# 1服务器角色

## Leader

1. 事务请求的唯一调度者和处理者。保证事务处理的顺序性。
2. 集群内部个服务器之间的调度者。

导致数据一致性的请求，事务请求

LearnerHandler，在集群中Leader与Follower、Observer的通讯的类。

1. PrepRequestProcessor：主要的作用就是生成zxid。一个Request提交后，Leader可以断言（带来的影响，顺序节点的序列号）
2. ProposalRequestProcessor 事务投票处理器。

事务请求：除了提交给CommitProcessor，还会根据对应的请求创建对应的Proposal，并发送给所有的Follower服务器进行一次集群内部事务投票。

非事务请求：直接发送给CommitProcessor，不再做其他处理。

1. SyncRequestProcessor 事务+日志的处理器

处理事务并且将事务记录到日志文件中，同时触发数据快照（有条件的触发）

1. AckRequestProcessor

处理完事务并且记录事务日志后，向Proposal的投票收集器发送ACK反馈，通知投票收集器当前服务器已经完成对Proposal事务日志记录。

1. CommitProcessor 事务提交处理器

CommitProcessor等待集群内针对于该Proposal的投票直到该Proposal可以被提交（多数派协议）。并且将请求交给下一个处理器。

1. ToBeAppliedRequestProcessor

在向所有的Follower发送Commit，必须保证Commit执行顺序。同时把请求交给下一个处理器，然后从队列中将Commit移除。

1. FinalRequestFollower

## Follower

1. 处理客户端的非事务请求。事务请求必须转发给Leader服务器。
2. 参与事务请求Proposal的投票
3. 参与Leader选举

FollowerRequestProcessor。

处理client请求队列，用take方法逐个取出，然后交付给下一RequestProcessor。

如果事务请求，那么必须转发给Leader。

SyncRequestProcessor

对于FollowerRequestProcessor转发给leader的请求，Leader经过处理发送给所有Follower（Proposal）

SendAckRequestProcessor

当Follower收到Proposal，就会回馈Leader一个ACK包。

CommitProcessor

FinalRequestProcessor

## Observer

在实际运行中，它只是负责读，Leader不会将事务的投票发送给Obsserver。

ObserverRequestProcessor

# 序列化与通信协议

序列化协议

* ProtoBuf
* Thrift
* Hessian
* Kryo
* Avro
* JDK Serializable
* Jsoniter/Jackson

Jute是zk序列化、反序列化协议。

Record：

serialize(OutputArchive archive, String tag);

deserialize(InputArchive archive, String tag);

几乎org.apache.zookeeper.proto.\*下所有的类都实现了Record接口

OutputArchive 输出类

InputArchive 输入类

tag 代表表示对象的标识。

BinaryOutputArchive/BinaryInputArchive 网络传输、本地磁盘持久化

CsvOutputArchive/CsvInputArchive 方便展示

XmlOutputArchive/XmlInputArchive 还没使用。以xml保存和还原

## 通信协议

基于TCP/IP协议，所以是一个长连接。zookeeper在这个基础上完成客户端和服务器，服务器和服务器之间的通信。

ConnectRequest

ConnectResponse

ReplyHeader

ResponeHeader

org.apache.zookeeper.proto.\*

Zk请求包：请求头+请求体

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0-3 | 4-11 | | 12-n | | |
| len | 4-7 | 8-11 | 12-15 | 16-(n-1) | n |
| xid | type | len | path | watch |

Zk响应包：响应头+响应体

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0-3 | 4-19 | | | 20 - n | | | |
| len | 4-7 | 8-15 | 16-19 | 20-23 | len位 | 48位 | 8位 |
| xid | zxid | err | len | data | ...... | pzxid |

# 数据存储

## 内存数据

Zk有个内存数据库：ZkDataBase、DataTree、DataNode

## 日志数据FileTxnLog

运行时，不停地有数据写入。

当日志的剩余空间不足4K（4096），日志就做扩充，扩充64M，后面以“0”填充。

log都是使用zxid作为文件名的后缀。

查看日志方式：

$ java -cp ./zookeeper-3.4.9.jar::./lib/log4j-1.2.16.jar:./lib/slf4j-api-1.6.1.jar:./lib/slf4j-log4j12-1.6.1.jar org.apache.zookeeper.server.LogFormatter ~/tmp/zookeeper/zk1/version-2/log.f00000001

## 快照数据FileSnapTxnLog

快照数据：在某一时刻内存所有全量数据的一个磁盘文件。举例：快照阈值100000，触发快照数据。

快照数据都是使用zxid作为文件名的后缀。

查看快照命令：

$ java -cp ./zookeeper-3.4.9.jar::./lib/log4j-1.2.16.jar:./lib/slf4j-api-1.6.1.jar:./lib/slf4j-log4j12-1.6.1.jar org.apache.zookeeper.server.SnapshotFormatter ~/tmp/zookeeper/zk1/version-2/snapshot.f00000697

快照触发机制，非“半数机制”，过半随机策略。

logcount > (snapcount/2 + randroll)

logcount: 代表当前记录日志数量

snapcount: 多少次事务日志记录后触发一次数据快照

randroll： 1~snapcount/2 之间的一个随机数