# ZAB 协议

ZAB ，Zookeeper Atomic Broadcast，zk 原子消息广播协议，是专为 ZooKeeper 设计的一种支持崩溃恢复的原子广播协议，在 Zookeeper 中，主要依赖 ZAB 协议来实现分布式数据一致性。

Zookeeper 使用一个单一主进程来接收并处理客户端的所有事务请求，即写请求。当服务器数据的状态发生变更后，集群采用 ZAB 原子广播协议，以事务提案 Proposal 的形式广播到所有的副本进程上。ZAB 协议能够保证一个全局的变更序列，即可以为每一个事务分配一个全局的递增编号 xid。

当 Zookeeper 客户端连接到 Zookeeper 集群的一个节点后，若客户端提交的是读请求， 那么当前节点就直接根据自己保存的数据对其进行响应；如果是写请求且当前节点不是 Leader，那么节点就会将该写请求转发给 Leader，Leader 会以提案的方式广播该写操作，只 要有超过半数节点同意该写操作，则该写操作请求就会被提交。然后 Leader 会再次广播给所有订阅者，即 Learner，通知它们同步数据。

# ZAB 与 Paxos 的关系

ZAB 协议是 Fast Paxos 算法的一种工业实现算法。但两者的设计目标不太一样。

ZAB 协议主要用于构建一个高可用的分布式数据主从系统，即 Follower 是 Leader 的备用机，Leader 挂了，马上就可以选举出一个新的 Leader。

而 Fast Paxos 算法则是用于构建一个分布式一致性状态机系统，确保系统中各个节点的状态都是一致的。

另外，ZAB 还使用 Google 的 Chubby 算法作为分布式锁的实现，而Google 的 Chubby 也

是 Paxos 算法的应用。

# Zookeeper的角色

为了避免 Zookeeper 的单点问题，zk 也是以集群的形式出现的。

zk 集群中的角色主要有以下三类：

**Leader**：接收和处理客户端的读请求；zk 集群中事务请求的唯一处理者，并负责发起决 议和投票，然后将通过的事务请求在本地进行处理后，将处理结果同步给集群中的其它

主机

**Follower**：接收和处理客户端的读请求; 将事务请求转给 Leader；同步 Leader 中的数据； 当 Leader 挂了，参与 Leader 的选举（具有选举权与被选举权）

**Observer**：就是没有选举权与被选举权，且没有投票权的 Follower（临时工）。若 zk 集 群中的读压力很大，则需要增加 Observer，最好不要增加 Follower。

因为增加 Follower将会增大投票与统计选票的压力，降低写操作效率，及 Leader 选举的效率。

# Observer 设置为多少合适？

Observer 数量一般与 Follower 数量相同。并不是 Observer 越多越好，因为Observer 数量的增多虽不会增加事务操作压力，但其需要从 Leader 同步数据，Observer 同步数据的时 间是小于等于 Follower 同步数据的时间的。当 Follower 同步数据完成，Leader 的 Observer 列表中的 Observer 主机将结束同步。那些完成同步的 Observer 将会进入到另一个对外提供 服务的列表。那么，那些没有同步了数据无法提供服务的 Observer 主机就形成了资源浪费，其实也不是特别浪费，后期会通过心跳来和leader来同步，频繁与leader同步。

所以，对于事务操作发生频繁的系统，不建议使用过多的 Observer。

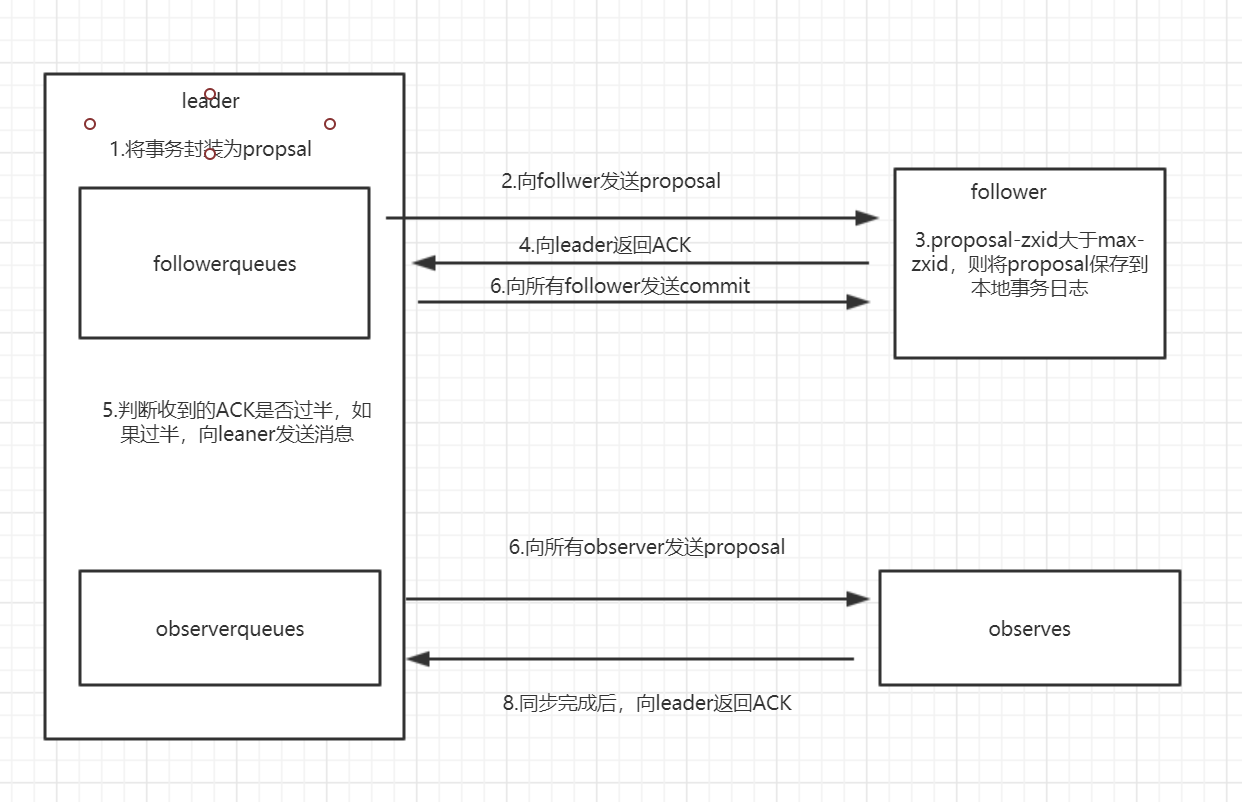
# ZK状态

LOOKING 选举状态

FOLLOWING FOLLOW的正常工作状态，从leader同步数据的状态

OBSERVING OBSERVER的正常工作状态，从leader同步数据的状态

LEADING leader的正常工作状态，leader广播数据更新的状态



Leader 中对于Observer 存在两个列表：all（包含所有 Observer 主机）、service（每发生 一次事务更新，service 列表就会发生一次变化）。

service <= all {all – service} 心跳

service 队列为OBSERVING状态的observer，这些observer是能提供服务的observer，{all – service} 的集合中observer无法提供服务，后期可以通过心跳链接，然后同步，变为OBSERVING状态，才能向外提供服务，

Leader 中对于 Follower 也存在两个列表：all（包含所有 Follower 主机）、service（每发 生一次事务更新，service 列表就会发生一次变化）。

若 service <= all/2 ，则说明同步失败，则 Leader 会重新广播，Follower会 重新同步，不会回滚，直到ACK的反应过半

{all – service} 如果和leader脸上心跳，同步数据，然后放回service队列，向外提供服务。

这三类角色在不同的情况下又有一些不同的名称：Quorum 翻译为 法定人数

Learner = Follower + Observer

QuorumServer = Follower + Leader

# ZK消息传递的三个重要数据

zxid：是一个 64 位长度的 Long 类型。其中高 32 位表示 epoch，低 32 表示 xid。

epoch：每个 Leader 都会具有一个不同的 epoch，用于区分不同的时期

xid：事务 id，是一个流水号

# ZK运行三种模式

ZAB 协议中对 zkServer 的状态描述有三种模式。这三种模式并没有十分明显的界线，它 们相互交织在一起。

恢复模式：当集群启动时，或当 Leader 挂了进，系统需要进入恢复模式，以恢复系统 对外提供服务的能力。其中包含两个很重要的阶段：Leader 的选举与初始化同步。

广播模式：分为两类，初始化广播与更新广播。

同步模式：分为两类，初始化同步与更新同步。