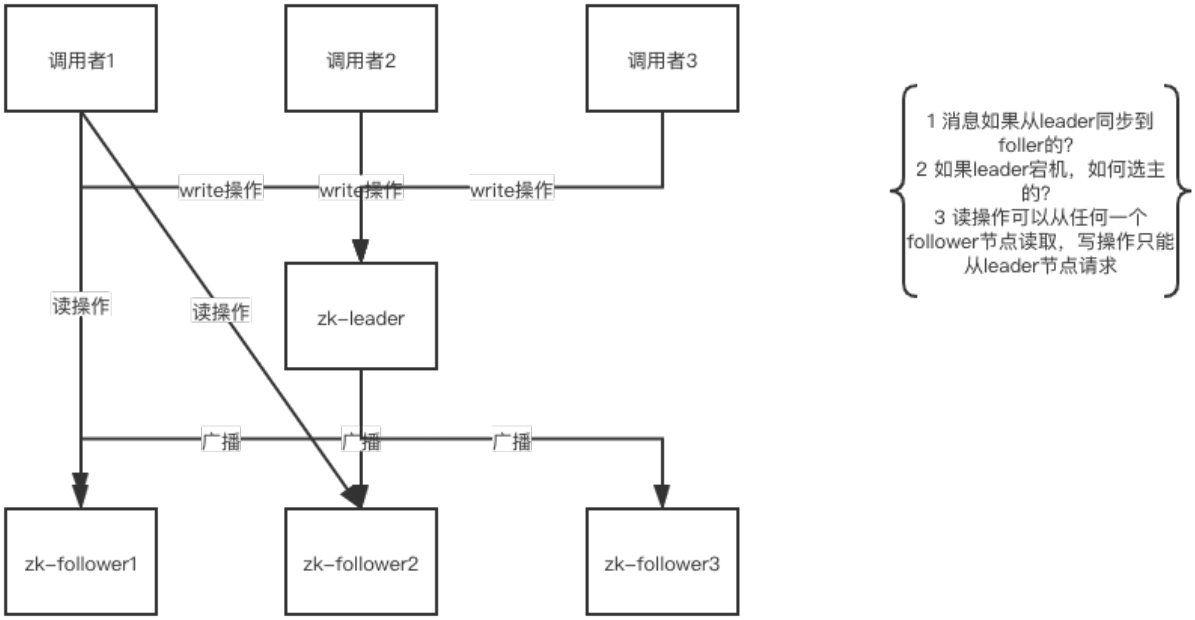
1 什么是zk，能做什么?

zoo-keeper分布式协调服务，可以实现服务注册（dubbo,spring cloud的注册中心）、分布式锁（有redis、zk两种）、选主、配置管理(项目配置可以放在zk，做到统一管理，动态更新)

原理是：zk底层是一个类似于linux文件系统的树型结构，同时提供了对每个节点的监控和通知机制。

2 zk集群架构，主丛复制的模式。一共分位三种角色：1 leader、follower、observer

zk集群架构如下:

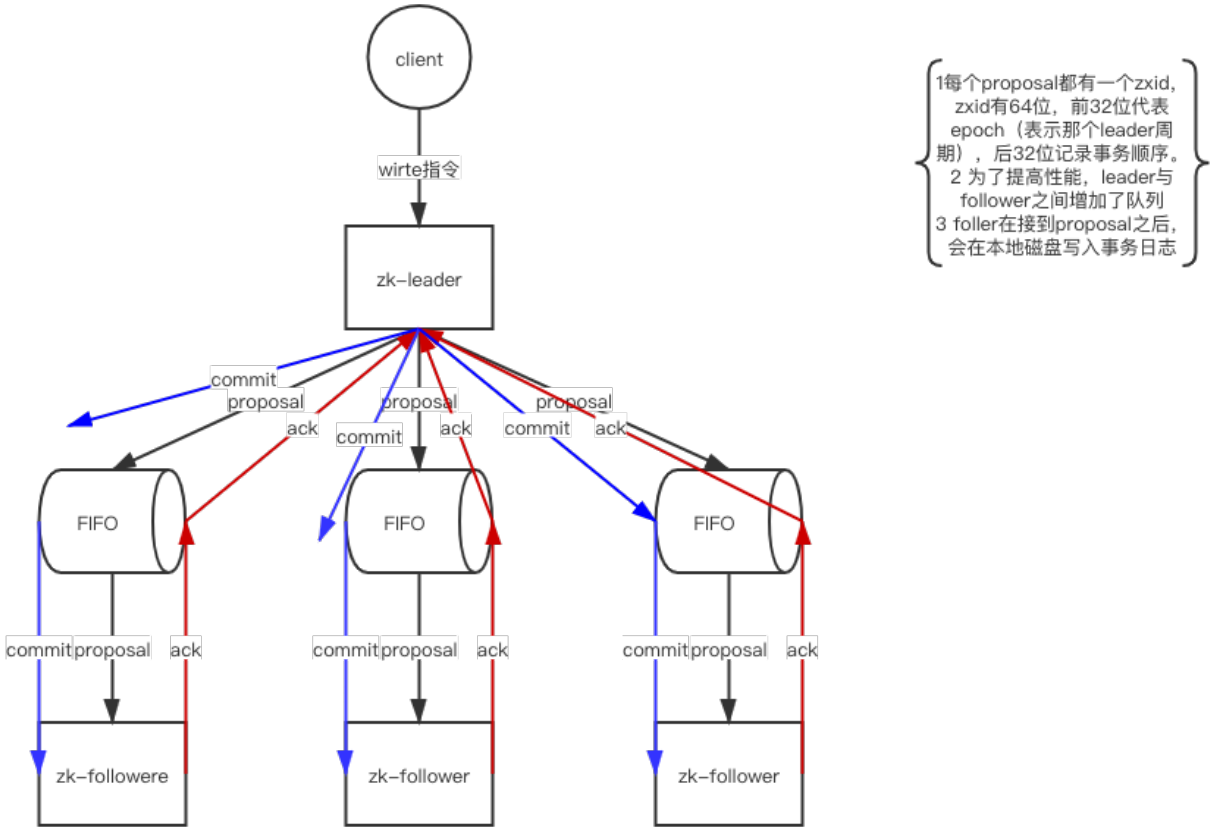


3 zk为了保证数据一致性，采用了zab协议（支持崩溃恢复的原子广播协议zookeeper automic broadcast）。

zab协议要求所有的写操作必须通过leader完成，leader写入本地日志后同步到所有的follower。

4 Zab协议为了解决上图中1、2两个问题，设计了两种模式

1 消息广播模式（类似与事务的两阶段提交），解决问题1



2 崩溃恢复模式，解决问题2(包含两个阶段，一leader选举，二数据恢复)

可以设置选举算法，目前使用的是fastleaderelection算法

服务器选举时会发送以下数据：

1 logicclock 每个服务器会维护一个自增的整数，表示的是第几轮投票

2 state 当前服务的状态 （locking\following\leading\observing）

3 self_id 当前服务器的myid

4 self_zxid 当前服务器的保存的最大的zxid

5 vote_id 被推荐服务器推荐的的myid

6 vote_zxid 被推荐服务器上保存的最大zxid

投票流程

1 自增logicclock

2 初始化投票：每个服务器在广播自己的选票前，会将自己的投票箱晴空。该投票想记录了所有的选票。例如：服务器2投给服务器3，服务器3投票给服务器1，则服务器1的投票箱为(2,3),(1,3),(1,1)。票箱中只会记录每一投票者的最后一票，如投票者更新自己的选票，则其他服务器收到该新选票后会在自己的票箱中更新该服务器的选票。

3 发送投票结果

4 接收投票结果：接收其他节点投票结果并记录自己的投票箱内。

5 判定投票结果：

5.1 判定选举轮次：收到投票结果之后，现根据投票信息中包含的logicclock进行不同的处理

5.1.1 如果外部投票的logicclock大于自己的logicClock，说明该服务器的选举轮次落后于其他服务器，立即清空自己的投票箱，并将自己的logicclock更新为收到的logicclock，然后再对比自己之前的投票与收到的投票以确定是否需要变更自己的投票，最终再次将自己的投票广播出去。

5.1.2 如果外部投票的logicclock小与自己的logicclock。则直接忽略该投票，继续处理下一个投票。

5.1.3 如果相等，则进行选票pk

5.2 选票pk

5.2.1 对比vote_zxid，如果外部投票的vote_zxid比较大，则把自己投票中的vote_id和vote_zxid更新为收到的票中的vote_id和vote_zxid并广播出去，并把接收到的票及自己更新后的票放到自己的票箱内。

5.2.2 如果vote_zxid一致，则比较二者的vote_id。

6 统计选票

如果已经有超过半数服务器任何了自己的投票，则投票终止。否则，继续接受其他服务器的投票。

7 更新服务器状态

投票结束后，服务器更新状态。如果超过半数投给自己，则将自己的服务器状态变成leading，否则变成following.

zk集群选主的场景：

1 集群启动选主

2 follower重启

3 leader重启

数据恢复(一致性保证)

zab协议保证再leader选举过程中，已经被commit的消息不会丢失，未被commit的数据对客户端不可见。

如何保证commit的消息不会丢失：

新的leader选出之后，所有的follower会将自己最大的zxid发给leader，leader会将leader的zxid与自身zxid之间所有被commit的消息同步给follower。同步完数据之后，leader会向所有的follower发送newleader命令等待服务器的ack，等收到超过半数服务的ack之后，向所有服务器广播uptodate命令，收到该命令的服务器可对外提供服务。

如果保证未commit的消息对客户端不可见

新的leader选出之后，先判断自身是否有未commint的消息，如果有，看下这个未commit的消息是否存在与大多数服务器中，从而决定是否要将其commit。然后leader可以得到已经被commit过的最小的zxid（min_zxid）与最大的zxid（max_zxid）。所有的follower向leader发送自身最大的commit过的最大的zxid（max_zxid）以及未被commit的所有消息（zxid_set）.根据这些信息作出如下操作：

如果max_zxid一致，说明完全同步，无须任何处理

如果follower的max_zxid小于leader的zxid，leader会通过trunc命令通知follower将其zxid_set中大于max_zxid的所有消息删除。

zk的每个节点称为/znode，由三部分组成：1 state(状态信息：如权限、版本等) 2 data(数据) 3 child(子节点)

ps：zk的心跳连接是master发送到salve的心跳

zk的服务器个数必须是奇数

zk接到数据之后先写本地，进行数据持久化

curator-封装的zk操作客户端

Zk的客户端和服务端是基于tcp长连接（tcp长连接是基于心跳机制保障的），

每个客户端连接，zk都会给它分配一个全局唯一的sessionid，每个客户端都有一个**Timeout**：会话超时时间，**TickTime**：下次会话超时时间点，默认 2000 毫秒，**isClosing**：该属性标记一个会话是否已经被关闭，当 server 端检测到会话已经超时失效，该会话标记为"已关闭"，不再处理该会话的新请求。

zookeeper 的 leader 服务器再运行期间定时进行会话超时检查，时间间隔是 ExpirationInterval，单位是毫秒，默认值是 tickTime，每隔 tickTime 进行一次会话超时检查。

常用命令

ls path

get path [watch]

create [-s] [-e] path data -s 和 -e 都是可选的，-s 代表顺序节点， -e 代表临时节点，注意其中 -s 和 -e 可以同时使用的，并且临时节点不能再创建子节点

set path data [version]

delete path [version]