链滴

从PAXOS进化到ZAB协议

作者：skyesx

原文链接：https://ld246.com/article/1465827201253

来源网站：链滴

<h1 data-source-line="1">从PAXOS进化到ZAB协议</h1>
没错，看了 从PAXOS到ZOOKEEPER这本书，然后写一篇博文总结下 PAOS吧。
<h2 data-source-line="5">一、PAXOS简介</h2>
之前一篇文章就对PAXOS做了简单的介绍
PAXOS解决的问题是 在分布式环境中确定某一个 “不可变量X” 的值。
PAXOS有以下特性：
<ul data-source-line="12">
只要过半数Acceptor存活，那么就能正常的进行 X 值议案选举，并选定X的值
只要有Leaner存活，那么外部要么拿不到对应的值，要么拿到的值就是最终确定的
需要被确定的某个值一旦被确定了，那么就不会再变更
只要有一个proposer存活，那么就能持续的给Acceptor提出议案

具体的PAXOS算法过程请参考我的另外一篇文章 说人话的PAXOS算法简介
<h2 data-source-line="20">二、PAXOS算法的缺点</h2>
我们可以从PAXOS算法的过程发现，PAXOS算法有以下性能缺点
<ul data-source-line="22">
多个Proposer同时提议案会降低议案的抉择速度
PAXOS算法单个实例只能决定一个X的值，但通常运用场景中，我们需要确定多个X的值
Learner越多，Acceptor和Learner之间的通讯消耗会成指数级别上升

下面，我们来一一解决PAXOS的这些缺点
<h2 data-source-line="28">三、解决多Propser的问题</h2>
多proposer存在的意义是避免单点故障，实际上，正常时刻，我们同时间只需要一个proposer正常运作，其他proposer standby。因此,解决方案很简单，设计一个主备法即可。
选定主proposer的一个算法可以如下：
<ul data-source-line="34">
任意一个proposer觉得有必要发起主proposer选举时，就可以发起proposer选举
发起时，同时把自己的投票的结果 及 自己当前事件对应的逻辑时间 告诉所有proposer
当某个proposer获得过半的选票时，该proposer就成为了主proposer
proposer投票的依据是 每个proposer的当前逻辑时间。
拥有最大逻辑间的proposer将会被选举为主proposer

选定了主proposer后，议案的提议就全都交给了主proposer提高效率减少了议案选定过程中的通讯。
选定了主proposer后，当前处于主proposer就可以跳过 prepare promise这个步骤，直接提交议案。因为正常运行时，只有他会提出议案，因此，该proposer的 逻时钟一直都是最新的。若这个proposer失去了主proposer的地位时，他直接提出议案时，会收到acceptor的reject反馈，这意 味着有其他机器竞争主proposer，要重新进行选举
<h2 data-source-line="45">四、解决多PAXOS实例的需求</h2>
正常的运用场景中，我们需要确定多个未知议案的值。因此需要多个PAOS算法实例一起运作
我们回顾一下，单个paxos算法 提出及选定一个议案的过程是
收集accptor收到的议案信息，然后由proposer根据整合的议案息 发送一个带有逻辑时间编号的议案到各个acceptor中，等待接收过半数acceptor的选定结果。 </e>
那有多个议案的时候，我们有什么东西可以重用的呢？ 实际上，proposr,acceptor以及逻辑时间等信息都是可以重用的，因为单个paxos算法对于逻辑时间的要求仅仅是递而已，多个 paoxs实例的值可以一次过由proposer提交给acceptor投票，并有acceptor选择全部通，或者全部失败。

若对于多个PAXOS算法实例有 顺序编号等要求的话，编号可作议案X的值的一部分，一起予以选定。编号的值的选取可参考逻辑时间的设定
<h2 data-source-line="57">五、解决LEANER个数上升，通讯次数指数上升的问题</h2>
其实这个问题很好解决，实际上leaner之间获得最终结果的时间也不是致的。因此 leaner的个数设置2-3个即可，这样就解决了单点故障。如果有更多的进程想要知道程序定的结果的话，那么，由这两个leaner主动推送结果，或者由其他的进程主动过来向leaner查询即可这样，就解决了通讯次数指数级别上升的问题。
<h2 data-source-line="63">六、PAXOS中逻辑时钟的意义</h2>
PAXOS算法中，每个议案都需要附上议案对应的编号（或者说是逻辑钟的时间），这个时间代表的意义是：这个议案的提出已经综合了对应议案时间及之前的所有情况，ACCEPTOR你就看看这个议案是否还没过期适用于当前的情况把~
在原始的PAXOS算法中，每个proposer在给出propose的时候都不清自己掌握的情况 是否最新的，因此给出这个逻辑时间给acceptor来判断是必不可少的。但假设我们照之前的做法，给paxos算法的proposer加入了主备，一 个proposer一直都是主proposer，只能由提出建议的话，那么acceptor就可以默认相信该proposer的议案，而不用管议案的编 号，因为该proposer总掌握了当前最新的情况。
<h2 data-source-line="69">七、引入主PROPOSER周期</h2>
但当主proposer挂掉了，我们需要选举另外一个掌握了所有情况的proposer来担任主 proposer。那要如何获得一个掌握了所有情况的proposer呢？很简单，向过半数的acceptor获取统计运行情况即可。当获取完毕，这个 proposer就掌握所有情况，可以向accptor提出合法建议了
但这个时候，原来的主proposer又复活了怎么办？他之前只是由于网分区等问题失联了而已。现 在又给acceptor发propose了！其实...这种时候，我们可以参考之前的逻时钟的设定，搞一个第几任主proposer的编码。如某 proposer进程第一次当选时，其编号为1，下个进程当选时的那次，编号则为2。这样，当acceptor正处于编号为2的主进程周期时，收到了编 号为的propose时要予以拒绝。这里的主进程编号跟之前的逻辑时钟的编号的意义也是一样的，如进程编码为1的议案表示的是，本议案已结合了 主进程阶段1及之前所有阶段的情况，给出了这个建，acceptor你就看看要不要接受把。
<h2 data-source-line="75">八、主PAXOS周期+多PAXOS算法实例</h2>
以上这个主PROPOSER周期的改进，可以跟多PAXOS实例结合使用。以每个主 proposer周期内依次运行的PAXOS算法实例进行编号，主PROPOSER需要提交某些PROPOE的时候，忽略prepare- promise步骤，直接将 议案值 及 PROPOSER周期 及 PAXOS实例编号发给个ACCEPTOR 并收到过半数回应即可认为议案被选定。
<h2 data-source-line="80">九、ZAB协议与PAXOS</h2>
了解过ZAB协议看过上述PAXOS优化之后，应该能察觉到，上述经过优的协议与ZAB几乎一致。
<ul data-source-line="84">
ZAB没有PROPOSER,ACCEPTOR,LEANER的区别，但实际上，ZAB只是将PROPOSER,ACCEPTR,LEANER合并成同一个进程。
ZAB存在一个LEADER，对应上述PAXOS优化方案的主PROPOSER
ZAB协议分为 消息广播 及 崩溃恢复两部分。

消息广播部分对应省略了PREPARE-PROMISE步骤的PAXOS算法
崩溃恢复对应于主PROPOSER选举（当然，还有些细微差别）

ZXID对应于 主周期及PAXOS算法实例编号组合

<h2 data-source-line="92">十、总结</h2>
嗯，就这样。帮助巩固了PAXOS和ZAB