但是全部follower的数据一致确实是最终才能实现一致的

如果要求强一致性，可以手动调用zk的sync()操作

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/018\_现在再来看看ZooKeeper的Observer节点是用来干什么的？/笔记.docx**

《018\_现在再来看看ZooKeeper的Observer节点是用来干什么的？》

Observer节点是不参与leader选举的，他也不参与ZAB协议同步时候的过半follower ack的那个环节，他只是单纯的接收数据，同步数据，可能数据存在一定的不一致的问题，但是是只读的

leader在进行数据同步的时候，observer是不参与到过半写机制里去

所以大家思考一个问题了

zk集群无论多少台机器，只能是一个leader进行写，单机写入最多每秒上万QPS，这是没法扩展的，所以zk是适合写少的场景

但是读呢？follower起码有2个或者4个，读你起码可以有每秒几万QPS，没问题，那如果读请求更多呢？此时你可以引入Observer节点，他就只是同步数据，提供读服务，可以无限的扩展机器

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/017\_对于需要丢弃的消息是如何在ZAB协议中进行处理的？/笔记.docx**

《017\_对于需要丢弃的消息是如何在ZAB协议中进行处理的？》

每一条事务的zxid是64位的，高32位是leader的epoch，就认为是leader的版本吧；低32位才是自增长的zxid

~~老leader发送出去的proposal，高32位是1，低32位是11358~~

如果一个leader自己刚把一个proposal写入本地磁盘日志，就宕机了，没来得及发送给全部的follower，此时新leader选举出来，他会的epoch会自增长一位

proposal，高32位是2，低32位是继续自增长的zxid

然后老leader恢复了连接到集群是follower了，此时发现自己比新leader多出来一条proposal，但是自己的epoch比新leader的epoch低了，所以就会丢弃掉这条数据

启动的时候，过半机器选举leader，数据同步

对外提供服务的时候，2PC + 过半写机制，顺序一致性（最终的一致性）

崩溃恢复，剩余机器过半，重新选举leader，有数据不一致的情况，针对两种情况自行进行处理，保证数据是一致的（磁盘日志文件、zxid的高32位）

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/008\_ZooKeeper的数据模型：znode和节点类型/笔记.docx**

《008\_ZooKeeper的数据模型：znode和节点类型》

核心数据模型就是znode树，平时我们往zk写数据就是创建树形结构的znode，里面可以写入值，就这数据模型，都在zk内存里存放

有两种节点，持久节点和临时节点，持久节点就是哪怕客户端断开连接，一直存在

临时节点，就是只要客户端断开连接，节点就没了

还有顺序节点，就是创建节点的时候自增加全局递增的序号

大家去看一下，之前Java架构的分布式锁里，有一个zk锁的源码分析，curator框架，zk分布式锁的实现，在里面就是基于zk的临时顺序节点来实现的，加锁的时候，是创建一个临时顺序节点

zk会自动给你的临时节点加上一个后缀，全局递增的，编号

如果你客户端断开连接了，就自动销毁这个你加的锁，此时人家会感知到，就会尝试去加锁

如果你是做元数据存储，肯定是持久节点

如果你是做一些分布式协调和通知，很多时候是用临时节点，就是说，比如我创建一个临时节点，别人来监听这个节点的变化，如果我断开连接了，临时节点消失，此时人家会感知到，就会来做点别的事情

顺序节点，在分布式锁里用的比较经典

每个znode还有一个Stat用来存放数据版本，version（znode的版本），cversion（znode子节点的版本），aversion（znode的ACL权限控制版本）

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/001~020资料/003\_为什么我们在分布式系统架构中需要使用ZooKeeper集群？/笔记.docx**

《003\_为什么我们在分布式系统架构中需要使用ZooKeeper集群？》

ZooKeeper，功能和定位，满足的需求

使用ZooKeeper去满足自己需求的项目都有哪些

分布式集群的集中式元数据存储，Master选举实现HA架构，分布式协调和通知

我们写一个类似ZK的系统，单机版本，就是部署在一台机器上面，里面提供了一些功能，比如说允许你在里面存储一些元数据，支持你进行Master选举，支持你分布式协调和通知，也可以做到

单机版本的系统，万一挂掉了怎么办？

集群部署，部署一个集群出来，多台机器，保证高可用性，挂掉一台机器，都可以继续运行下去

3台机器

我现在要进行元数据的存储，我向机器01写了一条数据，机器01应该怎么把数据同步给其他的机器02和机器03呢？

自己写一个类似ZK的系统？不可能单机版本吧？肯定得集群部署保证高可用吧？一旦集群了之后，数据一致性怎么保证？多麻烦！

你的分布式架构中有需求，干脆就直接用工业级，久经考验的zookeeper就可以了，bug很少，功能很全面，运用在很多工业级的大规模的分布式系统中，HDFS、Kafka、HBase

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/117\_如何接收其他机器发送过来的投票？/笔记.docx**

《117\_如何接收其他机器发送过来的投票？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/105\_QuorumPeerMain启动时的基本流程/笔记.docx**

《105\_QuorumPeerMain启动时的基本流程》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/101\_将ZooKeeper源码导入Intellij IDEA中/笔记.docx**

《101\_将ZooKeeper源码导入Intellij IDEA中》

Zookeeper架构原理、集群部署+运维管理、客户端框架源码（Curator）、实践案例（日志中心、iplist服务）

zk：存储元数据、监听回调、集群化部署+ZAB协议

3~4周时间，把zk的核心源码剖析掉，leader选举算法、ZAB协议算法、日志机制、数据快照机制、数据清理机制、长连接机制、会话机制、监听回调机制，4周时间，对zk应该就是国内数一数二的专家了

去官网下载apache-ant-1.10.5-bin.tar.gz，解压缩，然后配置环境变量ANT\_HOME，把$ANT\_HOME/bin加入到PATH环境变量里去，最后ant -version检查一下ant的版本号，是否安装成功

去官网下载zookeeper-3.4.5-tar.gz，解压缩，在zk目录下，然后进去执行ant eclipse命令，等待几分钟即可，就会看到build success的字样

最后打开intellij idea，选择import project，选择这个zookeeper的源码目录，然后选择from existing external project，选择eclipse项目，也就是把eclipse项目导入intellij idea中，就这个意思

然后打开就可以看到源码了，非常的方便

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/116\_SenderWorker是如何将投票发送给其他机器的？/笔记.docx**

《116\_SenderWorker是如何将投票发送给其他机器的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/115\_当两台机器建立连接之后是如何开始发起投票的？/笔记.docx**

《115\_当两台机器建立连接之后是如何开始发起投票的？》

你完全可以这样，在connectAll()，在这个方法里面，你去判断一下，你只能对比你sid小的机器发起连接请求，myid=0的机器，就不对任何人发起连接请求

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/106\_使用集群模式启动的时候，QuorumPeer是用来干什么的？（上）/笔记.docx**

《106\_使用集群模式启动的时候，QuorumPeer是用来干什么的？（上）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/102\_如何在本地启动和调试ZooKeeper源码？/笔记.docx**

《102\_如何在本地启动和调试ZooKeeper源码？》

输入启动参数，打上断点，就可以开始调试

其实如果要在本地部署zk集群然后调试也是可以的，但是其实在集群化部署的时候打断点是没什么意义的，因为你胡乱卡死人家的执行流程，会导致一些超时之类的莫名其妙的问题，不按照你的想法去运行

所以，还是要习惯于阅读静态源码

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/114\_ZooKeeper是如何避免两台机器重复建立TCP连接的？/笔记.docx**

《114\_ZooKeeper是如何避免两台机器重复建立TCP连接的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/103\_如何在修改ZooKeeper源码之后重新打包？/笔记.docx**

《103\_如何在修改ZooKeeper源码之后重新打包？》

如果要修改zk源码，直接改就行了，完事儿了就是在zk目录下，执行ant命令，就会自动打包出来一个可执行的包，ant类似于maven，下载一些第三方的依赖jar包，在编译、测试、构建和打包的时候，自动化的把你依赖的第三方的jar包跟你打在一起

gradle

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/110\_QuorumPeer的线程主逻辑中是如何根据节点状态启动Leader选举的？/笔记.docx**

《110\_QuorumPeer的线程主逻辑中是如何根据节点状态启动Leader选举的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/109\_QuorumPeer是如何开启一次Leader选举的？/笔记.docx**

《109\_QuorumPeer是如何开启一次Leader选举的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/112\_QuorumPeer是如何尝试去跟其他机器建立连接的？/笔记.docx**

《112\_QuorumPeer是如何尝试去跟其他机器建立连接的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/119\_如何通过归档的选票统计出谁当选Leader？/笔记.docx**

《119\_如何通过归档的选票统计出谁当选Leader？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/107\_使用集群模式启动的时候，QuorumPeer是用来干什么的？（下）/笔记.docx**

《107\_使用集群模式启动的时候，QuorumPeer是用来干什么的？（下）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/108\_ZooKeeper的Leader选举流程分析/笔记.docx**

《108\_ZooKeeper的Leader选举流程分析》

三台机器，myid分别是0，1，2

另外一个，这是一个空的集群，所以说每台机器上没有数据的，当前他们本地磁盘上最大的一条数据的zxid认为是0

启动集群的顺序是：0，1，2

如果当myid=0的机器，先启动之后，他是无法开始进行leader选举的，因为集群的机器是3台，此时他的quorum=(3/2 + 1)=2，大多数机器是2台，集群中必须有两台机器都启动了，此时才可以开启一轮leader选举

接着启动了myid=1的机器，集群里有2台机器了，达到了quorum的数量，此时就可以发起leader选举了，myid=2的机器可能还没启动，但是已经有的两台机器已经开始leader选举了，甚至可能在myid=3的机器还没启动的时候，leader就已经选举出来了

第一轮投票

myid=0的机器，投票，（0, 0），发送给当前集群里其他的机器

myid=1的机器，投票，（1, 0），发送给当前集群里其他的机器

myid=0的机器，投出去（0,0），收到的票是（1,0），zxid最大的机器包含的数据是最新的，优先是zxid最大的机器成为leader，默认就是让myid最大的机器成为leader，推荐（1,0）成为leader

myid=1的机器，投出去（1,0），收到的票是（0,0），还是推荐自己（1,0）成为leader

第二轮投票

myid=0的机器，投出去（1,0），收到的票是（1,0），投一台机器的票数已经达到了集群的quorum大多数了，2台机器投同一个人作为leader，此时选举就结束了，确定（1,0）机器是leader

myid=1的机器，投出去（1,0），收到的票是（1,0），确定（1,0）机器是leader

投票结束

myid=0的机器，follower

myid=1的机器，leader

myid=2的机器，刚刚才启动，此时发现集群里已经选举出来leader了，此时自己让自己变成follower就可以了

一般来说如果你自己部署在windows上搞3台虚拟机，部署zk集群，一台一台接着启动，第二台机器往往是leader，第一台和第三台是follower，第二台是leader

myid=2的机器，(2,10)

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/113\_QuorumCnxManager是如何基于TCP监听其他机器的连接的？/笔记.docx**

《113\_QuorumCnxManager是如何基于TCP监听其他机器的连接的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/120\_如何根据选举结果更新自己的角色状态？/笔记.docx**

《120\_如何根据选举结果更新自己的角色状态？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/104\_阅读任何一个中间件源码的起点：从启动脚本开始分析/笔记.docx**

《104\_阅读任何一个中间件源码的起点：从启动脚本开始分析》

无论是hdfs、kafka、canal、zookeeper，开源的分布式系统，hdfs可以认为是分布式大数据系统，kafka、canal、zookeeper一般来说，我们把他们都归类到分布式中间件系统里面，分布式系统，部署启动的这种系统

用这些系统的第一步，就是去通过他们的启动脚本，去把他们给跑起来

QuorumPeerMain

Quorum：大多数的一个意思，zk集群，quorum集群，多台机器组成了一个集群，peer，集群各个机器的角色都是一样的，能干的事儿都是一样的，他不是master-slave的架构，peer集群的架构

每台机器都可以成为leader，可以干leader的事儿，每台机器也都可以成为follower，只不过在集群运行的时候，有的机器角色是leader，有的机器角色是follower，peer架构，每台机器能干的事儿都一样

hdfs，master-slave架构，namenode和datanode能干的事儿是绝对不一样的，datanode不能成为namenode

quorum，写一条数据到zk，大多数的follower机器都成功写入了日志，就可以认为这次写入就成功了，大多数机器在英文里就是quorum，quorum机制，大部分的机器都同步成功了，就算是成功了

peer集群，是一个多台peers机器，里面任何一个节点都可以认为是一个peer，跟其他机器一样的一台机器，我们在机器上看到的zk的进程，就是QuorumPeerMain进程，java命令，通过java命令启动一个jvm进程，执行里面指定的main类

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/111\_初步开始查看Leader选举的源码细节/笔记.docx**

《111\_初步开始查看Leader选举的源码细节》

我最不喜欢的就是老外，自以为很聪明，码农，把变量弄的特别的晦涩，sid，bid，cnx

serverId，connection

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/101~120资料/118\_如何进行选票PK以及选票归档？/笔记.docx**

《118\_如何进行选票PK以及选票归档？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/186\_再来看看查询子节点操作的源码实现流程/笔记.docx**

《186\_再来看看查询子节点操作的源码实现流程》

客户端1 -> leader，执行了一个写入操作

客户端2 -> follower，此时执行查询操作，未必可以查到你刚写入成功的数据

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/190\_我们注册的监听器是如何跟着请求发送给zk服务端的？/笔记.docx**

《190\_我们注册的监听器是如何跟着请求发送给zk服务端的？》

监听器，在客户端也要保存一份；在zk服务端也要保存一份

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/191\_监听器是什么时候在客户端完成注册的？/笔记.docx**

《191\_监听器是什么时候在客户端完成注册的？》

似乎在整个发送请求的时候，没有看到监听器在本地客户端进行注册的？如果说你仅仅是在zk服务端注册了一个监听器，zk的内存数据变化的时候，确实是会过来通知你的，但是此时通知你，是需要回调你自己创建的一个回调函数的！

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/188\_遗留的一些细节：每个proposal的zxid是如何创建的？/笔记.docx**

《188\_遗留的一些细节：每个proposal的zxid是如何创建的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/199\_Curator这类zk客户端的好处：使用便捷，功能强大/笔记.docx**

《199\_Curator这类zk客户端的好处：使用便捷，功能强大》

ZooKeeper，Curator，他们都是基于原生的ZooKeeper进行封装，Curator，监听器，他是会自己自动进行重新注册，如果说你的数据变化触发了监听器，自动会删除这个监听器，此时Curator他会在底层自动进行重新注册

基于zk的增删改查 + 两种节点类型 + 顺序节点 + 监听回调 + 数据多副本冗余 + 高可用机制 + 数据持久化存储 + 一致性协议保障 + Leader自动选举，可以实现很多的高阶的功能：分布式锁、master选举、别的一些东西

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/192\_zk服务端是如何处理查询请求中的watch标识的？/笔记.docx**

《192\_zk服务端是如何处理查询请求中的watch标识的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/180\_回顾一下在zk中执行create操作的全源码流程/笔记.docx**

《180\_回顾一下在zk中执行create操作的全源码流程》

create、del、set，增删改，唯一的区别就在于说最后commit的时候，在内存里的文件目录树中是如何进行更新的，在文件目录树立挂载一个znode进去，del就是删除一个znode，set就是跟新一个znode的值

sync，一般很少用，强一致，没有这个必要

这一周的课程，会带着大家去探索zk平时我们常用的一些操作的源码的原理，增删改一些节点，顺序节点是怎么玩儿，查，ZooKeeper客户端里提供的一些API的操作都带着大家去看一下

监听器，有数据的变化是如何反向来通知你的，这个监听器是否会一直生效

健壮的、高可用的、高稳定性的、高可靠性、数据一致性保障

如果中间短暂的发生你的客户端跟zk服务端断开了连接，在session timeout范围之内，此时重新连接之后，watcher会有什么影响吗。如果你的客户端宕机了，session timeout了，此时watcher会有什么影响吗

基于zookeeper才有世界上大量的优秀的开源的分布式系统的项目，中间件系统，大数据系统的项目

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/181\_zk中的删除和修改操作，核心原理是什么？/笔记.docx**

《181\_zk中的删除和修改操作，核心原理是什么？》

setData，delete

增删改，已经搞定了，如果请求发到follower，如何转发给leader

临时节点，如果你的客户端宕机了，临时节点会自动被删除掉；顺序节点是如何来处理的

ZooKeeper里其他的API对应的源码流程，get还有别的一些东西

watch的机制

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/189\_重新梳理一下我们通过原生API是如何创建监听器的？/笔记.docx**

《189\_重新梳理一下我们通过原生API是如何创建监听器的？》

zookeeper最最核心的一块机制，监听和回调机制，对znode文件目录树的数据结构进行增删改查的操作，临时节点，顺序节点，最多只能实现让一些其他的分布式系统、大数据系统可以基于zk进行集群元数据的存储和管理

zookeeper主要是用于一些分布式中间件系统、分布式大数据系统的基础设施，分布式系统，他们一般来说都需要一个集中的地方去存储和管理元数据，同时他们也需要其他的很多功能，协调、选举、高可用自动切换

提供监听和回调的机制，只有把这个功能给实现了，此时zookeeper才是工业级的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/182\_如果增删改请求发送到follower如何转发给leader？/笔记.docx**

《182\_如果增删改请求发送到follower如何转发给leader？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/194\_zk服务端触发监听个器的时候，是如何回调客户端连接的？/笔记.docx**

《194\_zk服务端触发监听个器的时候，是如何回调客户端连接的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/183\_如果创建的是顺序节点，此时会如何进行处理？/笔记.docx**

《183\_如果创建的是顺序节点，此时会如何进行处理？》

持久化的节点做的一些事情，persistent

在这个基础之上，我们来考虑一下，如果是持久化的顺序节点，此时是如何进行创建的呢？

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/198\_基于原生zk客户端的不便之处：重复注册监听器、功能简单/笔记.docx**

《198\_基于原生zk客户端的不便之处：重复注册监听器、功能简单》

ZooKeeper，增删改查 + 监听回调，监听器是一次性的，就是你必须得在每次收到zk的回调通知之后，都重新去注册一次监听器，只有这样才能让你一直保持对某个节点的数据变化的感知

就我个人而言，非常不支持zk原生的服务端+客户端，都是让监听器一次性的，完全可以让这个监听器一直存在，客户端API，取消一个监听器，自动删除他的监听器

在下一次监听器注册成功的过程中，此时如果数据变化，会导致你是没有监听器在上面的，zk在这点上，我觉得

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/197\_如果zk服务端挂掉了，客户端应该做哪些事儿？/笔记.docx**

《197\_如果zk服务端挂掉了，客户端应该做哪些事儿？》

此时那些客户端必须去找其他的机器，leader，follower，去建立长连接，建立会话，重新把自己内存里注册的那些监听器，在新建立连接的机器上去进行一个重新的注册，在新的机器里必须得有这个客户端加的这些监听器

后续的话呢，如果说有你关注的数据的变动，还是会回调通知到你

在这个过程中呢？如果有一些数据的变化，此时是没有办法通知到你的，注册监听器，后续真的在你设计系统的时候，如果说你要使用zookeeper，其实一个复杂的系统，在他出现故障，网络故障，机器故障

系统的行为出现一些不符合你期望的行为

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/184\_如果创建的是临时节点，会如何进行处理？/笔记.docx**

《184\_如果创建的是临时节点，会如何进行处理？》

如果说创建临时节点的客户端挂了，此时他的会话会被删除，会同时删除所有的临时节点，ephemeral，会全部删除掉

顺序节点

%10d，版本号，补齐成一个10位的数字，如果你是1，0000000001

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/187\_再来看看exists操作的源码实现流程是什么？/笔记.docx**

《187\_再来看看exists操作的源码实现流程是什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/193\_分别看一下增删改事件发生的时候是如何触发监听器的？/笔记.docx**

《193\_分别看一下增删改事件发生的时候是如何触发监听器的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/196\_如果客户端挂掉了，zk服务端会做哪些处理？/笔记.docx**

《196\_如果客户端挂掉了，zk服务端会做哪些处理？》

zk服务端而言，他其实是会删除掉你所有的这个客户端创建的临时节点，2PC的模式去进行删除，删除了之后到commit的时候，删除了内存里的节点，一定会触发对应的监听器的，必须的

必须把你的这个客户端加的这些监听器都给干掉

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/180~199资料/195\_zk客户端收到watch监听通知的请求之后，如何处理？/笔记.docx**

《195\_zk客户端收到watch监听通知的请求之后，如何处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/216\_zk重启的时候，是如何加载磁盘上的数据进行恢复的？/笔记.docx**

《216\_zk重启的时候，是如何加载磁盘上的数据进行恢复的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/202\_捕获到zk服务器故障时的IOException，该如何处理？/笔记.docx**

《202\_捕获到zk服务器故障时的IOException，该如何处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/204\_一旦网络连接断开后，客户端的监听器是如何清空的？/笔记.docx**

《204\_一旦网络连接断开后，客户端的监听器是如何清空的？》

一旦说你的zk服务器断开，所有的zk客户端，他们其实都会关闭连接，清空自己的监听器，然后重新去连接一台zk服务器

每次触发事件之后，你还得重新去进行一下注册，要不然监听器是一次性的，下次就没有了。如果跟zk服务器的连接断开了，此时监听器就都没有了，你还是得在创建ZooKeeper的时候给一个默认的监听器

如果你收到了连接断开的通知之后，此时你应该知道zk服务器已经断开了，此时你应该等待，他会自动重新去连接其他的zk服务器，如果连接成功了，还会给你一个通知，告诉你是连接已经重新建立了

重新去进行监听器的注册

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/218\_下次最后一周课程预告：站在ZK源码角度解决生产环境的疑难杂症/笔记.docx**

《218\_下次最后一周课程预告：站在ZK源码角度解决生产环境的疑难杂症》

zk进行集群运维和管理，看人家的开源项目对zk是如何来使用，你自己去使用zk开发一些项目，甚至于站在对zk源码级别透彻理解的角度，去非常的灵活的使用zk做一些事情，大家都已经达到了

集群部署和运维管理，zk的一些高阶使用，站在源码的角度解决生产环境下的zk的一些疑难杂症和问题，集群脑裂问题，如果出现集群故障会不会导致分布式锁的故障和失效，zk集群本身是高可用的，在什么场景下，zk集群会彻底崩溃和故障，在zk使用的场景中，各种各样的异常抛出来，是为什么，如何来解决

hdfs、kafka、zookeeper，在国内都是顶尖的高手之一，成为开源项目的commitor，贡献源码给开源社区了

搞大数据的同学，如果跟我学了一年的大数据的课的话，必然可以学习大量的Java底层技术，包括JDK源码、并发包、集合、分布式系统、电商业务系统、自研分布式海量小文件系统、网络通信、Netty

hdfs、kafka、zookeeper的源码，二次开发，源码级别解决问题的能力

所有大数据项目和系统，数据采集，主要是讲解了一套方案，还暂时没有落地开发，需要点时间，后续落地开发和实战，1个月，1月份~三四月份，是不是讲解一套大大数据面试突击系列的课程

各种新技术，数据仓库，flink实时数据，kylin数据分析，druid，ELK，数据湖，clickhouse，数据中台，整个体系过一遍，面试里常见的一些问题给大家分析一下，讲解一些项目案例，架构设计思路，生产环境的一些优化技巧

大项目的思路，去继续讲解很多个大型项目

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/206\_回顾一下follower主动跟leader建立网络连接的过程/笔记.docx**

《206\_回顾一下follower主动跟leader建立网络连接的过程》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/203\_客户端关闭网络连接之后，会如何重新尝试连接？/笔记.docx**

《203\_客户端关闭网络连接之后，会如何重新尝试连接？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/214\_如果剩下两个follower，可以重新选举出leader吗？/笔记.docx**

《214\_如果剩下两个follower，可以重新选举出leader吗？》

3台机器，最多可以宕机1台，2台机器，占据3台机器中的大多数，2台机器必然会选举出来一个人当leader，3 / 2 + 1 = 2

zk集群部署最好是奇数台，偶数

4台机器，最多也只能是宕机1台机器，4 / 2 + 1 = 3，最多也只能宕机一台机子，必须有3台机器留下来才可以选举出来一个leader

5台机器，quorum = 5 / 2 + 1 = 3，可以宕机2台

6台机器，quorum = 6 / 2 + 1 = 4，可以宕机2台

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/207\_一个follower宕机之后，leader是如何感知到的？/笔记.docx**

《207\_一个follower宕机之后，leader是如何感知到的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/208\_感知到follower宕机之后，leader会如何进行处理？/笔记.docx**

《208\_感知到follower宕机之后，leader会如何进行处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/212\_follower感知到leader挂掉之后，会如何进行处理？/笔记.docx**

《212\_follower感知到leader挂掉之后，会如何进行处理？》

如果说follower崩溃，其实影响倒还不是太大，基本还能运转的，所有客户端可以重新连接其他人，读写请求继续，读请求没有影响，写请求没法那么快达到过半follower返回ack过来，监听器还是可以正常的区施加

如果说是leader崩溃，对zk短时间内是灾难性的

其他的follower，他会把所有客户端的连接全部断开，对于客户端而言的话，他会短时间内频繁报错，他内部会不停的尝试去连接其他的zk机器，但是呢，此时是没有办法重新建立连接的

此时集群就需要重新进行一个leader选举

各个follower把状态设置为LOOKING，重新寻找或者选举一个leader出来

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/215\_选举出新的leader之后，从集群到客户端是如何恢复的？/笔记.docx**

《215\_选举出新的leader之后，从集群到客户端是如何恢复的？》

follower会跟新的leader建立连接，LearnerHandler跟他进行通信，还会进行数据恢复，另外，对外提供服务，所有的客户端会自动进行重连的，他会找到一台机器进行连接，读写请求继续执行，另外一个监听器重新施加

之前宕机的那台leader再次重启，他此时就会发现已经有leader了，此时他就是一个follower，跟leader建立长连接，然后进行数据的恢复，接着的话呢，整个集群就瞬间就可以恢复运作了，写入的数据

每次都是follower的ack数量 > 1，必须都得是两个follower都返回ack，才可以执行commit，即使你的follower没有宕机，也会导致你的性能有点问题的，3台机器，leader + 1个follower写poposal成功了，1个follower返回了ack

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/209\_挂掉一个follower，对leader的2PC过半写机制有影响吗？/笔记.docx**

《209\_挂掉一个follower，对leader的2PC过半写机制有影响吗？》

瞬间挂掉了一个follower之后，你觉得是不是会有一定的影响呢？

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/210\_重启挂掉的follower之后，如何进行数据恢复？/笔记.docx**

《210\_重启挂掉的follower之后，如何进行数据恢复？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/201\_在客户端源码中找一找，如何感知到zk服务器的宕机？/笔记.docx**

《201\_在客户端源码中找一找，如何感知到zk服务器的宕机？》

客户端正常的发送请求、定时发送ping，发送请求的时候，如果zk宕机了，此时socket.write()一定会抛出IOException，此时你就可以认为是zk服务器可能是宕机，也可能是网络出现了问题，反正就是跟这台zk服务器失联了

try catch，捕获到这个IOException，此时就可以自然的感知zk服务器出问题了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/205\_follower宕机过后，客户端需要做哪些容错处理？/笔记.docx**

《205\_follower宕机过后，客户端需要做哪些容错处理？》

你一下子会感知到有异常，所以对zk的所有操作，都必须要进行异常的try catch和处理，默认监听器，你就需要自己进行一个定时的重试操作，休眠个几百毫秒，再次尝试执行create这样的操作

默认的监听器还是挺重要的，你会感知到连接断开了，此时需要进行一个等待，再等待连接重新建立好，你此时就应该重新注册所有的监听器

一旦重新连接了其他的zk服务器之后

你的重试操作就会成功了，另外的你一旦默认监听器感知到说建立了新的连接，此时要重新去施加所有的监听器

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/217\_吃透这个ZooKeeper课程，你就是国内顶尖ZK高手！/笔记.docx**

《217\_吃透这个ZooKeeper课程，你就是国内顶尖ZK高手！》

玩儿大数据，放在hdfs和kafka前面来玩儿的，zk是绝对独立的，不依赖什么东西的主从架构的系统，hdfs、kafka、hbase，很多后续的一些技术，都是基于和依赖于zk来实现的，要明白这一点

Java的基础架构，分布式中间件系统，也应该搞一下zk，canal之类的，很多系统都是基于zk来实现的

网上讲解zk的书籍，两本，很多东西不到源码层面，可能看的还挺吃力的，书都是好书，都是那些写书的技术高手自己咀嚼透彻了zk之后，自己写出来的，新手去看，未必能很容易的看懂

zk的初步一些讲解，浅薄

99%的工程师，对zk仅仅是一个了解的程度

对zk从原理，到使用，到集群运维，到框架，到使用场景和案例，到底层的源码，剖析的非常透彻了，ZK这块的顶尖高手了，zk课的最后一周，还想再加一周的课

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/200\_重新回顾客户端与zk服务端建立长连接的过程/笔记.docx**

《200\_重新回顾客户端与zk服务端建立长连接的过程》

zk源码的最后一周，zk课程的最后一周，hdfs、kafka、zookeeper，三门最核心的技术，讲解的非常的透彻，spark课程，也是讲解的非常的细致，多做一些大数据的项目

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/213\_再次回顾leader选举过程的源码实现/笔记.docx**

《213\_再次回顾leader选举过程的源码实现》

最终会看，哪个follower接收到的zxid更大，或者zxid一样大，myid，两个人一定会投票给某一个人，两轮投票，结果出来了，两台机器都投票给一个人作为leader，2 > 3的quorum，3 / 2，1

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/200~218资料/211\_如果Leader挂掉的话，各个follower会如何感知到？/笔记.docx**

《211\_如果Leader挂掉的话，各个follower会如何感知到？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/175\_follower是如何执行commit操作的？/笔记.docx**

《175\_follower是如何执行commit操作的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/161\_看一下ZooKeeper客户端中的create方法的源码入口/笔记.docx**

《161\_看一下ZooKeeper客户端中的create方法的源码入口》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/171\_看一下Follower初始化的请求处理链条是什么？/笔记.docx**

《171\_看一下Follower初始化的请求处理链条是什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/163\_初步看一下create请求的预处理环节会做什么？/笔记.docx**

《163\_初步看一下create请求的预处理环节会做什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/164\_Leader是如何初始化自己的请求处理链条的？/笔记.docx**

《164\_Leader是如何初始化自己的请求处理链条的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/172\_完成flush之后如何对proposal请求返回ack给leader？/笔记.docx**

《172\_完成flush之后如何对proposal请求返回ack给leader？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/177\_zk如何创建数据快照写入到磁盘上去？/笔记.docx**

《177\_zk如何创建数据快照写入到磁盘上去？》

切换一个新的文件写事务日志，同时创建一份数据快照出来

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/170\_follower写多少条数据到事务日志文件之后，会执行flush？/笔记.docx**

《170\_follower写多少条数据到事务日志文件之后，会执行flush？》

follower要写入1000条事务日志之后，才能进行flush以及后续的处理，比如说返回ack给leader，会尽快的用while循环，把积压在内存队列里排队的proposal的请求全部都尽快的写入到磁盘上的事务日志里去

一旦将队列里积压的proposal都写入事务日志了，此时就可以执行flush了

或者是如果你连续写入1000条数到事务日志里去，此时也会强制性执行flush以及后续的操作

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/174\_commit时如何将数据写入内存数据库中去？/笔记.docx**

《174\_commit时如何将数据写入内存数据库中去？》

zk一定不是强一致的

sync方法，他必然是要等待所有的follower都执行了commit之后，才会返回，此时会确保你说，你无论是从leader还是follower，此时都可以查到一样的数据，如果你要保证强一致的话

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/168\_如何基于SyncProcessor写本地磁盘事务日志？/笔记.docx**

《168\_如何基于SyncProcessor写本地磁盘事务日志？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/165\_初步看一下Proposal处理器都干了一些什么事儿？/笔记.docx**

《165\_初步看一下Proposal处理器都干了一些什么事儿？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/169\_初步看一下Follower是如何接收proposal请求的？/笔记.docx**

《169\_初步看一下Follower是如何接收proposal请求的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/173\_过半follower返回ack之后，如何触发commit操作？/笔记.docx**

《173\_过半follower返回ack之后，如何触发commit操作？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/179\_本周源码剖析内容的一个总结/笔记.docx**

《179\_本周源码剖析内容的一个总结》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/167\_如何基于Leader业务逻辑将proposal同步给follower？/笔记.docx**

《167\_如何基于Leader业务逻辑将proposal同步给follower？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/166\_在进行2PC同步之前，是如何先将请求进行commit等待排队的？/笔记.docx**

《166\_在进行2PC同步之前，是如何先将请求进行commit等待排队的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/162\_再看一次SendThread是如何将排队的请求发送到服务端的？/笔记.docx**

《162\_再看一次SendThread是如何将排队的请求发送到服务端的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/176\_如何基于ZooKeeper的sync实现强一致等待？/笔记.docx**

《176\_如何基于ZooKeeper的sync实现强一致等待？》

create，set了一个path

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/178\_zk是如何定期清理磁盘上不需要使用的文件的？/笔记.docx**

《178\_zk是如何定期清理磁盘上不需要使用的文件的？》

1~10000条数据，在这个事务日志文件01里

切换事务日志文件成为02，以后的事务日志都写到事务日志文件02里去，从10001条开始

同时写一份数据快照，包含了1~10000条事务对应的一份内存文件目录树，数据快照01，包含了1~10000条事务更新后对应的文件目录树

10001~20000条数据，在这个事务日志文件02里

切换事务日志文件成为03，以后的事务日志都写到事务日志文件03里去，从20001条开始

同时写一份数据快照，包含了1~20000条事务对应的一份内存文件目录树，数据快照02，包含了1~20000条事务更新后对应的文件目录树

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/160~179资料/160\_先来回顾一下目前已经讲完的ZooKeeper核心架构原理/笔记.docx**

《160\_先来回顾一下目前已经讲完的ZooKeeper核心架构原理》

三块东西，create、set、del、get、watcher

zk高可用：follower突然宕机了，如何跟leader进行数据恢复；leader突然宕机，follower重新发起投票，选举leader；重启zookeeper，他会从磁盘上恢复数据到内存里去，这个数据恢复的过程

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/029\_你知道一个znode中最多能存储多大数据量吗？/笔记.docx**

《029\_你知道一个znode中最多能存储多大数据量吗？》

jute.maxbuffer

肯定会不停的在zk里写数据，znode，一个znode最多可以存储多少数据呢？1mb，1048575，bytes，换算成mb，1mb

一般来说建议，不要在znode中存储的数据量过大，几百个字节，几百kb，就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/033\_跟leader自己相关的两个参数的说明/笔记.docx**

《033\_跟leader自己相关的两个参数的说明》

leaderServers：yes，leader是否接受客户端的连接，写请求由follower转发给leader，leader主要接受follower的转发写请求进行处理

cnxTimeout：5000，在进行leader选举的时候，各个机器会基于3888那个端口建立TCP连接，在这个过程中建立TCP连接的超时时间

所有zk核心的参数都给大家讲完了

在启动zk集群之前，有哪些参数是必须要设置的，全都在zoo\_sample.cfg，把zk集群的信息配置进去，server.1，server.2，给每台机器配置dataDir，dataLogDir，开启自动清理数据文件的参数

zkServer.sh命令，分别启动leader和follower等多个节点

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/027\_ZooKeeper是什么时候进行数据快照的？相关的参数是什么？/笔记.docx**

《027\_ZooKeeper是什么时候进行数据快照的？相关的参数是什么？》

zk里的数据分成两份：一份是在磁盘上的事务日志，一份是在内存里的数据结构，理论上两份数据是一致的，即使是有follower宕机，也是内存里的数据丢失了，但是磁盘上的事务日志都是存在的

即使有的follower没收到事务日志就宕机了，也可以在启动之后找leader去同步数据

此时zk比如说在磁盘里有一份事务日志了，此时他启动之后要重建内存里的数据，如何重建呢？难道说把事务日志进行回放？一条一条重新执行每条事务日志到内存里去吗？

zk是有一个数据快照的机制

每次执行一定的事务之后，就会把内存里的数据快照存储到dataDir这个目录中去，作为zk当前的一个数据快照

现在在磁盘事务日志里有1000个事务，然后把1000个事务对应的内存数据写入到dataDir里作为一个数据快照，继续此时事务日志里有1032个事务，此时zk重启，他可以直接把包含1000个事务的快照直接加载到内存里来

然后把1000之后的32个事务，1001~1032的事务，在内存里回放一遍，就可以在内存里恢复出来重启之前的数据了

snapCount：100000，默认是10万个事务，存储一次快照

即使没到10万事务，就重启了，此时没有快照，10万个事务以内，不需要快照，因为直接读取事务日志，回放到内存就重建内存数据了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/030\_运行时的Leader与Follower是通过哪两个端口进行通信的？/笔记.docx**

《030\_运行时的Leader与Follower是通过哪两个端口进行通信的？》

server.1=zk01:2888:3888

一般来说每台机器的3888端口，是用来在集群恢复模式的时候进行leader选举投票的，也就是说所有的机器之间进行选举投票的时候就是基于3888端口来的

2888的端口，是用来进行leader和follower之间进行数据同步和运行时通信的

paxos算法，跟zk之间的关联关系

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/025\_ZooKeeper一般必须要配置的核心参数说明/笔记.docx**

《025\_ZooKeeper一般必须要配置的核心参数说明》

tickTime

dataDir

dataLogDir

机器的配置，zk在机器上如何部署和配置，zk的jvm参数如何配置，如何给zk分配他需要的内存资源

正式的启动zk之前，其实还需要其他的一些东西

对zk的一些参数进行配置

tickTime：zk里的最小时间单位，2000毫秒，2s

其他的一些参数就会以这个tickTime为基准来进行设置，比如有的参数就是tickTime \* 2

dataDir：主要是放zk里的数据快照，剖析zk的源码的时候

zk里会存储很多的数据，内存里有一份快照，在磁盘里其实也会有一份数据的快照，zk停机了，重启，才能恢复之前的数据

dataLogDir：写数据，2PC，proposal，每台机器都会写入一个本地磁盘的事务日志，主要是放一些日志数据

SSD固态硬盘，读写速度非常快，dataLogDir，事务日志磁盘写，是对zk的写性能和写并发的影响是很大的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/022\_ZooKeeper集群是如何部署和启动的？/笔记.docx**

《022\_ZooKeeper集群是如何部署和启动的？》

zk集群非常简单，准备三台机器，虚拟机也可以，每台机器上部署一个ZooKeeper的进程就可以了

每台机器上都得搞一个zookeeper，每台机器上的配置文件都是一样的，但是要加点东西，每个zk节点都得感知到集群中当前应该是有几台机器

server.1=zookeeper01:2888:3888

server.2=zookeeper02:2888:3888

server.3=zookeeper03:2888:3888

zk里，每个节点，都得有一个id，需要在你指定的datadir里创建一个myid的文件，写在里面，自己编号，123

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/023\_ZooKeeper集群部署需要用什么样配置的机器？/笔记.docx**

《023\_ZooKeeper集群部署需要用什么样配置的机器？》

详细分析过kafka生产集群的部署，这个东西没多少技术含量，一般来说肯定是用一些比较高配置的机器去部署中间件系统

4核8G的虚拟机

8核16G，16核32G，高配置虚拟机最好了，SSD固态硬盘

4核8G的机器，一般来说，每秒并发搞到1000是可以的

8核16G的机器，每秒并发搞到几千是可以的

16核32G的机器，每秒并发搞到上万或者是几万都是有可能的

3台机器，1个leader，2个follower，leader主要是写，每秒抗几万并发写入是可以的；leader+follower，读，每秒抗个5万~10万的读是没有问题的，每秒十多万的QPS都可以抗住了

写QPS无法线性的扩展

读QPS还可以，Observer节点，后面给大家来讲，修改一些配置，强制要求他的角色是Observer，不能是leader或者follower就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/021\_体验一把ZooKeeper单机模式是如何启动的？/笔记.docx**

《021\_体验一把ZooKeeper单机模式是如何启动的？》

已经基本上搞完了，zk他是原理，zk属于基础架构类别的系统，中间件系统，部署的机器少，但是你别用太普通的配置

zk，kafka，hbase，hdfs

虚拟机，3台机器，8核16g，16核32g，ssd固态硬盘

三台机器的小集群抗每秒十几万的并发是没有问题的，zk集群，基本上公司都够用了

zookeeper用的比较多的，一般都是3.4.5这个版本，3.4.x版本都差不多，3.5.x版本，也是类似的

windows电脑，弄一台虚拟机，里面装linux操作系统，在虚拟机里就可以跟我一样儿玩

tar.gz

znode为核心的类似文件系统的一套数据结构

节点，层级关系，类似于很多的目录和子目录，zkCli.sh，命令行，类似于linux的文件系统的命令，ls /

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/028\_一台机器上最多能启动多少个ZooKeeper客户端？/笔记.docx**

《028\_一台机器上最多能启动多少个ZooKeeper客户端？》

maxClientCnxns

在客户端上可能会运行一个Java开发的系统，去使用zk，比如说Kafka（Scala，也是JVM类的语言），HBase，Canal，HDFS，Canal比如部署在一台机器上，此时他会去使用zk，他就是一个zk的客户端

会使用zk的Java API，去建立一个zk客户端，zk客户端负责跟zk servers进行连接和通信

一台机器上，我们可以创建多少个zk客户端？也就是说可以跟zk servers建立多少个连接呢？是有限制的，默认来说60，3.4.0以前是10个

如果我们自己开发一个系统去使用zk的话，一台机器上，你一般是就用一个zk客户端去跟zk servers进行交互就可以了，不要无限制的搞很多的zk客户端，会连接不上去的，他zk servers最多只能允许你的一台机器跟他建立60个连接

每次请求都创建一个zk客户端，跟他建立连接，进行通信，再销毁zk客户端，如果并发有很多个请求一起连接zk，此时会导致一台机器上有很多zk客户端，会被zk servers拒绝的，要明白这一点

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/032\_在2PC阶段中写入磁盘的事务日志有没有丢失的风险？/笔记.docx**

《032\_在2PC阶段中写入磁盘的事务日志有没有丢失的风险？》

事务进行commit提交的时候，有没有日志丢失的风险？默认情况下，在2PC阶段的第一个阶段里，各个机器把事务日志写入磁盘，此时一般进入os cache的，没有直接进入物理磁盘上去

在commit提交的时候一般默认会强制把写的事务fsync到磁盘上去

forceSync：yes

zk在写的时候，其实是会涉及到多台机器写磁盘

写日志，顺序写磁盘，跟随机写内存的性能是差不多的，性能是很高的

os cache，他的性能就更高了

commit的时候，需要fsync到磁盘上去，机器坏了，有可能会丢失部分os cache里没刷入磁盘的数据，如果是leader宕机

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/036\_开启ZooKeeper的JMX端口以及使用JConsole观察JVM内存使用/笔记.docx**

《036\_开启ZooKeeper的JMX端口以及使用JConsole观察JVM内存使用》

-Dcom.sun.management.jmxremote.port=21811

-Dcom.sun.management.jmxremote.ssl=false

-Dcom.sun.management.jmxremote.authenticate=false

ZOOMAIN

JDK自带了一个可视化的JVM进程内存分析的工具，JConsole

在你自己的笔记本电脑上就可以打开使用，百度一下，看看就知道了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/034\_ZooKeeper提供给运维人员使用的命令说明/笔记.docx**

《034\_ZooKeeper提供给运维人员使用的命令说明》

echo conf | nc localhost 2181

conf（查看配置）、cons（查看连接）、crst（重置客户端统计）、dump（输出会话）、envi（查看环境）、ruok（检查是否在运行）、stat（查看运行时状态）、srst（重置服务器统计）、wchs（查看watcher信息）、wchc（输出watche详细信息）、wchp（输出watcher，以znode为单位分组）、mntr（输出比stat更详细的）

开发一个可视化的监控平台，可以自己基于他的这些命令的输出绘制出来一些图形就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/035\_基于jstat命令监控和检查ZooKeeper的JVM运行情况/笔记.docx**

《035\_基于jstat命令监控和检查ZooKeeper的JVM运行情况》

jvm，内存使用情况，gc情况

狸猫技术窝，《从0开始带你成为JVM实战高手》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/031\_事务日志和数据快照是如何进行定时清理的？/笔记.docx**

《031\_事务日志和数据快照是如何进行定时清理的？》

不停的zk运行，事务日志会越来越多，不可能是无限多的，切割出来多个事务日志文件，每次你执行一次数据快照，每次都有一个独立的数据快照文件，多个事务日志文件，多个数据快照文件

默认来说没有开启定时清理数据文件的功能

autopurge.purgeInterval=1

autopurge.snapRetainCount=3

让他自己后台默默的自动清理掉多余的事务日志文件和数据快照文件

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/026\_影响Leader与Follower组成集群运行的两个核心参数说明/笔记.docx**

《026\_影响Leader与Follower组成集群运行的两个核心参数说明》

initLimit：zk集群启动的时候，默认值10，10 \* tickTime，20s，leader在启动之后会等待follower跟自己建立连接以及同步数据，最长等待时间是20s，在20s内，follower一般必须要跟leader建立连接

leader就不等follower，直接进入消息广播模式，对外提供服务了

你的zk里存储的数据量比较大了，follower同步数据需要的时间比较长，此时可以调大这个参数

syncLimit：默认值5，5 \* tickTime，10s，leader跟follower之间会进行心跳，如果超过10s没有心跳，leader就把这个follower给踢出去了，认为他已经死掉了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/024\_如何合理设置ZooKeeper的JVM参数以及内存大小？/笔记.docx**

《024\_如何合理设置ZooKeeper的JVM参数以及内存大小？》

你现在手头搞好了几台机器，大内存的机器，zk也都配置好了，准备启动了，能直接启动吗？不能啊，生产环境下，是有一些特殊的东西要改一改的

zk本身是用java写的，是基于JVM虚拟机来运行的，启动的是一个JVM进程，JVM进程里面会执行zk的全部代码

必须得合理的设置一些JVM的参数

三大块东西，必须要合理的调优和设置的，你首先，分配你的各块内存区域的大小，堆内存，栈内存，Metaspace区域的内存，根据你机器上内存的情况去设置

机器如果有16G的内存，堆内存可以分配个10G，栈内存可以分配每个线程的栈是1MB，Metaspace区域可以分配个512MB都可以

另外得设置垃圾回收器，新生代+老年代，ParNew+CMS，如果是大内存机器，不建议这个组合了，G1回收所有的垃圾对象，还得设置一些G1的参数，region的大小，预期的每次GC的停顿时间是多少毫秒，比如100ms

线上JVM运行的时候，频繁gc问题，oom问题，加入一些参数，gc日志要写入哪个目录中，如果发生oom的话，此时要自动dump内存快照出来放哪个目录去

三大块的jvm参数都合理的设置好

当你的zk启动之后，在你的公司开始使用他了，在各种情况下，一些运气高峰期，jstat观察一下他的情况，jvm运行的情况，包括他的新生代的对象增长速率，ygc的频率，老年代增长速率，fgc的频率

gc有问题，得进行gc调优，合理优化jvm参数

另外，如果有监控系统，需要对zk运行时的jvm的情况进行监控，比如说gc频繁，内存使用率

不是在我们的课程里会讲解的，都是一些比较基础的东西，狸猫技术窝，专栏，《从0开始带你成为JVM实战高手》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/021~037资料/037\_如何在集群中加入Observer节点提升读QPS？/笔记.docx**

《037\_如何在集群中加入Observer节点提升读QPS？》

配置一台机器上的zk

peerType=observer

所有机器的配置文件，都要加入一个server.4=zk04:2888:3888:observer

leader仅仅会同步数据给他，但是他不参与leader选举，也不参与过半写机制

看看磁盘空间是否足够，zk集群建立的连接是否过多，升级一下版本，3.4.5够用了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/090\_生产案例：如何基于ZooKeeper实现一个分布式日志中心系统？/笔记.docx**

《090\_生产案例：如何基于ZooKeeper实现一个分布式日志中心系统？》

你需要在一个日志中心可视化工作台里注册你的应用对应的日志名称，此时会自动在kafka中创建topic，一般来说对于同一个系统或者服务，就对自己的日志都用一个名字，就会写入到kafka的一个topic中去

而且此时你还需要对你的日志进行一定的配置，比如说用日志里的哪些字段在ES里构建索引可以方便你后续检索和排查日志？

你可以想一下，比如说你要检索自己服务的日志，可能有多个索引字段，比如有应用里有多个模块，每个模块对应不同的功能，你是不是可以精准搜索某个模块的某个功能的日志？或者是你的系统有多个接口，每个日志得有一个接口id

你可以搜索从一个接口收取一个请求开始，后面连带着这个请求在每个模块的每个功能点里流转的日志

或者你需要对一个分布式系统的日志进行链路搜索，此时你需要有一个链路对应的唯一的请求id，然后一旦搜索这个请求id，此时会根据时间顺序统计出来这个请求在每个服务里的日志

这些索引建立的细节，都需要你在创建日志的时候指定好

同时创建topic的时候，会在zk里也写入目前待收集的topic列表

/log-center/topics/#{topic}

日志中心系统slave集群化部署，去zk注册自己

/log-center/slaves/#{host}

master节点可以知道有哪些topic待收集，有哪些slave节点

master节点会自动对slave进行分组，一个分组是多个slave，然后他会在group中写入slave列表，同时会给每个slave分组都分配一批topics，刚开始肯定是每个分组都是均匀分配一批topics

/log-center/topics/#{topic}

/log-center/slaves/#{host}

/log-center/slave-groups/#{group01}/#{slave01}

/log-center/slave-groups/#{group01}/#{slave02}

/log-center/slave-groups/#{group01}/topics/#{topic}

/log-center/slave-groups/#{group02}/#{slave03}

/log-center/slave-groups/#{group02}/#{slave04}

/log-center/slave-groups/#{group02}/topics/#{topic}

Slave而言，他需要去监听，如果说 有变化，此时他就需要去得到通知，每个Slave需要知道自己属于哪个group的；，感知自己的group被分配的topic的变化

为啥slave要分组？

因为topic一般都是有多个partition的，你一个分组搞多个slave，多个slave作为一个consumer group他就是有多个consumer，他会自动对每个topic都并行消费多个partition的数据

master节点在zk中的每个slave分组下写入这个分组要负责消费的topic列表，slave节点会监听自己所在组负责消费的topic列表，就会负责对这个topic列表进行消费

接着正常来说，你的服务就会按照设置好的格式开始打日志了，日志进入对应的Topic，每个slave组就会负责消费一批topic了，而且是一个组内部的多个slave并行消费每个topic的多个partition，然后根据一定的业务逻辑处理之后写入到es中去，具体如何处理，就看你的日志中心提供哪些功能了，在es中建立好索引

接着你就可以根据你的需求随时调阅和查询索引了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/081\_Curator实现的ZK读写锁的代码示例以及初步介绍/笔记.docx**

《081\_再读Curator框架中基于ZK实现的分布式读写锁的源码》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/088\_如何基于ZK实现分布式系统的集群扩容与宕机自动感知机制？/笔记.docx**

《088\_如何基于ZK实现分布式系统的集群扩容与宕机自动感知机制？》

集群中加入一台机器，自动在zk中写入一个znode，临时节点，一旦节点关闭或者宕机，临时节点自动消失。由集群Master控制节点监听zk目录子节点变化，自动感知集群中节点的上线和下线

分布式系统部署的集群，里面很多机器

每一台机器上部署的节点启动之后自动在zk里创建一个属于自己的节点

/nodes/#{host01}

/nodes/#{host02}

Master节点，监听/nodes节点的子节点的变动，一旦有子节点新增或者删除，就说明集群里有机器上线或者下线了。如果有机器宕机了，都是临时节点，消失，也就是删除，此时你收到通知，就知道有机器故障宕机，也有可能重启的时候正常下线了

集群里的元数据存储，无非就是对znode进行crud

监听与回调，对你需要感知的znode加监听器，回调通知你，你就知道了

master选举，无非也都是说创建一些临时顺序节点，排在第一位的就是leader

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/095\_基于ZooKeeper与Elasticsearch的日志中心课程的预告/笔记.docx**

《095\_基于ZooKeeper与Elasticsearch的日志中心课程的预告》

日志中心，对于大规模的分布式业务系统，必须的，日志中心、监控中心、服务治理平台，必备的，日志中心检索日志帮助你处理问题，业务metrics、JVM、机器资源、服务宕机、程序异常进行监控，自动进行报警

服务治理平台，可以看到每个服务当前的状态，在各种环境里有哪些机器，各个机器的状态，每台机器的QPS，接口性能，可用性（99.6%），去进行一些服务降级、灰度发布，还有很多大规模服务管理的一些功能

Java架构课程

日志中心一般是基于ES，自己手撸，自己做一套，分布式日志中心，基于Elasticsearch+ZooKeeper，分布式日志中心出来

监控中心，Open-Falcon

服务治理平台，一般来说我们可以自己去自研一套出来

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/093\_生产案例：如何而基于ZooKeeper实现日志中心的自动负载均衡？/笔记.docx**

《093\_生产案例：如何而基于ZooKeeper实现日志中心的自动负载均衡？》

如果有新的topics被注册了呢？

/log-center/topics/#{topic}

/log-center/slaves/#{host}

/log-center/slave-groups/#{group01}/#{slave01}

/log-center/slave-groups/#{group01}/#{slave02}

/log-center/slave-groups/#{group01}/topics/#{topic}/workload/#{partition}

/log-center/slave-groups/#{group01}/topics/#{topic}/offset/#{partiton}

/log-center/slave-groups/#{group02}/#{slave03}

/log-center/slave-groups/#{group02}/#{slave04}

/log-center/slave-groups/#{group02}/topics/#{topic}/workload/#{partition}

/log-center/slave-groups/#{group02}/topics/#{topic}/offset/#{partiton}

如果他把你的group消费的所有topic的负载都获取出来进行汇总，就可以达到你的group整体的负载。他还可以仅仅针对每个topic进行汇总，就可以计算出来每个topic数据速率，此时就可以做出精准的负载均衡

group01：有3个topic，每个topic大概每分钟有1000条日志，每分钟3000条日志

group02：有2个topic，每个topic大概每分钟有300条日志，每分钟600条日志

group01：有2个topic，每个topic大概每分钟有1000条日志，每分钟2000条日志

group02：有3个topic，两个topic大概每分钟有300条日志，1个topic每分钟1000条，每分钟1600条日志

每个salve对workload的写，是覆盖写，每个topic的每个partition，仅仅保留最近一分钟的工作负载，保留的是最近一小时内每分钟的工作负载

此时当然会自动把topics分配给负载较低的slave group

如果在运行过程中发现有的slave group负载高，有的slave group负载低呢？此时当然master节点就需要自动定时检查各个slave group的负载，然后自动进行负载均衡，自动把一些topic分配给他group去消费

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/086\_Curator读写锁目前依然存在的羊群效应以及优化思路/笔记.docx**

《086\_Curator读写锁目前依然存在的羊群效应以及优化思路》

[读锁，读锁，读锁]

一旦说排在第一位的写锁释放了，会导致列表中可能大量的读锁都会羊群效应被惊醒，然后再次去计算第一个写锁的位置，比较自己是否在第一个写锁的前面，羊群效应。面试，聊分布式锁，聊zk的读写锁，Curator框架的源码，羊群效应

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/084\_Curator读写锁在写锁释放后是如何继续加锁的？/笔记.docx**

《084\_Curator读写锁在写锁释放后是如何继续加锁的？》

[写锁, 读锁, 读锁]

假设是另外一个锁列表：[读锁，读锁，写锁，读锁，读锁，写锁]

如何判断自己有没有获取到锁，如果没有获取到锁对哪个节点进行监听

写锁：自己如果是第一个就获取了写锁；如果没有获取到锁对前一个节点加监听器

读锁：读锁的位置排在第一个写锁的前面，就可以获取读锁；如果没有获取到读锁，监听第一个写锁

4个读锁监听了第一个写锁，第二个写锁也监听了第一个写锁

此时第一个写锁释放锁了之后，4个读锁和第二个写锁都会收到通知

原先排在第二位的写锁，此时成了排在第一位的写锁，他会发现自己的index < maxLeases，所以此时他发现自己排在第一位，就可以获取写锁了；此时有4个读锁发现自己还是在第一个写锁后面，获取读锁都失败

4把读锁都会监听此时的第一个写锁

此时他们计算出来，第一个写锁的位置是2，此时对于前面2个读锁，会发现自己排在第一个写锁前面，获取读锁成功了，后2个读锁发现自己再第一个写锁之后，获取读锁失败，继续等待，会监听在列表中位置=2的第一个写锁

前2个读锁都释放之后，位置=2的写锁排在第一位，获取写锁

写锁释放，后两个读锁获得通知，第一个写锁的位置是Integer.MAX\_VALUE

最后两个读锁都加锁成功

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/089\_生产案例：基于Elasticsearch的日志中心架构设计/笔记.docx**

《089\_生产案例：基于Elasticsearch的日志中心架构设计》

业务系统部署很多机器，日志过于分散，需要统一收集到ES中进行日志存储，对外暴露可视化工作台随时方便检索日志

日志中心系统需要集群部署，不同的应用有日志客户端，需要自动把日志异步化写入到Kafka中去，对应的是不同的Topic，日志中心系统分为Master-Slave架构，Slave负责从Kafka中消费某些Topic里的日志写入ES，Master负责给Slave分配对应要消费的Topics

一个服务的日志对应Kafka中的一个Topic

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/092\_生产案例：如何基于ZooKeeper实现日志中心的动态扩容和故障转移？/笔记.docx**

《092\_生产案例：如何基于ZooKeeper实现日志中心的动态扩容和故障转移？》

/log-center/topics/#{topic}

/log-center/slaves/#{host}

/log-center/slave-groups/#{group01}/#{slave01}

/log-center/slave-groups/#{group01}/#{slave02}

/log-center/slave-groups/#{group01}/topics/#{topic}/workload/#{partition}

/log-center/slave-groups/#{group01}/topics/#{topic}/offset/#{partiton}

/log-center/slave-groups/#{group02}/#{slave03}

/log-center/slave-groups/#{group02}/#{slave04}

/log-center/slave-groups/#{group02}/topics/#{topic}/workload/#{partition}

/log-center/slave-groups/#{group02}/topics/#{topic}/offset/#{partiton}

如果说某个slave宕机，或者是重启正常下线了，此时他的临时节点一定就没有了，master会感知到某个机器的下线，此时那个slave负责消费的partition会自动转移给这个slave group中的其他slave

每个slave group就是一个consumer group，一旦某个consumer宕机了之后，kafka会自动把他的partition分配给group中其他的consumer进行消费和处理

此时group中其他的slave接管了他的工作，会自动接着他之前消费的offset继续去进行消费和处理，继续去记录这些topic partition的运行中的元数据，速率和offset

此时会导致slave group中的其他slave的负载会增加的

3个salve，每个slave有一定工作负载，突然挂了一个slave，另外两个salve工作负载会增加，每秒钟需要消费和处理的日志的数量会增加，CPU等资源会更加的繁忙

因此这个时候，master需要进行负载均衡，他需要把那个slave group的某些topic转移到其他slave group上去，挑那些负载较低的slave group就可以，因为你是知道每个slave group的具体负载的

此时master可以指引这些slave group对某些topics停止消费，同时最后一次将对topics消费的offset写入zk中，然后部分topics迁移分配给其他负载较低的slave group

此时原来的slave group少了一些topics，自动监听到topics列表变动，然后取消对一些topic的消费，同时有的slave group此时会增加一些topics，他们会从这些topics指定的offset地方开始消费，继续处理日志

如果是扩容多部署了一些slave，此时master会自动感知到多了slave，然后就会根据各个slave group的负载，给负载较高的slave group加入一些新的slave，让每个slave处理更少的partiton，负载降低

或者是干脆直接就用新的slave成立一个新的slave group，然后直接把其他负载高的group负责的一些topics分配给新的slave group即可

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/094\_生产案例：日中中心Master节点的双机部署以及自动HA切换/笔记.docx**

《094\_生产案例：日志中心Master节点的双机部署以及自动HA切换》

创建一个临时节点

/log-center/master/#{临时顺序节点}

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/099\_如何基于ZooKeeper实现IM系统课程中的iplist服务？/笔记.docx**

《099\_如何基于ZooKeeper实现IM系统课程中的iplist服务？》

接入系统，请求分发系统，消息投递系统，他们都是可以在启动的时候在zk里注册自己

/im-system/tcp-gateway/#{host}

/im-system/dispatcher/#{host}

/im-system/delivery/#{host}

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/096\_生产案例：一个数据同步系统的整体架构设计/笔记.docx**

《096\_生产案例：一个数据同步系统的整体架构设计》

基于zk，集群元数据存储，监听和回调，Leader选举，这三个功能一般来说是分布式中间件系统，或者是分布式大数据系统，基本都是必备，必须的

数据源，Collector集群，Kafka，Store集群，任务管理系统，MySQL，ZooKeeper

你需要配置一些数据同步任务，比如说从ES同步数据到HBase，或者是从MySQL同步数据到ES，等等，还有当前的数据总量，每天的数据增量，等等，具体的一些配置都是存放在ZK里的

然后任务管理系统节点负责监听ZK里数据同步任务的变化，如果有新的任务，就会给一些Collector下发同步任务，让这些Collector从目标数据源去同步数据，写入到Kafka中去，然后给一些Store下发存储任务，让他们负责从Kafka中对应的Topic中获取数据然后写入目标存储中去

在这个过程中，Collector对数据获取的进度都会写入到zk里去，Store将数据存储的进度也会写入到zk里去

/data-sync-system/data-sync-tasks/#{data-sync-task}

/data-sync-system/collectors/#{collector}/data-sync-tasks/#{data-sync-task}

/data-sync-system/stores/#{store}/data-store-tasks/#{data-store-task}

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/078\_如果不使用基于ZK的配置中心，业务系统运行中会有什么痛点？/笔记.docx**

《078\_如果不使用基于ZK的配置中心，业务系统运行中会有什么痛点？》

配置基本上都是存放在系统本地磁盘上的配置文件里，系统部署的时候，配置文件会跟着一起部署到线上机器的磁盘中去

库存系统，他有一套自己的配置，此时都是基于配置文件来存放，需要集群化部署，部署10台机器，10台机器的每台机器中的磁盘上都会有一份你的配置文件，假设你对磁盘配置文件的读取策略，定时热加载

10台机器，每台机器都会每隔1分钟去磁盘文件里加载最新的配置信息

假设你要修改一个配置，打开一个手动降级开关，此时你就得自己登录到10台机器上去，每台机器都找到那个配置文件修改里面的手动降级开关的配置，修改10台机器的配置，等待10台机器自动热加载配置文件里最新的配置

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/087\_分布式系统在集群监控与管理上的常见需求和场景/笔记.docx**

《087\_分布式系统在集群监控与管理上的常见需求和场景》

四大块：集群运行元数据、自定义metrics、JVM、机器（CPU、IO、内存、磁盘）

管理与监控分开，基于ZK进行集群元数据存储从而管理一个分布式集群，基于特定的监控系统（zabbix，open-falcon，Agent部署模式）进行监控，上报metrics，自动监控jvm和机器

分布式系统的集群：集群里的机器进行管理，元数据，基于监听和回调，管理集群里的各个机器的工作任务，管控节点（Master）的HA主备切换（类似于Leader选举的效果）

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/098\_基于ZooKeeper实现一个通用数据同步系统的课程预告/笔记.docx**

《098\_基于ZooKeeper实现一个通用数据同步系统的课程预告》

在我们的Java架构课程里，在比较靠后的地方，在讲解ES的一些项目实战的时候，我会带着大家来做一套分布式数据同步系统，必然会涉及到MySQL -> ES，用一些开源的组件，其实不是特别的好

MySQL -> MySQL，MySQL -> ES

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/085\_Curator读写锁是如何针对羊群效应进行优化的？/笔记.docx**

《085\_Curator读写锁是如何针对羊群效应进行优化的？》

[写锁，写锁，读锁，读锁，写锁，读锁，读锁]

不管是写锁还是读锁，大家都监听排在第一位的锁，等待他释放掉，一旦一个锁释放了之后，此时后续其他所有的锁都会得到通知，羊群效应，养了几十头羊，突然在他们羊群里扔一个石子儿，羊群就惊了，一起动了起来

[锁，锁，锁，锁，锁，锁]

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/076\_不使用HDFS等源码讲解ZooKeeper使用方式的原因说明/笔记.docx**

《076\_不使用HDFS等源码讲解ZooKeeper使用方式的原因说明》

zk server架构原理，zk集群部署和运维，zk client使用以及curator框架源码，zk如何运行的、zk集群如何生产级部署、zk是如何来用的以及怎么用，搞明白了，第一，你自己已经会用zk了，第二，你看别的开源项目使用zk的代码，绝对可以看懂了

hdfs、kafka、canal本身自己的源码都非常的复杂，从凌乱的源码中找到对zk的使用，很麻烦，没必要，在hdfs之前先讲zk，在读hdfs、kafka和canal的源码过程中，就顺带把如何使用zk讲了

curator客户端框架操作zk的方式和原理，curator客户端框架的源码，如何使用原生的zk API去实现各种高级复杂的功能，即使你自己以后在读hdfs、kafka、canal的源码过程中，遇到了对zk的操作和使用

把zk在行业里各种各样的生产级、企业级的使用场景和原理，都给大家介绍一下，工业界中是如何在各种企业级场景下使用zk的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/079\_一个经典的面试问题：你为什么要使用分布式锁？/笔记.docx**

《079\_一个经典的面试问题：你为什么要使用分布式锁？》

ZK的第二个工业级使用场景，分布式锁

Java架构课里也深入分析了分布式锁这块的场景，此时就有人出去面试说自己对分布式锁很精通，为什么你不用数据库的行锁

如果说你有分布式业务系统里的多个子系统，订单系统和库存系统，都要更新数据库里的一行数据，此时没有必要使用分布式锁，正常更新，数据库自己会行锁的机制保证说更新的时候先后顺序

比如说有时候你是有一个较为复杂的业务场景

znode：/locks/lock\_1

znode：/locks/lock\_2

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/100\_本周课程总结以及下周课程预告/笔记.docx**

《100\_本周课程总结以及下周课程预告：ZK在工业界的使用案例》

用一些案例来演示和分析了一下，zk在各种分布式系统、配置中心、分布式锁，他一般是怎么来用的，听一听思路

HDFS、HBase、Kafka、Dubbo、Canal，我们不讲源码，就讲ZK在各种工业级开源项目里的广泛的运用，架构原理和运用方案

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/083\_Curator读写锁是如何实现已有写锁之后无法加读锁的？/笔记.docx**

《083\_Curator读写锁是如何实现已有写锁之后无法加读锁的？》

/locks/lock/\_\_WRITE\_\_{很多临时顺序节点}，排队等待加写锁

此时判断你加写锁是否成功，看一下你在/locks/lock下面的位置，如果你的写锁临时顺序节点是在/locks/lock下面的第一个，位置=0，maxLeases=1，说明你获取写锁成功了，否则的话你就只能等待

/locks/lock/\_\_READ\_\_{很多临时顺序节点}，加读锁

会去找到排在最前面的写锁，如果发现排在位置=0有一个写锁，此时获取读锁就一定失败了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/076~100资料/097\_生产案例：数据同步系统的核心工作组件HA方案/笔记.docx**

《097\_生产案例：数据同步系统的核心工作组件HA方案》

/data-sync-system/data-sync-tasks/#{data-sync-task}/#{task-collector}

/data-sync-system/data-sync-tasks/#{data-sync-task}/#{collect-metadata}

/data-sync-system/data-sync-tasks/#{data-sync-task}/#{task-store}

/data-sync-system/data-sync-tasks/#{data-sync-task}/#{store-metadata}

/data-sync-system/collectors/#{collector01}/data-sync-tasks/#{data-sync-task}

/data-sync-system/collectors/#{collector02}/data-sync-tasks/#{data-sync-task}

/data-sync-system/stores/#{store01}/data-store-tasks/#{data-store-task}

/data-sync-system/stores/#{store02}/data-store-tasks/#{data-store-task}

任务负载均衡，自动检查各个collector和store的工作负载，自动进行负载迁移

每个collector和store组件都可能会宕机

任务管理系统在分配任务的时候，直接对每个Collector同时分配一个备用Collector，主Collector会在zk中为自己的任务创建一个临时节点，然后备用Collector会监听这个临时节点

一旦说主Collector挂了，此时备用Collector立马监听到，然后接管对应的工作，并且根据主Collector之前在zk中记录的工作状态，继续同步数据写入kafka中

对于Store而言，则是一组Store共同消费一个Topic，保证自动基于kafka的consumer group机制去负载均衡，任何一个Store挂了，一定有其他的Store会自动接管他的工作，都会自动接着offset消费下去的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/123\_Follower是如何向Leader发起连接请求的？/笔记.docx**

《123\_Follower是如何向Leader发起连接请求的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/134\_ClientCnxn是如何进行初始化的？/笔记.docx**

《134\_ClientCnxn是如何进行初始化的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/124\_Leader跟Follower建立连接之后会干什么？/笔记.docx**

《124\_Leader跟Follower建立连接之后会干什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/136\_客户端如何采用NIO的方式与服务端建立长连接？/笔记.docx**

《136\_客户端如何采用NIO的方式与服务端建立长连接？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/129\_服务端的ServerCnsFactory到底是个什么东西？/笔记.docx**

《129\_服务端的ServerCnxFactory到底是个什么东西？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/122\_Leader是如何启动Follower连接监听器的？/笔记.docx**

《122\_Leader是如何启动Follower连接监听器的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/138\_客户端在连接建立之后会做什么事情？/笔记.docx**

《138\_客户端在连接建立之后会做什么事情？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/127\_Leader是如何处理Follower的注册请求的？/笔记.docx**

《127\_Leader是如何处理Follower的注册请求的？》

反序列化，保存一些数据

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/128\_Follower注册完毕之后的数据同步通信架构/笔记.docx**

《128\_Follower注册完毕之后的数据同步通信架构》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/133\_ZooKeeper初始化的时候会干什么？/笔记.docx**

《133\_ZooKeeper初始化的时候会干什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/131\_服务端基于NIO的ServerSocket是什么时候启动的？/笔记.docx**

《131\_服务端基于NIO的ServerSocket是什么时候启动的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/130\_ServerCnxFactory是什么时候完成初始化的？/笔记.docx**

《130\_ServerCnxnFactory是什么时候完成初始化的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/125\_Jute是什么序列化协议？ZooKeeper内部通信是如何使用Jute的？/笔记.docx**

《125\_Jute是什么序列化协议？ZooKeeper内部通信是如何使用Jute的？》

zk里的序列化的协议是jute，类似于im系统里用的protobuf

序列化和反序列化

比如我们有一个类，我现在要通过网络来传输这个类的对象，序列化，你得把类转化为字节数组/字节流，通过网络传输类对象的字节流，到了对方收到自己流之后，就把字节转换为一个类的对象，反序列化

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/139\_下周课程预告：Session的建立以及工业级的Session管理机制/笔记.docx**

《139\_下周课程预告：Session的建立以及工业级的Session管理机制》

我们可以学到的东西

（1）对集群内部的通信，完全可以参考zk采用传统的BIO通信架构

（2）面向客户端的网络通信，还是需要使用NIO通信架构

下周的内容

参考zookeeper的一套工业级的session管理机制

（1）Session建立：通过发送ConnectRequest到服务端

（2）海量Session的分桶管理机制

（3）Session的维护机制：心跳机制，重连机制，故障感知，超时机制

（4）Session的数据结构：sessionId是如何生成的

（5）Session的清理机制

读写请求的发送，请求的转发，数据的同步，ZAB协议（2PC+过半写机制），事务日志机制，内存数据库机制，定期快照机制，值得用2周的时间去学习，常见的分布式协议，Raft、Paxos、ZAB协议之前的对比

启动时的数据恢复机制，初始化数据同步机制，leader崩溃时候的数据恢复机制

一个半月的时间，能把zookeeper如此优秀，老牌，运用广泛，底层的一个分布式系统源码里面蕴含的所有优秀的架构设计思想，网络通信机制，磁盘读写机制，日志+磁盘+快照+内存的存储机制，分布式一致协议和机制，peers+leader+follower分布式架构，海量客户端接入和工业级session管理机制，集群数据恢复机制 + 高可用的崩溃恢复机制

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/137\_服务端接收到连接之后会做什么事情？/笔记.docx**

《137\_服务端接收到连接之后会做什么事情？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/126\_Follower在完成连接建立之后是如何向Leader进行注册的？/笔记.docx**

《126\_Follower在完成连接建立之后是如何向Leader进行注册的？》

jute序列化和字节流输出的一个过程，并发，集合，IO，网络，netty，带着做底层的项目，看zk的源码，其实是很简单的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/132\_创建一个ZooKeeper对象实例就是与服务端建立一个长连接？/笔记.docx**

《132\_创建一个ZooKeeper对象实例就是与服务端建立一个长连接？》

慢慢的一点一点的看明白这个东西，这里的源码细节还是挺多的，其实我对hdfs、kafka、zookeeper的源码都是讲解的比较细致的，代码的细节基本上都讲到位了，自己再看一些额外的细节性的源码

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/135\_启动ClientCnxn的时候是在干什么？/笔记.docx**

《135\_启动ClientCnxn的时候是在干什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/121~139资料/121\_Leader选举完毕之后各自确定角色以及创建核心实体对象/笔记.docx**

《121\_Leader选举完毕之后各自确定角色以及创建核心实体对象》

zk这种知名的开源项目，nio、netty去做的网络通信，人家就是用的最最简单的，连参加过6个月速成java就业培训的同学，都会写的网络通信代码，java bio，server socket，socket，建立连接

数据传输，就特别的简单，无非就是利用DataOutputStream，DataInputStream，字节输出流，字节输入流，传输一些数据这样子

如果以后我们要做类似的Peer-to-Peer的集群架构，里面有leader-follower的角色区分，在多个peers节点之间要选举一个Leader出来的话，我们完全可以参考zk这套非常成熟的leader选举的算法

netty去建立节点之间的长连接，Java架构IM系统项目，基于netty做长连接，分布式海量小文件存储系统的项目，nio做长连接，我觉得zk里有一些源码细节做的不是太好，我不得不说，相交于kafka、hdfs的服务端的源码而言

我觉得zk的源码质量和代码设计要比hdfs、kafka要好很多，scala来写，特别不赞同也不喜欢用scala编程语言，过渡追求一些花哨的语法糖了

QuorumPeer：封装的是zk作为peer通用的代码，比如Leader选举，状态判断

他封装了不同的角色的实体对象，Leader、Follower、Observer

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/064\_Curator的NodeCache节点监听机制的源码分析/笔记.docx**

《064\_Curator的NodeCache节点监听机制的源码分析》

node cache，监听一个node的数据和deleted事件的变化，zookeeper的API，去加一个watcher，有事件反过来回调你的listener

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/075\_研究完Curator源码之后对我们后续学习的意义在哪里？/笔记.docx**

《075\_研究完Curator源码之后对我们后续学习的意义在哪里？》

彻底搞透了ZooKeeer客户端一般怎么玩儿

要不然是看人家怎么用，要不然是自己用

比如说，HDFS、Kafka、Canal，人家怎么用zk，curator，ZooKeeperClient，原生的ZK API，实现各种各样的高阶功能，如何基于原生ZK API实现高阶功能，一定能看懂各种开源项目是如何使用zk的

如果是自己用zk，如果有些你需要的zk高阶功能，curator框架没有，此时你可以自己很熟练的基于zk原生API去玩儿出一些花来，基于curator封装好的crud + 监听回调玩儿出花来

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/062\_快速过一下Curator的CRUD操作的源码流程/笔记.docx**

《062\_快速过一下Curator的CRUD操作的源码流程》

curator的crud的操作，底层都是调用的原生zk的API，支持异步化的操作

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/066\_小实验：PathCache实现自动重复注册监听器的效果/笔记.docx**

《066\_小实验：PathCache实现自动重复注册监听器的效果》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/072\_Double Barrier机制的源码剖析/笔记.docx**

《072\_Double Barrier机制的源码剖析》

readyPath: /double/barrier/ready

ourPath: /doble/barrier/dsfsd-23423-fsd-23423-dfdf

每台机器都会在你指定的path下创建一个UUID的子节点，临时节点

每台机器都会检查一下/double/barrier下的子节点，数量超过了你指定的数量，此时就会创建一个ready节点出来

如果说子节点的数量没有超过你指定的数量，wait()

10个，第一台机器创建一个UUID，getChildren()检查发现没有达到指定的10个，wait()，同时对readyPath加一个watcher，等待人家创建出来

直到第10台机器创建了UUID节点，此时已经达到了你指定的数量了，此时创建一个readyPath出来，会通过watcher回调之前所有的监听readyPath的机器

他会把你的节点给删除掉

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/074\_如何基于ZK原生API封装出各类高阶功能？/笔记.docx**

《074\_如何基于ZK原生API封装出各类高阶功能？》

zk核心的API功能，crud + 节点类型（持久，临时，顺序），监听+回调

leader选举 -> 创建临时顺序节点 -> leader -> 监听 -> 回调 -> 重新leader

分布式锁 -> 创建临时顺序节点 -> leader -> 监听 -> 回调 -> 重新leader

Barrier -> 创建节点 -> 监听+回调 -> 删除

分布式计数器 -> 用一个节点存储数值 -> 变更了回调

分布式队列（官方不推荐） -> 创建很多节点 -> 变更回调

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/063\_Curator的PathCache子节点监听机制实现源码分析/笔记.docx**

《063\_Curator的PathCache子节点监听机制实现源码分析》

zk，crud，znode增删改查，就可以基于zk去存放一些分布式系统的集群元数据

针对这些元数据进行监听，如果有变化就反向通知我们的客户端

ZooKeeper对象，zk原生API，负责跟zk server建立长连接，心跳，维持一个session，发送请求到zk server，注册监听器，接收zk server推送过来的事件

PathCache，子节点数据缓存，注册监听器，接收事件；NodeCache

zookeeper，getChildren()，watching，给他一个监听器，zk如果有子节点发生事件，会通知原生的watcher，反过来会调用你注册的listener

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/069\_第一种Leader选举机制的源码流程分析（2）/笔记.docx**

《069\_第一种Leader选举机制的源码流程分析（2）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/065\_小实验：对子节点注册监听器后如何获取事件回调通知？/笔记.docx**

《065\_小实验：对子节点注册监听器后如何获取事件回调通知？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/067\_小实验：NodeCache实现节点变化事件监听的效果/笔记.docx**

《067\_小实验：NodeCache实现节点变化事件监听的效果》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/058\_Curator客户端实例创建的源码剖析：如何与ZooKeeper建立连接？/笔记.docx**

《058\_Curator客户端实例创建的源码剖析：如何与ZooKeeper建立连接？》

构造器模式是什么，阶段一，20多种设计模式实战开始学起，curator框架，底层是对zookeeper原生API进行的封装，在原生ZooKeeper的API里，构建一个ZooKeeper对象，去代表了ZooKeeper的客户端

CuratorZooKeeperClient，他是Curator里封装了ZooKeeper的客户端，他里面传进去了一大堆的ZooKeeper需要的参数，另外搞了一个Watcher监听器，在zk里，你可以施加一个watcher，去监听zk里发生的一些事件

ZooKeeper（负责跟zk建立连接） -> Watcher -> 回调CuratorFramework上加的Listerner

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/060\_Curator客户端是如何在启动的时候跟ZK建立连接的？/笔记.docx**

《060\_Curator客户端是如何在启动的时候跟ZK建立连接的？》

启动一个线程，监听eventQueue，代表的是跟zk之间的网络连接变化的事件，收到这种事件之后，应该就会通知ConnectionStateListener

return new ZooKeeper(connectString, sessionTimeout, watcher, canBeReadOnly);

start，会创建原生ZooKeeper的客户端，连接串，session过期时间，watcher，监听zk里的事件的变更，回传给curator的framework，他在底层的源码，就会去跟zk的一台机器建立真正的TCP长连接

之后就会进行心跳，维护一个session，发送各种请求过去给zk server，注册watcher，zk有事件变更，会通过TCP长连接，反向通知你的ZooKeeper客户端，他会回调你提供给他的watcher

ZooKeeper原生API的源码，我们等到最后两三周的时候再讲，最好是先把zk server端的源码看一下，再看zk客户端的源码，就很简单了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/070\_第二种Leader选举机制的源码分析/笔记.docx**

《070\_第二种Leader选举机制的源码分析》

分布式锁来竞争成为leader的，如果说你获取到了锁，就说明你是leader，获取锁的过程跟我们上一讲讲的第一种leader选举机制的底层是几乎一样的

竞争leader的时候获取锁失败了呢？

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/059\_Curator底层是如何创建原生ZK客户端以及注册Watcher监听器的？/笔记.docx**

《059\_Curator底层是如何创建原生ZK客户端以及注册Watcher监听器的？》

CuratorFramework -> CuratorZooKeeperClient -> zookeeperFactory + watcher

ZooKeeper客户端跟zk连接，以及注册watcher监听器

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/071\_Barrier机制的源码剖析/笔记.docx**

《071\_Barrier机制的源码剖析》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/073\_分布式计数器的源码剖析/笔记.docx**

《073\_分布式计数器的源码剖析》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/068\_第一种Leader选举机制的源码流程分析/笔记.docx**

《068\_第一种Leader选举机制的源码流程分析》

crud + 监听回调机制：组合起来，就是大部分的分布式系统用zk的场景

如果我们用原生的zk去注册监听器的话，监听子节点或者节点自己，如果发生了对应的事件，会通知你一次，但是下一次再有事件就不会通知了。zk原生的API里，需要你每次收到事件通知之后，都需要自己重新注册watcher

PathCache，NodeCache，给你的节点加监听器，如果说监听到了事件之后，然后得到了通知，下次他会自动给你重新注册监听器

竞争成为leader，是通过znode的创建

就是在你指定的目录下创建一个子节点，创建一个临时顺序节点，会加上一个编号，zk分布式锁

/leader/latch

三台机器同时竞争自己能否成为leader，此时都会依次在/leader/latch下面创建临时顺序节点

/leader/latch/lock-dfsfaa000000000001

/leader/latch/lock-dfsfaa000000000002

/leader/latch/lock-dfsfaa000000000003

每台机器创建完临时顺序节点之后，此时就会对/leader/latch目录执行getChildren()，去获取里面所有的子目录，通过backgroundCallback回调通知给你，对获取到的子节点做一个排序，然后看看自己是不是第一个子节点

就是leadership

如果你发现自己不是leader，对自己上一个节点施加一个监听器，在加监听器的同时，使用的是checkExists方法，会去检查一下自己的上一个节点是否存在，是否存在的结果，会通过backgrondCallback回调给你

如果发现上一个节点不存在了，此时会重新再次尝试去创建一个znode，相当于是竞争成为leader，如果说上一个节点对应的客户端释放了leader角色，此时上一个节点会消失，会回调通知第二个节点对应的客户端

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/058~075资料/061\_基于Curator客户端创建znode的源码流程分析/笔记.docx**

《061\_基于Curator客户端创建znode的源码流程分析》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/057\_研究Curator客户端的核心功能源码的意义在哪里？/笔记.docx**

《057\_研究Curator客户端的核心功能源码的意义在哪里？》

ZooKeeper本身人家是有Java客户端框架的，原生的，但是只能实现最最基本的功能，功能也足够，但是用起来很繁琐，很复杂

Curator实际上是对ZooKeeper原生框架进行了一个封装，提供更好用的API，更简单易用的API，更加强大的一些功能，Leader选举，光是会用Curator，很多开源项目，ZooKeeper原生的框架

分布式锁，研究了里面的源码，他是如何通过底层ZooKeeper的最基本的一些操作去实现复杂的功能的

CRUD、监听和通知、Leader选举机制、Barrier、Counter，Curator，他是如何基于最基础的ZooKeeper的一些操作来实现复杂功能，ZooKeeper原生客户端的使用，你也就知道是怎么回事了，怎么玩儿的了

HDFS、Kafka、Canal里面是如何通过ZooKeeper客户端框架去操作他，实现各种他们需要的生产级的功能的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/047\_基于Curator实现的分布式系统的Barrier机制/笔记.docx**

《047\_基于Curator实现的分布式系统的Barrier机制》

很多台机器都创建了一个barrier，此时他们都不能动了，除非满足一个条件，才能让他们一起往下动

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/054\_Curator支持的ZK中各种节点的说明/笔记.docx**

《054\_Curator支持的ZK中各种节点的说明》

持久化节点，如果创建节点的客户端连接断开了，一般也会必须要让数据保存在zk里的

临时节点，如果创建节点的客户端连接断开了，此时他之前创建的临时节点就会自动消失，自动删除掉，所以说如果你监听了那个临时节点，一旦他被删除了，此时你就会收到这个通知的

顺序节点，顺序持久化节点，顺序临时节点

比如在/distributed/lock，创建临时顺序节点，就会自动创建子节点，子节点自动进行编号，000000000001，000000000002

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/048\_基于Curator实现的双重Barrier机制/笔记.docx**

《048\_基于Curator实现的双重Barrier机制》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/046\_分布式锁在哪些系统里和场景中有用？/笔记.docx**

《046\_分布式锁在哪些系统里和场景中有用？》

在分布式业务系统里有用，基础架构里的分布式系统，分布式大数据系统，分布式业务系统

分布式消息中间件，分布式缓存系统，分布式数据库系统，属于基础架构里的东西，分布式服务注册中心，他们一般是服务于各种Java业务系统的

分布式大数据，HDFS（分布式存储系统），YARN（分布式资源调度系统），Spark（分布式计算系统），HBase（分布式NoSQL数据库），Flink（分布式流式计算系统）

分布式业务系统，分布式电商系统，分布式社交系统，分布式广告系统，一般是基于Dubbo、Sping Cloud构建的分布式业务系统，拆分成很多的子系统或者是服务，互相之间进行通信，构成了一个分布式系统

在Java架构的分布式系统里面，基于电商业务的分布式锁的实战

对一个数据进行复杂操作的时候，对一个订单，要同时在缓存、数据库、ES里进行数据更新，此时必须加一把分布式锁，保证同一个时间别人不能来更新

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/044\_Curator的第二种Leader选举机制的代码实验/笔记.docx**

《044\_Curator的第二种Leader选举机制的代码实验》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/043\_Curator的第一种Leader选举机制的代码实验/笔记.docx**

《043\_Curator的第一种Leader选举机制的代码实验》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/049\_分布式系统Barrier机制在哪些场景中有用？/笔记.docx**

《049\_分布式系统Barrier机制在哪些场景中有用？》

几乎没用，大部分分布式系统，集群元数据管理 + 监听和通知，Master选举

分布式业务系统，分布式锁

Barrier，协调一批机器，一起等待，等到满足某个条件，再让你一起动

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/038\_通过命令行来操作ZooKeeper试一试/笔记.docx**

《038\_通过命令行来操作ZooKeeper试一试》

sh zkServer.sh start

sh zkCli.sh

就是类似于文件系统一样的东西而已，创建节点，有层级关系，节点的增删改查，创建监听器，监听节点和数据的变化，他可以反过来通知给你的客户端

创建一个节点：create

查看目录：ls

读取数据：get

更新数据：set

删除节点：delete

三种类型：持久化节点，临时节点，顺序节点（自动给节点排上序号）

zk的数据模型就是这样，命令都很简单

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/039\_业内使用最广泛的ZooKeeper客户端框架：Curator是什么？/笔记.docx**

《039\_业内使用最广泛的ZooKeeper客户端框架：Curator是什么？》

命令行操作zk，一般就是一些zk集群运维的工程师可能会去使用，一般我们都不会去用的，Curator，分布式锁，ZookeeperClient，官方Java API

netflix公司，雅虎、linkedin、facebook、google、netflix、Amazon，百度、腾讯、微博、优酷土豆、新浪、阿里

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/042\_基于Curator和ZK实现的Leader选举的作用：HA主备启动切换/笔记.docx**

《042\_基于Curator和ZK实现的Leader选举的作用：HA主备启动切换》

HDFS、Kafka和Canal都用到了ZK进行leader选举

HDFS，NameNode是可以部署HA架构的，主备两台机器，如果主机器宕机了，立马备用的机器可以感知到，立马选举为Leader，作为新的NameNode对外提供服务

Kafka，有一个很关键的角色，Controller，负责管理整个集群的协作，任何一个Broker都恶意变成Controller，Leader角色

Canal，部署主备两台机器，主机器挂掉了，备用机器就可以跟上去

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/050\_基于Curator实现的ZooKeeper分布式计数器的介绍/笔记.docx**

《050\_基于Curator实现的ZooKeeper分布式计数器的介绍》

如果真的要实现分布式计数器，应该是用redis来实现，并发量更高，性能更好，功能更加的强大，分布式技术，lua脚本嵌入进去复杂的业务逻辑

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/056\_ZooKeeper客户端功能大总结/笔记.docx**

《056\_ZooKeeper客户端功能大总结》

基于CRUD + 监听和通知 + Leader选举：分布式中间件系统，分布式大数据系统

分布式锁：分布式业务系统

Barrier、Counters、Queues：基于ZK实现的分布式Barrier、分布式计数器、分布式队列，都不推荐使用，使用场景很少，推荐用Redis做分布式计数，推荐用RabbitMQ / RocketMQ做分布式队列

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/041\_基于Curator进行ZK数据的管理是如何实现集群元数据管理的？/笔记.docx**

《041\_基于Curator进行ZK数据的管理是如何实现集群元数据管理的？》

curator可以操作zk，比如说你现在自己研发了一套分布式系统，类似于kafka，canal，现在想把集群运行的核心元数据，都放到zk里去

一堆的znode，在里面写入对应的值，推荐你值可以用json格式

其他人在需要的时候就可以从里面读取出来集群元数据

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/052\_基于Curator实现ZK的节点数据监听机制/笔记.docx**

《052\_基于Curator实现ZK的节点数据监听机制》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/055\_基于Curator实现的ZK分布式队列（官方不建议使用）/笔记.docx**

《055\_基于Curator实现的ZK分布式队列（官方不建议使用）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/040\_基于Curator框架进行基本的ZooKeeper数据操作/笔记.docx**

《040\_基于Curator框架进行基本的ZooKeeper数据操作》

znode进行增删改查

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/051\_基于Curator实现ZK的子节点监听机制/笔记.docx**

《051\_基于Curator实现ZK的子节点监听机制》

增删改查做元数据的维护和管理，监听和通知机制，监听元数据的变化得到通知

Leader选举

/cluster

/cluster/id

/cluster/data

/cluster/changes

/cluster/id/current

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/053\_基于ZooKeeper的数据管理和监听机制可以实现的典型场景/笔记.docx**

《053\_基于ZooKeeper的数据管理和监听机制可以实现的典型场景》

配置中心，可以把分布式业务系统，分布式电商系统，可以把他的一些配置不要写在本地磁盘文件里，可以放入zk中，你可以基于zk封装一个配置中心

分布式电商系统，就可以从配置中心（zk）获取一些配置，对配置加入监听

如果说配置变更了，立马就可以通过zk通知到所有监听配置项的系统，让他们立马根据配置做出一些变化

分布式业务系统，降级，打开一个降级开关，所有系统从配置感知到到了之后，立马就可以进行降级

集群负载均衡，比如说IM系统，Java架构课程，这个大数据的学员都可以看的，你有一些系统，启动可以去zk里注册自己，创建一个临时节点，ip-list服务可以被反过来通知到最新的机器list变化，做负载均衡，别的服务如果要调用你的系统，此时可以找ip-list，可以随机选择一台机器

如果你部署的系统宕机了一台，重启一台，此时zk里的临时节点就没了

此时zk通知ip-list服务，就能感知到某台机器不可以对外提供服务了，此时调用他的其他系统从ip-list服务这里就不会获取到宕机或者重启的机器了

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/038~057资料/045\_基于Curator实现的各种分布式锁的复习和回顾/笔记.docx**

《045\_基于Curator实现的各种分布式锁的复习和回顾》

Java架构，分布式阶段里，分布式锁，透彻的剖析了Curator实现各种分布式锁的源码

ZK分布式锁，通过顺序临时节点

所有人去竞争一个锁，此时在你指定的锁目录下，创建一个顺序临时节点，看看你是不是第一个节点，如果是的话，就是获取到了锁，如果不是第一个节点，就对你上一个节点加监听器

如果有人释放锁了，第二个节点会得到一个通知，自己是第一个节点，就获取到了锁

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/143\_初步看一下服务端是如何创建一个Session的？/笔记.docx**

《143\_初步看一下服务端是如何创建一个Session的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/150\_客户端是如何定期发送Ping心跳到服务端的？/笔记.docx**

《150\_客户端是如何定期发送Ping心跳到服务端的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/157\_SessionTracker如何高效性的使用分桶策略过期Session？/笔记.docx**

《157\_SessionTracker如何高效性的使用分桶策略过期Session？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/140\_客户端的SendThread是如何将请求发送给服务端的？/笔记.docx**

《140\_客户端的SendThread是如何将请求发送给服务端的？》

建立连接主要说的是TCP物理连接，他其实还需要进行一些通信，建立一个Session会话

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/156\_SessionTracker是如何去定期检查会话是否过期的？/笔记.docx**

《156\_SessionTracker是如何去定期检查会话是否过期的？》

expireTime是如何确定的呢？如果会话一直被touch激活是如何逃避掉被过期会话的命运的呢？具体过期一个会话是如何执行的呢？

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/158\_如果客户端网络出现问题，Session是如何处理的？/笔记.docx**

《158\_如果客户端网络出现问题，Session是如何处理的？》

如果说session还在过期时间范围内，在这个范围内你的客户端恢复了网络，正常发送请求或者ping过去，就会把你的session在过期之前，成功的转移到下一个分桶里去，你的session是不会过期掉的

如果说你太长时间没有发送请求或者ping了，此时session在zk服务端被过期了，此时你发送请求过去，人家发现你的session已经过期了，你大不了就是重新建立一次session就可以了

如果说你的客户端多次请求zk服务端无果，然后你重新尝试连接其他的zk服务端了，大不了你跟其他的zk服务器建立连接和重新初始化一个会话即可；对于之前跟那个zk服务端的会话，他自己会正常给你过期掉的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/154\_会话touch机制、请求处理链条以及响应发送/笔记.docx**

《154\_会话touch机制、请求处理链条以及响应发送》

submitRequest -> touch session -> request processors -> response

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/146\_SessionId是如何通过二进制位运算实现唯一性的？/笔记.docx**

《146\_SessionId是如何通过二进制位运算实现唯一性的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/148\_如何对Session按照过期时间进行分桶管理？（上）/笔记.docx**

《148\_如何对Session按照过期时间进行分桶管理？（上）》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/155\_客户端收到CreateSession、Ping的响应之后会干什么？/笔记.docx**

《155\_客户端收到CreateSession、Ping的响应之后会干什么？》

自己写的分布式海量小文件存储系统，kafka，zookeeper，底层的网络通信，BIO、NIO，没什么区别，Kafka是有一套支撑高并发的网络通信架构，reactor架构，对于zk而言，他所有的请求都是基于队列，进行异步化的链式处理

kafka，通过网络连接读取请求，请求进入队列，由其他的线程来进行处理，reactor架构

dfs，跟kafka类似

zookeeper，解析请求，放入队列，后台通过几个队列加上几个线程进行链式处理

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/144\_ZooKeeperServer组件在架构中的已纳入以及初始化时机/笔记.docx**

《144\_ZooKeeperServer组件在架构中的已纳入以及初始化时机》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/141\_服务端接收到ConnectRequest请求后会如何处理？/笔记.docx**

《141\_服务端接收到ConnectRequest请求后会如何处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/147\_用于保存Session的核心数据结构都有哪些？/笔记.docx**

《147\_用于保存Session的核心数据结构都有哪些？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/142\_如何基于Jute将ConnectRequest进行反序列化？/笔记.docx**

《142\_如何基于Jute将ConnectRequest进行反序列化？》

看一下工业级的session管理机制

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/159\_如果客户端崩溃或者重启，此时对会话是如何处理的？/笔记.docx**

《159\_如果客户端崩溃或者重启，此时对会话是如何处理的？》

如果是zk服务器崩溃了呢？丢失之前所有在内存里维护的会话

之前他连接的所有的客户端都会发现网络异常，此时都会尝试连接其他的zk服务器，跟其他zk服务器建立连接

工业级的会话管理的机制，zk的session机制

会话的sessionId基于位运算实现唯一性，另外一个是session分桶机制，只要有请求或者ping，不强制要求是心跳，都会touch session，分桶转移，session tracker而言就是不停的过期一个一个分桶

session过期检查的效率是很高的

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/153\_PreRequestProcessor处理完之后接下来干什么？/笔记.docx**

《153\_PreRequestProcessor处理完之后接下来干什么？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/151\_服务端是如何接收和处理Ping心跳请求的？/笔记.docx**

《151\_服务端是如何接收和处理Ping心跳请求的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/152\_遗漏的源码：服务端创建完Session之后干了什么？/笔记.docx**

《152\_遗漏的源码：服务端创建完Session之后干了什么？》

（1）不管如何都会touch一下session，更新他的expireTime，重新分桶

（2）submit Request

（3）会交给RequestProcessor线程来进行处理

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/145\_SessionTracker是如何创建一个新的会话出来的？/笔记.docx**

《145\_SessionTracker是如何创建一个新的会话出来的？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/08\_ZooKeeper顶尖高手课程：从实战到源码unzip/140~159资料/149\_如何对Session按照过期时间进行分桶管理？（下）/笔记.docx**

《149\_如何对Session按照过期时间进行分桶管理？（下）》