

Zookeeper——一致性协议:Zab协议



_Zy 关注

9 2018.06.28 10:47:49 字数 5,268 阅读 63,233

声明：本文写的时候，当时就是完全不懂zk，边看网上的文章边学习归纳和整理，这不是我的产出，不用点赞打赏。大家理智友善的讨论，有错误欢迎指出。
不过我最近不怎么上简书了。可能没来得及改。
如果想系统性的学习，推荐大家去看一下《从Paxos到Zookeeper 分布式一致性原理与实践》

- 什么是Zab协议
- Zab 协议的作用
- Zab 协议原理
- Zab 协议核心
- Zab 协议内容
- 原子广播
- 崩溃恢复
- 如何保证数据一致性
- Zab 协议如何数据同步
- 如何处理需要丢弃的 Proposal
- Zab 协议实现原理
- 选主过程

什么是Zab协议？

Zab协议 的全称是 **Zookeeper Atomic Broadcast**（Zookeeper原子广播）。
Zookeeper 是通过 **Zab** 协议来保证分布式事务的最终一致性。

1. Zab协议是为分布式协调服务Zookeeper专门设计的一种 **支持崩溃恢复** 的 **原子广播协议**，是Zookeeper保证数据一致性的核心算法。Zab借鉴了Paxos算法，但又不像Paxos那样，是一种通用的分布式一致性算法。它是特别为Zookeeper设计的支持崩溃恢复的原子广播协议。
2. 在Zookeeper中主要依赖Zab协议来实现数据一致性，基于该协议，zk实现了一种主备模型（即Leader和Follower模型）的系统架构来保证集群中各个副本之间数据的一致性。
这里的主备系统架构模型，就是指只有一台客户端（Leader）负责处理外部的写事务请求，然后Leader客户端将数据同步到其他Follower节点。

Zookeeper 客户端会随机的链接到 zookeeper 集群中的一个节点，如果是读请求，就直接从当前节点中读取数据；如果是写请求，那么节点就会向 Leader 提交事务，Leader 接收到事务提交，会广播该事务，只要超过半数节点写入成功，该事务就会被提交。

Zab 协议的特性：

- 1) Zab 协议需要确保那些已经在 **Leader 服务器**上提交（Commit）的事务最终被所有的服务器提交。
- 2) Zab 协议需要确保丢弃那些只在 **Leader** 上被提出而没有被提交的事务。

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

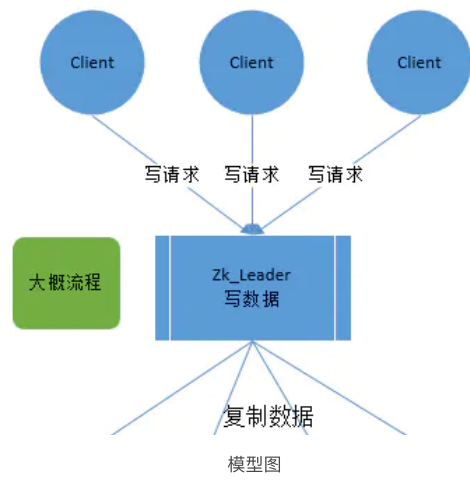
Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564

LeetCode 110. Balanced Binary Tree





推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564

Zab 协议实现的作用

- 1) 使用一个单一的主进程（Leader）来接收并处理客户端的事务请求（也就是写请求），并采用了Zab的原子广播协议，将服务器数据的状态变更以 **事务proposal**（事务提议）的形式广播到所有的副本（Follower）进程上去。
- 2) 保证一个全局的变更序列被顺序引用。
Zookeeper是一个树形结构，很多操作都要先检查才能确定是否可以执行，比如P1的事务t1可能是创建节点"/a"，t2可能是创建节点"/a/bb"，只有先创建了父节点"/a"，才能创建子节点"/a/b"。
- 为了保证这一点，Zab要保证同一个Leader发起的事务要按顺序被apply，同时还要保证只有先前Leader的事务被apply之后，新选举出来的Leader才能再次发起事务。
- 3) 当主进程出现异常的时候，整个zk集群依旧能正常工作。



Zab协议原理

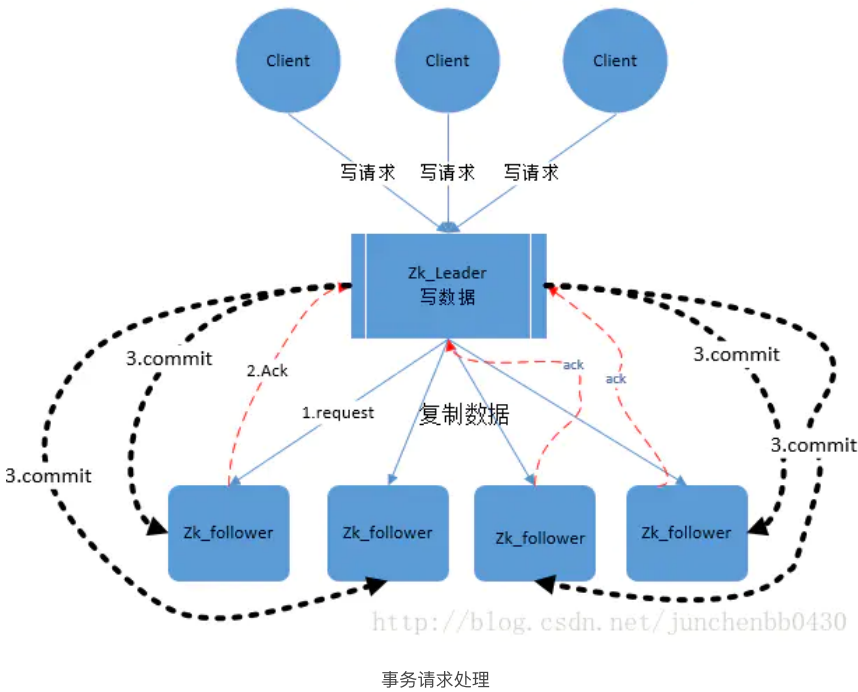
Zab协议要求每个 Leader 都要经历三个阶段：**发现，同步，广播**。

- **发现**：要求zookeeper集群必须选举出一个 Leader 进程，同时 Leader 会维护一个 Follower 可用客户端列表。将来客户端可以和这些 Follower节点进行通信。
- **同步**：Leader 要负责将本身的数据与 Follower 完成同步，做到多副本存储。这样也是提现了CAP中的高可用和分区容错。Follower将队列中未处理完的请求消费完成后，写入本地事务日志中。
- **广播**：Leader 可以接受客户端新的事务Proposal请求，将新的Proposal请求广播给所有的 Follower。

Zab协议核心

Zab协议的核心：定义了事务请求的处理方式

- 1) 所有的事务请求必须由一个全局唯一的服务器来协调处理，这样的服务器被叫做 **Leader服务器**。其他剩余的服务器则是 **Follower服务器**。
- 2) Leader服务器 负责将一个客户端事务请求，转换成一个 **事务Proposal**，并将该 Proposal 分发给集群中所有的 Follower 服务器，也就是向所有 Follower 节点发送数据广播请求（或数据复制）
- 3) 分发之后Leader服务器需要等待所有Follower服务器的反馈（Ack请求），**在Zab协议中，只要超过半数的Follower服务器进行了正确的反馈后**（也就是收到半数以上的Follower的Ack请求），那么 Leader 就会再次向所有的 Follower服务器发送 Commit 消息，要求其将上一个 事务proposal 进行提交。



推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



Zab协议内容

Zab 协议包括两种基本的模式：**崩溃恢复** 和 **消息广播**

协议过程

当整个集群启动过程中，或者当 Leader 服务器出现网络中弄断、崩溃退出或重启等异常时，Zab协议就会 **进入崩溃恢复模式**，选举产生新的Leader。

当选举产生了新的 Leader，同时集群中有过半的机器与该 Leader 服务器完成了状态同步（即数据同步）之后，Zab协议就会退出崩溃恢复模式，**进入消息广播模式**。

这时，如果有一台遵守Zab协议的服务器加入集群，因为此时集群中已经存在一个Leader服务器在广播消息，那么该新加入的服务器自动进入恢复模式：找到Leader服务器，并且完成数据同步。同步完成后，作为新的Follower一起参与到消息广播流程中。

协议状态切换

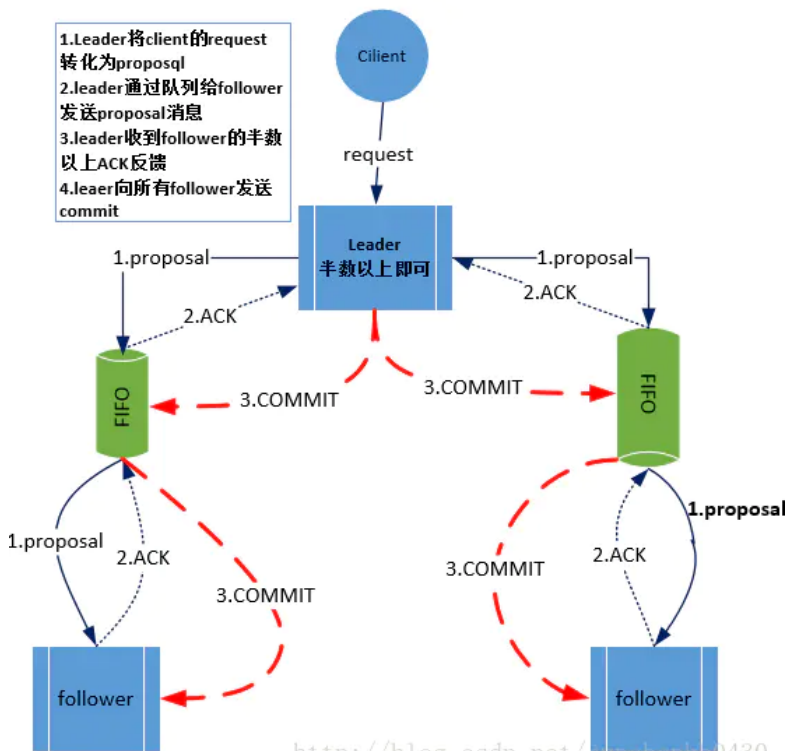
当Leader出现崩溃退出或者机器重启，亦或是集群中不存在超过半数的服务器与Leader保存正常通信，Zab就会再一次进入崩溃恢复，发起新一轮Leader选举并实现数据同步。同步完成后又会进入消息广播模式，接收事务请求。

保证消息有序

在整个消息广播中，Leader会将每一个事务请求转换成对应的 proposal 来进行广播，并且在广播 事务Proposal 之前，Leader服务器会首先为这个事务Proposal分配一个全局单递增的唯一ID，称之为事务ID（即zxid），由于Zab协议需要保证每一个消息的严格的顺序关系，因此必须将每一个proposal按照其zxid的先后顺序进行排序和处理。

消息广播

- 1) 在zookeeper集群中，数据副本的传递策略就是采用消息广播模式。zookeeper中农数据副本的同步方式与二段提交相似，但是却又不同。二段提交要求协调者必须等到所有的参与者全部反馈ACK确认消息后，再发送commit消息。要求所有的参与者要么全部成功，要么全部失败。二段提交会产生严重的阻塞问题。
- 2) Zab协议中 Leader 等待 Follower 的ACK反馈消息是指“只要半数以上的Follower成功反馈即可，不需要收到全部Follower反馈”



<http://blog.csdn.net/junchenbb0430>

消息广播流程图

消息广播具体步骤

- 1) 客户端发起一个写操作请求。

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



- 2) Leader 服务器将客户端的请求转化为事务 Proposal 提案，同时为每个 Proposal 分配一个全局的ID，即zxid。
- 3) Leader 服务器为每个 Follower 服务器分配一个单独的队列，然后将需要广播的 Proposal 依次放到队列中取，并且根据 FIFO 策略进行消息发送。
- 4) Follower 接收到 Proposal 后，会首先将其以事务日志的方式写入本地磁盘中，写入成功后向 Leader 反馈一个 Ack 响应消息。
- 5) Leader 接收到超过半数以上 Follower 的 Ack 响应消息后，即认为消息发送成功，可以发送 commit 消息。
- 6) Leader 向所有 Follower 广播 commit 消息，同时自身也会完成事务提交。Follower 接收到 commit 消息后，会将上一条事务提交。

zookeeper 采用 Zab 协议的核心，就是只要有一台服务器提交了 Proposal，就要确保所有的服务器最终都能正确提交 Proposal。这也是 CAP/BASE 实现最终一致性的一个体现。

Leader 服务器与每一个 Follower 服务器之间都维护了一个单独的 FIFO 消息队列进行收发消息，使用队列消息可以做到异步解耦。Leader 和 Follower 之间只需要往队列中发消息即可。如果使用同步的方式会引起阻塞，性能要下降很多。

崩溃恢复

一旦 Leader 服务器出现崩溃或者由于网络原因导致 Leader 服务器失去了与过半 Follower 的联系，那么就会进入崩溃恢复模式。

在 Zab 协议中，为了保证程序的正确运行，整个恢复过程结束后需要选举出一个新的 Leader 服务器。因此 Zab 协议需要一个高效且可靠的 Leader 选举算法，从而确保能够快速选举出新的 Leader 。

Leader 选举算法不仅仅需要让 Leader 自己知道自己已经被选举为 Leader，同时还需要让集群中的所有其他机器也能够快速感知到选举产生的新 Leader 服务器。

崩溃恢复主要包括两部分：**Leader选举** 和 **数据恢复**

Zab 协议如何保证数据一致性

假设两种异常情况：

- 1、一个事务在 Leader 上提交了，并且过半的 Follower 都响应 Ack 了，但是 Leader 在 Commit 消息发出之前挂了。
- 2、假设一个事务在 Leader 提出之后，Leader 挂了。

要确保如果发生上述两种情况，数据还能保持一致性，那么 Zab 协议选举算法必须满足以下要求：

Zab 协议崩溃恢复要求满足以下两个要求：

- 1) 确保已经被 Leader 提交的 Proposal 必须最终被所有的 Follower 服务器提交。
- 2) 确保丢弃已经被 Leader 提出的但是没有被提交的 Proposal。

根据上述要求

Zab协议需要保证选举出来的Leader需要满足以下条件：

- 1) 新选举出来的 Leader 不能包含未提交的 Proposal 。

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



即新选举的 Leader 必须都是已经提交了 Proposal 的 Follower 服务器节点。

2) 新选举的 Leader 节点中含有最大的 zxid 。

这样做的好处是可以避免 Leader 服务器检查 Proposal 的提交和丢弃工作。

Zab 如何数据同步

1) 完成 Leader 选举后（新的 Leader 具有最高的zxid），在正式开始工作之前（接收事务请求，然后提出新的 Proposal），Leader 服务器会首先确认事务日志中的所有的 Proposal 是否已经被集群中过半的服务器 Commit。

2) Leader 服务器需要确保所有的 Follower 服务器能够接收到每一条事务的 Proposal，并且能将所有已经提交的事务 Proposal 应用到内存数据中。等到 Follower 将所有尚未同步的事务 Proposal 都从 Leader 服务器上同步过啦并且应用到内存数据中以后，Leader 才会把该 Follower 加入到真正可用的 Follower 列表中。

Zab 数据同步过程中，如何处理需要丢弃的 Proposal

在 Zab 的事务编号 zxid 设计中，zxid是一个64位的数字。

其中低32位可以看成简单的单增计数器，针对客户端每一个事务请求，Leader 在产生新的 Proposal 事务时，都会对该计数器加1。而高32位则代表了 Leader 周期的 epoch 编号。

epoch 编号可以理解为当前集群所处的年代，或者周期。每次Leader变更之后都会在 epoch 的基础上加1，这样旧的 Leader 崩溃恢复之后，其他Follower 也不会听它的了，因为 Follower 只服从epoch最高的 Leader 命令。

每当选举产生一个新的 Leader，就会从这个 Leader 服务器上取出本地事务日志充最大编号 Proposal 的 zxid，并从 zxid 中解析得到对应的 epoch 编号，然后再对其加1，之后该编号就作为新的 epoch 值，并将低32位数字归零，由0开始重新生成zxid。

Zab 协议通过 epoch 编号来区分 Leader 变化周期，能够有效避免不同的 Leader 错误的使用了相同的 zxid 编号提出了不一样的 Proposal 的异常情况。

基于以上策略

当一个包含了上一个 Leader 周期中尚未提交过的事务 Proposal 的服务器启动时，当这台机器加入集群中，以 Follower 角色连上 Leader 服务器后，Leader 服务器会根据自己服务器上最后提交的 Proposal 来和 Follower 服务器的 Proposal 进行比对，比对的结果肯定是 Leader 要求 Follower 进行一个回退操作，回退到一个确实已经被集群中过半机器 Commit 的最新 Proposal。

实现原理

Zab 节点有三种状态：

- Following：当前节点是跟随者，服从 Leader 节点的命令。
- Leading：当前节点是 Leader，负责协调事务。
- Election/Looking：节点处于选举状态，正在寻找 Leader。

代码实现中，多了一种状态：Observing 状态

这是 Zookeeper 引入 Observer 之后加入的，Observer 不参与选举，是只读节点，跟 Zab 协

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



议没有关系。

节点的持久状态：

- history：当前节点接收到事务 Proposal 的Log
- acceptedEpoch：Follower 已经接受的 Leader 更改 epoch 的 newEpoch 提议。
- currentEpoch：当前所处的 Leader 年代
- lastZxid：history 中最近接收到的Proposal 的 zxid（最大zxid）

Zab 的四个阶段

1、选举阶段（Leader Election）

节点在一开始都处于选举节点，只要有一个节点得到超过半数节点的票数，它就可以当选准 Leader，只有到达第三个阶段（也就是同步阶段），这个准 Leader 才会成为真正的 Leader。

Zookeeper 规定所有有效的投票都必须在同一个轮次中，每个服务器在开始新一轮投票时，都会对自己维护的 logicalClock 进行自增操作。

每个服务器在广播自己的选票前，会将自己的投票箱（recvset）清空。该投票箱记录了所受到的选票。

例如：Server_2 投票给 Server_3，Server_3 投票给 Server_1，则Server_1的投票箱为 (2,3)、(3,1)、(1,1)。（每个服务器都会默认给自己投票）

前一个数字表示投票者，后一个数字表示被选举者。票箱中只会记录每一个投票者的最后一次投票记录，如果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。

这一阶段的目的是为了选出一个准 Leader，然后进入下一个阶段。

协议并没有规定详细的选举算法，后面会提到实现中使用的 Fast Leader Election。

推荐阅读

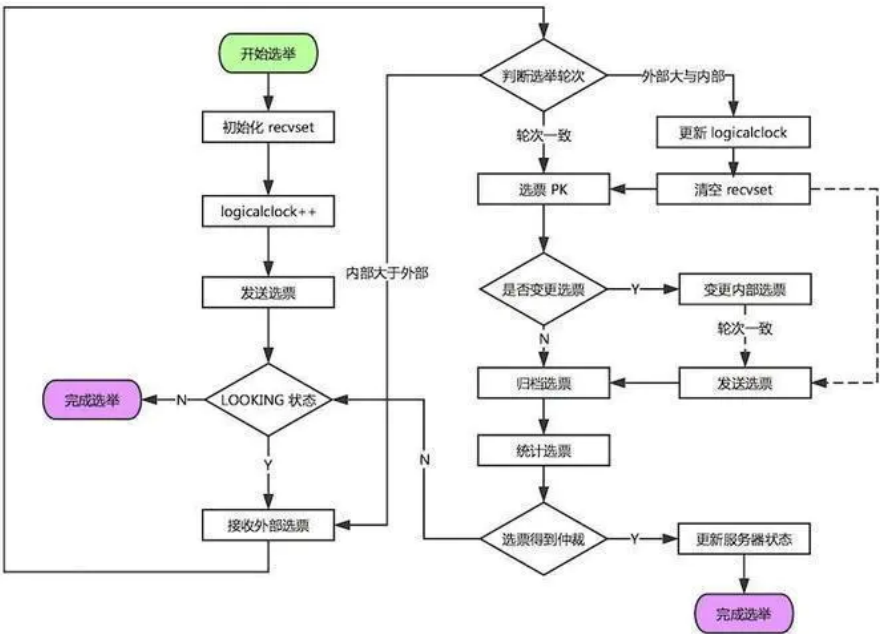
zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564

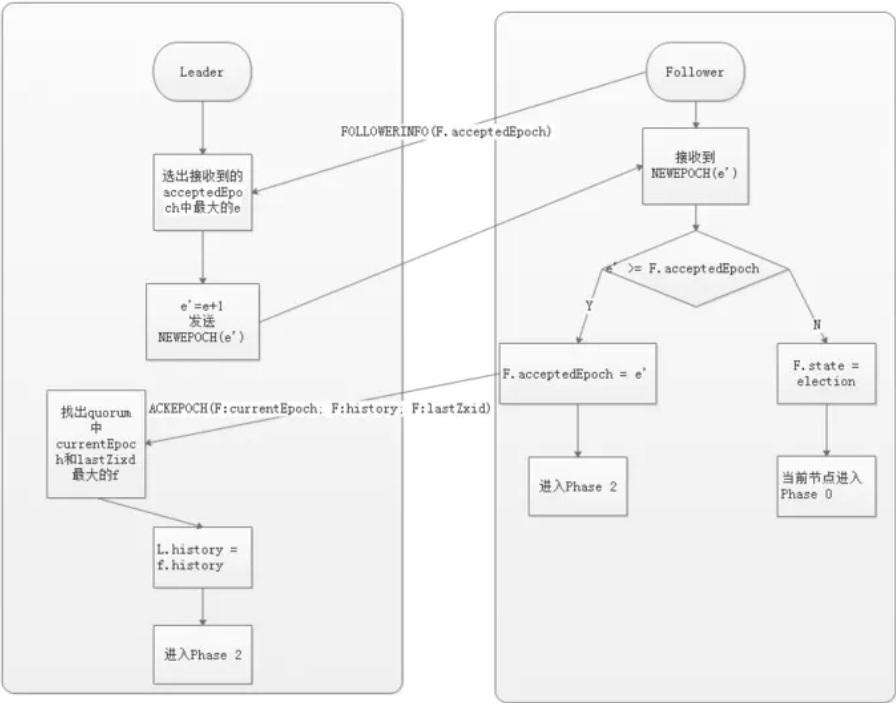


选举流程

2、发现阶段 (Discovery)

在这个阶段，Followers 和上一轮选举出的准 Leader 进行通信，同步 Followers 最近接收的事务 Proposal。
一个 Follower 只会连接一个 Leader，如果一个 Follower 节点认为另一个 Follower 节点，则会在尝试连接时被拒绝。被拒绝之后，该节点就会进入 Leader Election阶段。

这个阶段的主要目的是发现当前大多数节点接收的最新 Proposal，并且准 Leader 生成新的 epoch，让 Followers 接收，更新它们的 acceptedEpoch。



发现流程

3、同步阶段 (Synchronization)

同步阶段主要是利用 Leader 前一阶段获得的最新 Proposal 历史，同步集群中所有的副本。
只有当 quorum（超过半数的节点）都同步完成，准 Leader 才会成为真正的 Leader。
Follower 只会接收 zxid 比自己 lastZxid 大的 Proposal。

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行
主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是
如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564

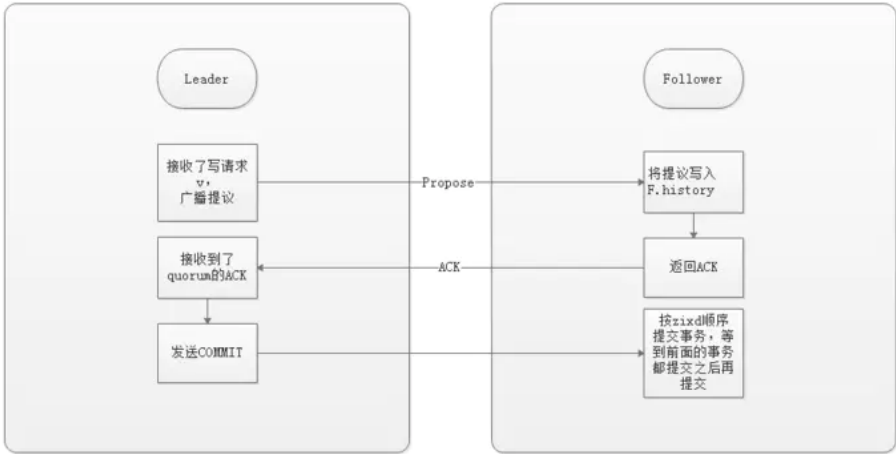




同步流程

4、广播阶段 (Broadcast)

到了这个阶段，Zookeeper 集群才能正式对外提供事务服务，并且 Leader 可以进行消息广播。同时，如果有新的节点加入，还需要对新节点进行同步。
需要注意的是，Zab 提交事务并不像 2PC 一样需要全部 Follower 都 Ack，只需要得到 quorum（超过半数的节点）的Ack 就可以。



广播流程

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



协议实现

协议的 Java 版本实现跟上面的定义略有不同，选举阶段使用的是 Fast Leader Election（FLE），它包含了步骤1的发现指责。因为FLE会选举拥有最新提议的历史节点作为 Leader，这样就省去了发现最新提议的步骤。

实际的实现将发现和同步阶段合并为 Recovery Phase（恢复阶段），所以，Zab 的实现实际上有三个阶段。

Zab协议三个阶段：

- 1) 选举 (Fast Leader Election)
- 2) 恢复 (Recovery Phase)
- 3) 广播 (Broadcast Phase)

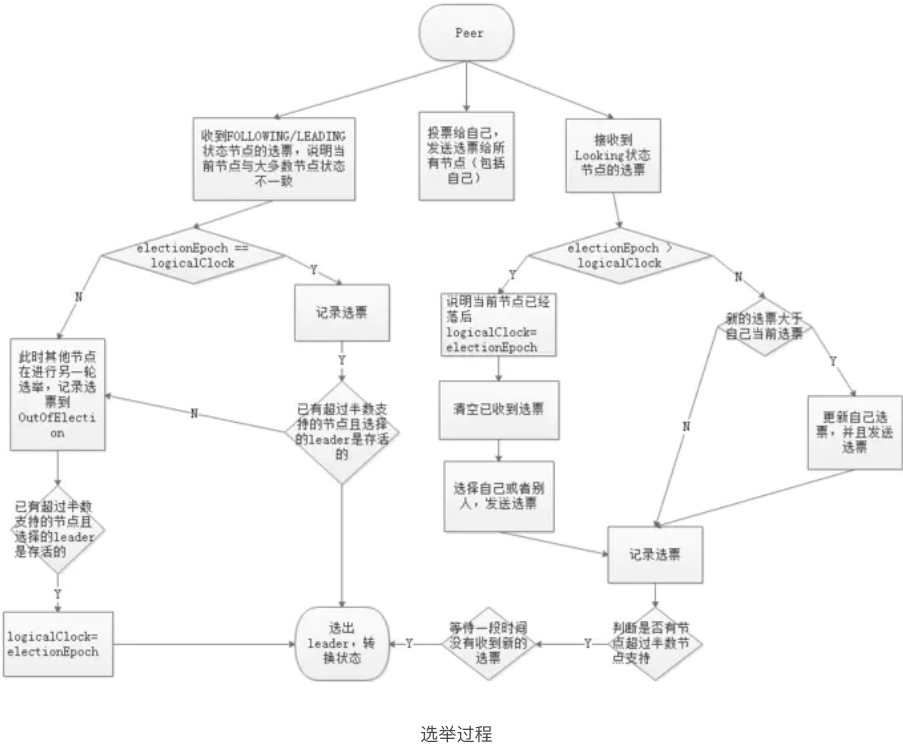
Fast Leader Election（快速选举）

前面提到的 FLE 会选举拥有最新Proposal history（lastZxid最大）的节点作为 Leader，这样就省去了发现最新提议的步骤。这是基于拥有最新提议的节点也拥有最新的提交记录

- 成为 Leader 的条件：
 - 1) 选 epoch 最大的
 - 2) 若 epoch 相等，选 zxid 最大的
 - 3) 若 epoch 和 zxid 相等，选择 server_id 最大的（zoo.cfg中的myid）



节点在选举开始时，都默认投票给自己，当接收其他节点的选票时，会根据上面的 Leader 条件判断并且更改自己的选票，然后重新发送选票给其他节点。当有一个节点的得票超过半数，该节点会设置自己的状态为 Leading，其他节点会设置自己的状态为 Following。



推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

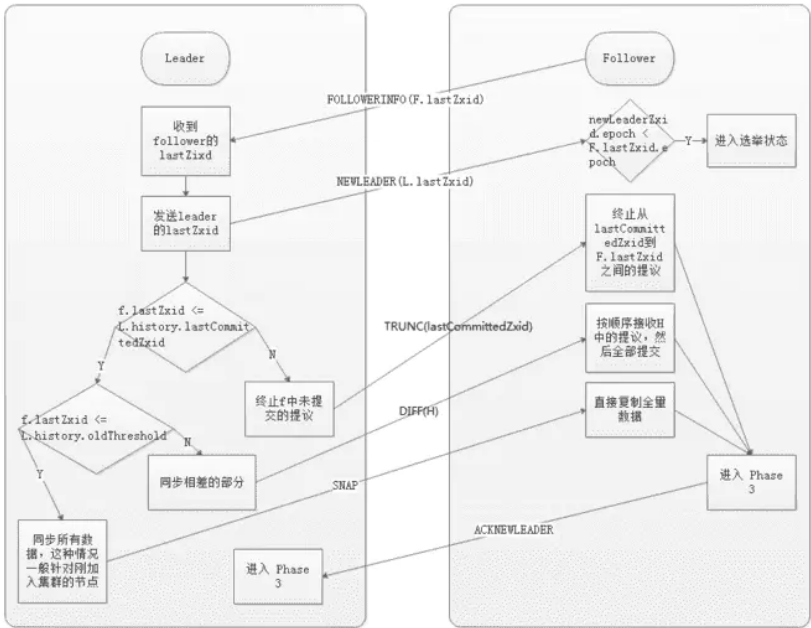
Kafka笔记
阅读 564



Recovery Phase（恢复阶段）

这一阶段 Follower 发送他们的 lastZxid 给 Leader，Leader 根据 lastZxid 决定如何同步数据。这里的实现跟前面的 Phase 2 有所不同：Follower 收到 TRUNC 指令会终止 L.lastCommittedZxid 之后的 Proposal，收到 DIFF 指令会接收新的 Proposal。


history.lastCommittedZxid：最近被提交的 Proposal zxid
history.oldThreshold：被认为已经太旧的已经提交的 Proposal zxid





恢复阶段


(如果有什么错误或者建议，欢迎留言指出)
(本文内容是对各个知识点的转载整理，用于个人技术沉淀，以及大家学习交流用)


- 参考资料：
- [Zookeeper中的ZAB协议理解](#)
 - [简书-Zookeeper之Zab协议](#)
 - [简书-Zookeeper Zab协议](#)
 - [一致性理论——一致性协议之zab协议](#)
 - [Zookeeper系列（5）--ZAB协议，消息广播，崩溃恢复，数据同步](#)
 - [《实例详解Zk的 Zab 协议分布式锁与领导选举》](#)
 - [Zab协议介绍](#)

 116人点赞 >




 分布式





_Zy 菜得抠脚的Java程序猿一枚

总资产12 共写了7.9W字 获得223个赞 共73个粉丝




怎样为自己设计签名



写下你的评论...

全部评论 37 只看作者

按时间倒序 按时间正序



_Zy 作者

14楼 2020.11.30 18:11

声明一句啊，我就是归纳整理。这些不是我写的。

好久没看了，这篇文章是被编辑过了？
我明明在第一行就写了，这全是基于网络上的整理和转载。
这就是我当做自己记录用，大家看个热闹就好。
有错误欢迎指正，也欢迎友善的交流。

我的个人说明就说了，我是菜鸡。还有我已经关闭赞赏了，大家不要赞赏了。
不好意思之前不懂简书的赞赏规则，完全没有想拿这些东西骗赞赏的。

推荐阅读

- [zookeeper面试常见问题](#)
阅读 291
- [Broker是如何基于Dledger技术进行主从同步的](#)
阅读 114
- [Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的](#)
阅读 334
- [4. ZooKeeper](#)
阅读 297
- [Kafka笔记](#)
阅读 564



👍 2 💬 回复



敢死队111
13楼 2020.11.27 15:00

leader 收到半数以上ACK之后，会发送commit, 如果follower网络抖动没有收到commit 呢？ 岂不是会出现的结果是只有非一部分结点(有可能不到半数)commit了

👍 1 💬 回复

SaaS软件排行榜



敢死队111
12楼 2020.11.27 14:30

在广播消息进行同步时，如果leader发现过半的follower都返回了ack,就会通知大家执行commit

这里有两个问题，1. 如果有些follower之前就没有接收到proposal, 怎么执行commit? 2. 如果接受到commit的机器有部分或者是全部都commit失败呢？就没有然后了么？

👍 赞 💬 回复



itoocs
11楼 2020.08.04 10:50

博主有一个疑问，在同步阶段，新选出的leader肯定是xid最大的了，为啥还会存在同步flower节点数据的过程，还是我理解错了，应该是flower同步leader的数据呀

👍 赞 💬 回复



nged
2020.09.09 16:34

很明显是Follower 去同步leader的数据

💬 回复

✍️ 添加新评论



丿醉...逍遥
7楼 2019.12.28 13:50

Zab协议原理：“这样也是提现了CAP中的高可用和分区容错”
这里ZAB支持CAP中的CP吧，没有高可用~

👍 赞 💬 回复



qlmmys
2020.02.24 19:54

半数Follower确认，Leader就会提交这个事务，可见并没有实现强一致性

💬 回复



6fd49755b0d4
2020.03.03 23:23

没有高可用，leader死了就不能提供服务了

💬 回复



b333770fc14d
2020.04.26 11:07

Zab协议保证了AP.

💬 回复

推荐阅读

zookeeper面试常见问题
阅读 291


Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



 添加新评论 | 还有3条评论, [查看更多](#)



人醉逍遥
6楼 2019.12.10 16:36



 赞  回复



9e8396a57f6a
5楼 2019.11.02 13:06

博主写的确实不错由浅入深

 赞  回复



编程大道
4楼 2019.08.13 18:14

请教一下，一个事务被leader广播后只收到过半ACK就认为可以commit，但如果那些没返回ACK的节点，是真的失败了，导致事务没有在这些节点最终提交，那么这个follower数据就不是新的，那这个follower的数据什么时候被同步？

 赞  回复



告白摩天轮_adonis
2019.08.19 21:00

重新选举leader的时候

 回复



6b99d5dfa16d
2019.11.06 11:44

这涉及的问题是zookeeper如何实现最终一致性的问题。如果是重新选举的时候就违背了zookeeper的最终一致性原则，因为从你的提的问题的条件判断，这个机器没有宕机没有网络延时只是commit失败，那么这个节点就会进入崩溃恢复阶段，而又存在leader故直接链接leader进行同步，如果一直commit失败那么运维人员就会看出端倪只能认为解决commit失败原因，当解决后将该节点加入集群，重新开始崩溃恢复然后实行follower的正常工作


 回复



6b99d5dfa16d
2019.11.06 11:45

@Mr_Stevens 人为

 回复

 添加新评论 | 还有4条评论, [查看更多](#)



507c52fdeff0
3楼 2019.06.11 21:24

ab 协议崩溃恢复要求满足以下两个要求：

1) 确保已经被 Leader 提交的 Proposal 必须最终被所有的 Follower 服务器提交。

如果上一个Leader在执行提交Proposal事务后，Leader挂了，当前所有的Follower都是待执行被提出的Proposal事务，都处于待提交Proposal事务状态，这个时候怎么满足这个重新选举Leader条件？

 赞  回复



告白摩天轮_adonis
2019.08.19 21:00

最终一致性

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114


Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564




回复

 CCQ_405d
2019.12.25 14:47


这个时候重新选leader的时候，这个Proposal是会自动提交吗？还是说等这个leader崩溃恢复的时候，会和当前leader同步这条proposal

回复

 1b88c3bdeb2a
2020.05.11 18:33

会选取follower已经提交的最大的xid，作为Leader

回复

 添加新评论 | 还有1条评论, [查看更多](#)

 无福卷毛
2楼 2019.05.21 10:58


有一个小问题 希望请教一下 客户端发送了一个事务到主服务器，主服务器还没来得及处理就挂掉了 这时候会怎么样 这个事务会丢掉吗 还是客户端抛出了异常

 赞  回复

 _Zy 作者
2019.05.22 15:28


Leader挂掉会进入恢复模式，发到Leader上的事务还没提交，那这个事务会被集群丢弃。恢复模式做的事情就是：1，确保已经提交的事务最终被集群中所有机器提交。2、丢弃那些只在Leader上提出的事务。需要保证集群中事务的最终一致性。

回复

 无福卷毛
2019.05.23 14:17

@_Zy 还有一个 就是一个watch触发了 还没来得及重新设置 被监控的节点就第二次发生了变化 这样不就丢掉了一次 怎么预防

回复

 88dfade28cb1
2019.08.30 17:04

@无福卷毛 zookeeper的监听通知并不会带上数据，仅仅只是通知客户端发生了变化，并且只会触发一次，比如你调用“getData获取数据为设置监听，此时获取到的数据为空，数据发生了变化，触发监听，你需要在回调函数中再次调用getData并设置监听，这样数据是不会丢失的。

回复

 无福卷毛
2019.09.09 10:13


@蔡勋适 非常感谢

回复

 superxcp
2020.03.21 11:38

写的真好，赞美

回复

 b333770fc14d
2020.04.24 10:37

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



@_Zy 如果集群中只有Leader提交了事务，Follower并没有提交事务，这时Leader挂了，如何确保Leader上被提交了的事务最终也被Follower提交呢？我认为重新选举出来的Leader上最后一个事务肯定没被提交，按道理这个未被提交的事务是会被丢弃的，既然都被丢弃了，那么如何保证旧leader上已提交的事务最终也被Follower提交呢？这是存在矛盾的，这个问题博主一直没说清楚。

回复

6952a5e01149
2020.06.03 01:52

@_Zy 怎么确定事务已经被提交？zookeeper本身没有回滚机制为啥已经发起的事务在超过半数节点已经ACK的情况下，新leader为啥不提交之前的事务。

回复

期许不经意间的美好
2020.09.12 11:12

写的挺好的，值得仔细阅读

回复

36eca4001fdf
2020.11.12 21:22

@_Zy 你除了复制粘贴还会干啥，说了跟没说一样

回复

添加新评论 | 收起

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564



被以下专题收入，发现更多相似内容

- Zookeeper
- zookeeper
- 分布式
- Zookeeper
- 框架原理
- 分布式系统
- zookeeper
- 展开更多

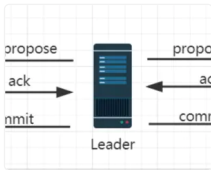
推荐阅读

更多精彩内容>

深入zookeeper之ZAB 数据一致性协议的原理

之前我更新了一篇文章是关于分布式数据一致性算法Paxos算法的，但是实际上zookeeper并没有使用这种算法作为...

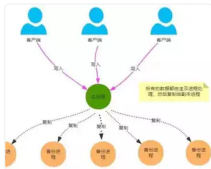
先生zeng 阅读 1,608 评论 0 赞 4



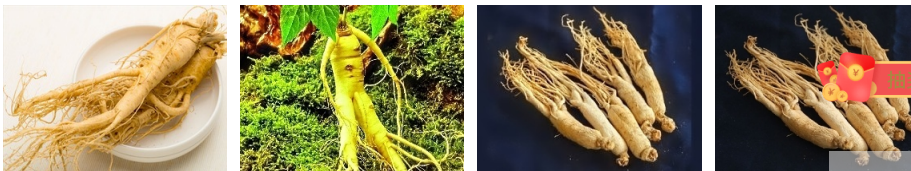
面试官问：ZooKeeper 一致性协议 ZAB 原理

一致性协议有很多种，比如 Paxos, Raft, 2PC, 3PC等等，今天我们讲一种协议，ZAB 协议，该协议应该是...

架构师springboot 阅读 447 评论 0 赞 15




人参食用方法与注意事项



11. 山有乔松隰有龙，不见子充见狡童

黑色的海岛上悬着一轮又大又圆的明月，毫不嫌弃地把温柔的月色照在这寸草不生的小岛上。一个少年白衣白发，悠闲自如地倚坐...

 小水Vivian 阅读 1,563 评论 1 赞 5


扯下窗帘，阳光倾泻而下。

渐变的面目拼图要我怎么拼？ 我是疲乏了还是投降了？ 不是不允许自己坠落， 我没有滴水不进的保护膜。 就是害怕变得面...

 闷热当乘凉 阅读 2,525 评论 0 赞 12

0906

感觉自己有点神经衰弱，总是觉得手机响了；屋外有人走过；每次妈妈不声不响的进房间突然跟我说话，我都会被吓得半死！一整...

 章鱼的拥抱 阅读 813 评论 1 赞 4

推荐阅读

zookeeper面试常见问题
阅读 291

Broker是如何基于Dledger技术进行主从同步的
阅读 114

Spring Cloud Alibaba——Nacos 是如何同时实现AP与CP的
阅读 334

4. ZooKeeper
阅读 297

Kafka笔记
阅读 564

