Zookeeper分布式系统开发实战 第5课

DATAGURU专业数据分析社区

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

第五讲 ZK选举---ZAB协议

- ZooKeeper Atomic Broadcast 即ZooKeeper原子消息广播协议，简称为ZAB
  - 选举过程需要依赖此协议
  - 数据写入过程也需要此协议
  - Zab的核心是定义了那些会改变zk服务器数据状态的事务请求的处理方式

所有事务请求必须由一个全局唯一的服务器来协调处理，这样的服务器被称为Leader服务器，而余下的其它服务器则成为Follower服务器。Leader服务器负责将一个客户端事务请求转换成那个一个事务Proposal（提议），并将该Proposal分发给集群中所有的Follower服务器。之后Leader服务器需要等待所有Follower服务器的反馈，一旦超过半数的Follower服务器进行了正确的反馈后，那么Leader就会再次向所有的Follower服务器分发Commit消息，要求其将前一个Proposal进行提交

第五讲 ZK选举---ZAB协议

- ZAB协议三阶段
  - 发现（Discovery），即选举Leader过程
  - 同步（Synchronization），选举出新的Leader后，Follwer或者Observer从Leader同步最新的数据
  - 广播，同步完成后，就可以接收客户端新的事务请求，并进行消息广播，实现数据在集群节点的副本存储

第五讲 ZK选举---服务器角色

- Leader
  - 事务请求的唯一调度和处理者，保证集群事务处理的顺序性
  - 集群内部各服务器的调度者
- Follower
  - 处理客户端非事务请求，转发事务请求给Leader服务器
  - 参与事务请求Proposal的投票
  - 参与Leader选举投票
- Observer
  - 处理客户端非事务请求，转发事务请求给Leader服务器
  - 不参与任何形式的投票，包括选举和事务投票（超过半数确认）
  - 此角色存在通常是为了提高读性能

第五讲 ZK选举---服务器状态

- LOOKING
  - 寻找Leader状态
  - 当服务器处于此状态时，表示当前没有Leader，需要进入选举流程
- FOLLOWING
  - 跟随者状态，表明当前服务器角色是Follower
- OBSERVING
  - 观察者状态，表明当前服务器角色为Observer
- LEADING
  - 领导者状态，表明当前服务器角色为Leader

org.apache.zookeeper.server.quorum. ServerState类维护以上四种状态

第五讲 ZK选举---集群通信

- 基于TCP协议
  - 为了避免重复创建两个节点之间的tcp连接，zk按照myid数值方向来建立连接，即小数的节点发起大的节点连接，比如id为1的向id为2的发起tcp连接

- 多端口
  - 配置中第一个端口是通信和数据同步端口，默认是2888
  - 第二个端口是投票端口，默认是3888

第五讲 ZK选举---选举算法

- LeaderElection
  - Udp协议
- FastLeaderElection
  - Udp
  - Tcp
- AuthFastLeaderElection
  - udp

从3.4.0版本后，只支持FastLeaderElection的tcp协议版本的选举算法

第五讲 ZK选举---触发时机

- 集群启动
  - 寻找Leader状态
  - 当服务器处于此状态时，表示当前没有Leader，需要进入选举流程

- 崩溃恢复
  - Leader宕机
  - 网络原因导致过半节点与Leader心跳中断

第五讲 ZK选举---影响成为Leader因素

- 数据新旧程度
  - 只有拥有最新数据的节点才能有机会成为Leader
  - 通过事务id（zxid）的大小来表示数据的新旧，越大代表数据越新
- myid
  - 集群启动时，会在data目录下配置myid文件，里面的数字代表当前zk服务器节点的编号
  - 当zk服务器节点数据一样新时，myid中数字越大的就会被选举成为Leader
  - 当集群中已经有Leader时，新加入的节点不会影响原来的集群
- 投票数量
  - 只有得到集群中多半的投票，才能成为Leader
  - 多半即：n/2+1,其中n为集群中的节点数量

第五讲 ZK选举---zxid的构成

- 主进程周期
  - 也叫epoch
  - 选举的伦次，每多一次选举，则主进程周期加一
  - Zxid总共64位来表示，其高32位代表主进程周期
  - 比较数据新旧的时候，先比较epoch的大小

- 事务单调递增的计数器
  - zxid的低32位表示，选举完成后，从0开始

第五讲 ZK选举---初次启动

- 三个zk节点，都没有数据，对应的myid为1，2，3
  - 第一步：启动myid为1的节点，此时zxid为0，此时没法选举出主节点
  - 第二步：启动myid为2的节点，它的zxid也为0，此时2这个节点成为主节点
  - 第三步：启动myid为3的节点，因为已经有主节点，则3加入集群，2还是leader

第五讲 ZK选举---运行过程

- 场景说明
  - 3台机器，此时server2为主，并且server2宕机
- 选举流程
  - 变更状态
    - 当leader宕机后，其它节点的状态变更为LOOKING
  - 每个server发出一个投自己的票的投票
    - 生成投票信息（myid，ZXID）
    - 假定：server1为（1,123），server3为（2，122）
    - Server1发给server3，server3发给server1
  - 接收投票

第五讲 ZK选举---运行过程

- 选举流程—接着上一页
  - 投票处理
    - server3收到server1，因为server1的123比122大，所以，server3修改自己的投票为（1，123）然后发给server1
    - Server1收到server3的投票，因为123大于122，因此不改变自己的投票
  - 统计投票
    - Server3统计：自己收到投票（包括自己投的）中，（1，123）是两票
    - Server1统计：自己收到投票（包括自己投的）中，（1，123）是两票
  - 修改服务器状态
    - Server3，选出的leader是1，而自己是3，因此自己进入followering状态，即follower角色
    - Server1，选出的leader是1，自己就是1，因此自己进入LEADING状态，即自己是leader角色

第五讲 ZK选举---同步

- 同步时机
  - 当leader完成选举后，follower需要与新的leader同步数据
- 同步准备—leader
  - Leader告诉其它follower当前最新数据是什么即zxid
    - Leader构建一个NEWLEADER的包，包括当前最大的zxid，发送给所有的follower或者Observer
  - Leader给每个follower创建一个线程LearnerHandler来负责处理每个follower的数据同步请求，同时主线程开始阻塞，只有超过一半的follwer同步完成，同步过程才完成，leader才能成为真正的leader
  - 根据同步算法进行操作，详见后面ppt
- 同步准备—follower端
  - 选举完成后，尝试与leader建立同步连接，如果一段时间没有连接上就报错超时，重新回到选举状态
  - 向leader发送FOLLOWERINFO封包，带上follower自己最大的zxid
  - 根据不同同步算法进行操作，详见后面场景描述

第五讲 ZK选举---同步

- 初始化
  - minCommittedLog:最小的事务日志id，即zxid（没有被快照存储的日志文件的第一条，每次快照存储完，会重新生成一个事务日志文件）
  - maxCommittedLog：事务日志中最大的事务，即zxid
- 同步算法
  - 直接差异化同步（DIFF同步）
  - 仅回滚同步（TRUNC），即删除多余的事务日志，比如原来的 主宕机后又重新加入，可能存在它自己写入提交但是别的节点还没来得及提交
  - 先回滚再差异化同步（TRUNC+DIFF同步）
  - 全量同步（SNAP同步）
  - 同步算法举例：见后页

第五讲 ZK选举---同步

- 场景一
  - 把Follower最后的事务zxid称做peerLastZxid
  - 当minCommittedLog < peerLastZxid < maxCommittedLog

- 同步方案
  - 直接差异化同步
  - Leader会给follower服务器发送DIFF指令，意思是：进入差异化数据同步阶段，leader会把proposal同步给follower
  - 实际同步过程会先发送数据修改proposal，然后再发送COMMIT指令数据包

第五讲 ZK选举---同步

- 举例说明
  - 某个时刻Leader服务器未proposal队列对应的ZXID依次是 0x500000001 0x500000002 0x500000003 0x500000004 0x500000005
  - 此时follower的peerLastZxid为0x500000003，因此需要把0x500000004 0x500000005同步给follower
  - 差异化同步的消息发送顺序如下

发送顺序	数据包类型	对应的ZXID
1	PROPOSAL	0x500000004
2	COMMIT	0x500000004
3	PROPOSAL	0x500000005
4	COMMIT	0x500000005

第五讲 ZK选举---同步

- 举例说明
  - Follower端收到Diff指令，然后进入DIFF同步阶段
  - Follower收到同步的数据和提交命令，并应用到内存数据库中
  - 同步完成后
    - Leader会发送一个NEWLEADER指令，通知follower已经将最新的数据都同步给follower了
    - Follower收到NEWLEADER指令后悔反馈一个ACK消息，表明自己已经完成同步
  - 单个follower的同步完成，Leader进入集群的“过半策略”等待状态
  - 当有超过一半的follower都同步完成后，leader会向已经完成同步的follower发送UPTODATE指令，用于通知follower已经完成数据同步，可以对外提供服务了
  - Follower收到leader的UPTODATE指令后，会终止数据同步流程，向Leader再次反馈一个ACK消息

第五讲 ZK选举---同步

- 场景二
  - Leader在提交本地事务完成，还没有把事务Proposal提交给其它节点前，leader宕机了
  - 假设3个节点的集群，分别是A,B,C；没有宕机前，leader是B，已经发送过0X500000001和0X500000002的数据和事务提交proposal，并且发送了0X500000003的数据修改提议，但是在B节点发送事务提交的proposal之前，B宕机了，由于B是本机发送，所以B的本地事务已经提交，即B最新的数据是0X500000003
  - 在A和C进行选举后，C成为主，并且进行过两次数据修改，对应的Proposal是0X600000001 0X600000002
  - B机器恢复后加入新集群（AC），重新进行数据同步，对于B来说，peerLastZxid为0X500000003；对于当前的主C来说， minCommittedLog= 0X500000001 maxCommittedLog= 0X600000002

第五讲 ZK选举---同步

- 同步方案
  - B恢复后，并且向已有的集群（AC）注册后，向C发起同步连接请求
  - B向leader（C）发送FOLLOWERINFO封包，带上follower自己最大的zxid
  - C发现B上有没有自己的事务提交记录（ 0X500000003 ），则向B发送TRUNC命令，让B回滚到0X500000002
  - B回滚完成后，向C发送信息包，确认完成，并说明当前的zxid为0X500000002
  - C向B发送DIFF同步命令
  - B收到DIFF命令后进入同步状态，并向C发送ACK确认包
  - C陆续把对应的差异数据和Commit提交proposal发送给B，当数据发送完成后，再发送通知包给B
  - B应用用于内存的数据结构，当收到C通知已经完成同步后，B给回应ACK，并且结束同步

第五讲 ZK选举---同步

- 场景三
  - 某个节点宕机时间太长，当恢复并且加入集群后，数据的事务日志文件已经生成多个，此时的minCommittedLog比节点宕机时的最大日志还要大
  - 假设B宕机后，几天后才恢复，此时minCommittedLog为0X6000008731，而peerLastZxid为0X500000003
- 同步方案
  - 采用全量同步（SNAP）
  - 当leader（C）发现，B的zxid小于minCommittedLog时，向B发送SNAP指令
  - B收到指令，进入同步阶段
  - Leader(C)会从内存数据库中获取全量的数据发送给B，
  - B获取数据处理完成后，C还会把最新的proposal（全量同步期间产生）通过增量的方式发送给B

第五讲 ZK选举---广播

- 广播流程
  - 集群选举完成，并且完成数据同步后，即可开始对外服务，接收读写请求
  - 当leader接收到客户端饿新的事务请求后，会生成对应的事务proposal，并根据zxid的顺序向所有的follower发送提案，即proposal
  - 当follower收到leader的事务proposal时，根据接收的先后顺序处理这些proposal，即如果先后收到1,2,3条，则如果处理完了第3条，则代表1,2两条一定已经处理成功
  - 当Leader收到follower针对某个事务proposal过半的ack后，则发起事务提交，重新发起一个commit的proposal
  - Follower收到commit的proposal后，记录事务提交，并把数据更新到内存数据库
  - 补充说明
    - 由于只有过半的机器给出反馈，则可能存在某时刻某些节点数据不是最新的
    - 业务上如果需要确定读取到的数据是最新的，则可以在读取之前，调用sync方法进行数据同步

Thanks

FAQ时间