

Paxos理论介绍(3): Master选举



LynnCui
软件·音乐

已关注

▲ 你关注的 朱一聪 赞同

发布于 2016-07-09 13:45，编辑于 2016-12-06 19:06

前文：[Paxos理论介绍\(2\): Multi-Paxos与Leader](#)

建议没有阅读前面文章的读者可以先花少许时间阅读一下。

Master

开门见山，我们先明确一下Master的定义。Master是一个角色，这个角色的特点是，在我们选定的一些节点集合内，任一时刻，仅有一个节点成为Master或者没有任何节点成为Master。这是一个非常严格的单点定义。

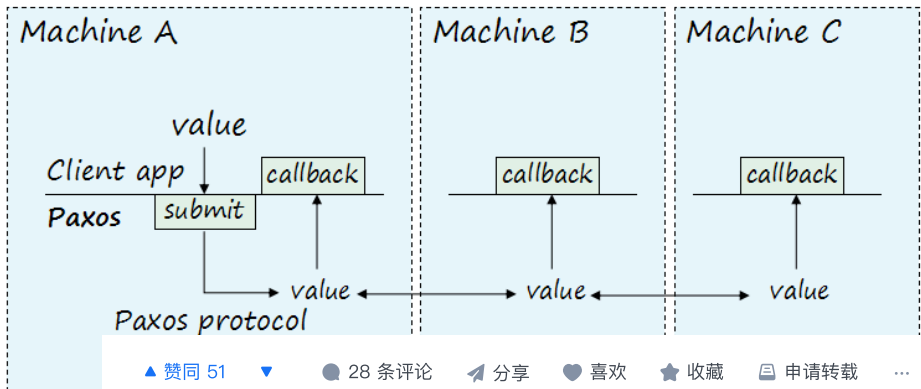
Master的应用非常广泛。比如在分布式存储里面，我们希望读取一个最新的值，那么常见的做法是我们先选举出一个Master，读写都由Master来完成，那么在Master上读取到的就肯定是最新的。另外还比如一些仲裁模块，往往也希望有Master来协助。

Master选举与Paxos的关系

如何选举Master？由于Master具有严格的单点定义，那么必须有一个强一致性的算法才能完成选举，当然我们这里采用了Paxos。但Master选举算法自身也是一个通用性的算法，它可以与任何强一致性算法搭配来完成，而无需要求一定是Paxos。所以这里我们希望设计一个与Paxos完全解耦的工程实现，也就是Master选举只用到Paxos工程实现的API，而无需侵入Paxos算法内部。

Paxos的工程应用

这个涉及到Paxos工程上API设计以及状态机，这里先不展开讲，来看一张图相信大家就懂了，图片来自论文"Paxos Made Live"。

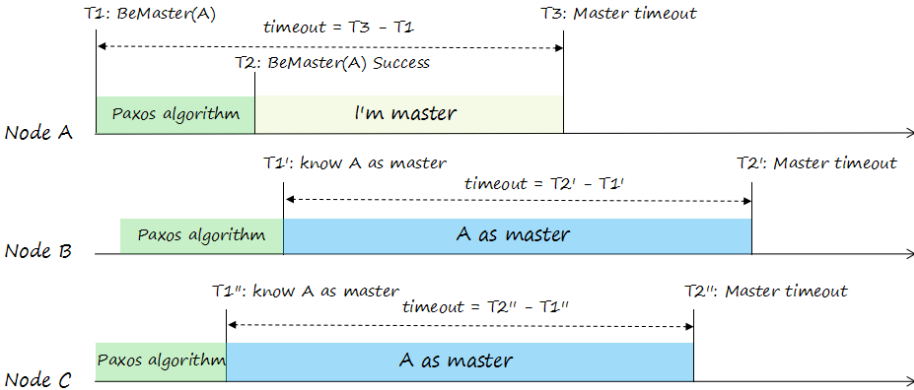


▲ 赞同 51 ▼ ● 28 条评论 ↗ 分享 ♥ 喜欢 ★ 收藏 📄 申请转载 ...



Paxos的应用简明来讲就是由算法确定一个操作系列，通过编写这些操作系列的callback（也就是状态机的状态转移函数），使得节点进行相同顺序的callback，从而保证各个节点的状态一致。

Master选举租约算法



BeMaster是一个操作，这个操作很简单，就是提议自己成为Master，图片里面A节点希望自己成为Master。任何节点都可以发起这个操作尝试将自己提升为Master，除了已经得知别人已被选为Master。当得知别人被选为Master后，必须等待timeout长度时间，才能发起BeMaster操作。而如果是获知自己成为Master，那么从BeMaster开始的timeout时间内可认为自己是Master，如图示，T2-T3的时间窗内，视作Master的任期。

如何将上面所述的租约算法与Paxos结合起来？

- BeMaster可以认为是一个Submit操作，其Value携带的就是自己的节点信息。
- callback做两件事情，第一：发现Value的节点非自己，则等待timeout时间再发起BeMaster。第二：发现Value的节点是自己，那么将自己提升为Master，并在T(BeMaster) + timeout后过期。

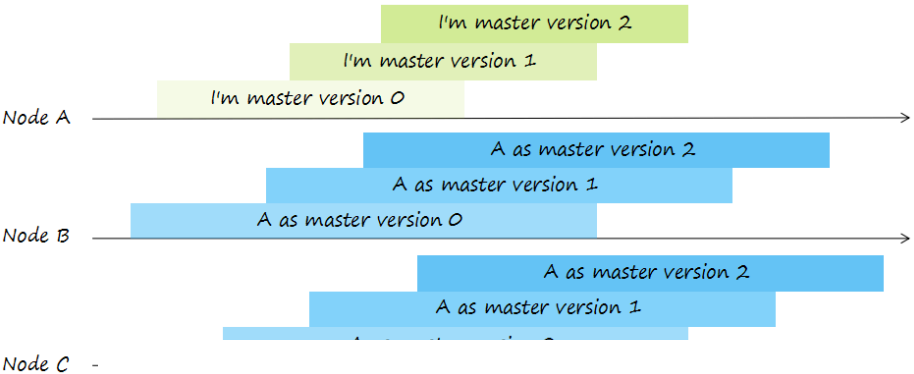
算法正确性如何保证？

- 一致性由Paxos保证，也就是只要Value被Paxos选出来，那么其包含的肯定是同一个节点信息，不会出现选举冲突。
- Master的单点性通过租约算法保证。由于恒定T(BeMaster) < T(Know other as master)，那么Master的过期时间肯定要比非Master节点认为Master过期的时间早，从而保证Master任期内，肯定不会出现其他节点尝试来抢占Master。

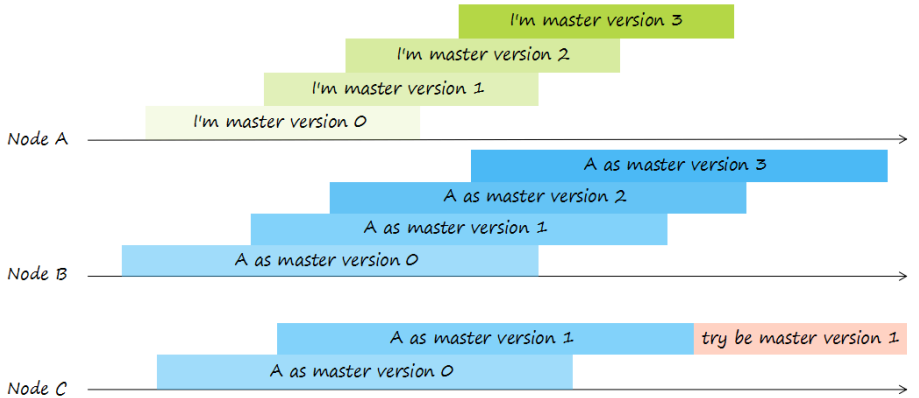
这里给大家提一个问题，图示里面，为何Master任期的起始时间是从BeMaster算起，而不能是从BeMaster success算起？相信如果理解了Paxos算法的读者，应该可以很轻松回答这个问题。

Master续任

只需要在Master任期内成功完成一次BeMaster操作，即可延长Master任期，在正常情况下这样不断迭代下去，一般会使得Master非常的稳定。



上图可以看到在多次的BeMaster选举里面，我们需要给每一个任期赋予一个version，这是为什么？下面通过一个例子来解释这个问题。



这个图示情况是NodeA不断的在续任，但NodeC可能与NodeA无法通信或者其他原因，在获知NodeA第二次续任成功后就再也收不到任何消息了，于是当NodeC认为A的Master任期过期后，即可尝试发起BeMaster操作。这就违背了算法的保证了，出现了NodeA在任期内，但NodeC发起BeMaster操作的情况。

这里问题的本质是，NodeC还未获得最新的Master情况，所以发起了一次错误的BeMaster。version的加入是参考了乐观锁来解决这个问题。发起BeMaster的时候携带上一次的version，如果这个version已经不是最新，那么这一次BeMaster自然会失效，从而解决问题。理解乐观锁的读者应该可以很快脑补出version的作用，这里就不详细展开了。

小贴纸

说的再多不如阅读源码，猛击进入我们的开源Paxos类库实现：[github.com/tencent-wech...](https://github.com/tencent-wechat/paxos)在src/master目录有完整的Master租约算法实现代码。

如果觉得文章对你有帮助或者启发，麻烦点一波关注，谢谢各位。

发布于 2016-07-09 13:45，编辑于 2016-12-06 19:06

「真诚赞赏，手留余香」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

分布式系统 算法 计算机



欢迎参与讨论

28 条评论

默认 最新



朱一聪 我关注的人

同一个版本不同的节点 bemaster 对应一个 paxos实例，不同版本的 bemaster 对应不同 paxos 实例？

2016-07-09

回复 2



LynnCui 作者

可以这么理解

2016-07-09

回复 喜欢




Ga - 01-

出]

2023-04-04

回复 喜欢


 三年不开张

...

你这个应该是对paxos made live 的笔记，我觉得你理解有点偏差
Lease 是保证master 的延续，而不是用来选举，lease 对应的应用就是read，确保自己的确是master ，不需要再走一遍paxos，直接read local

2018-09-05

回复 喜欢


 赵宝宝

...

这里问题的本质是，NodeC还未获得最新的Master情况，所以发起了一次错误的BeMaster。这个问题是不是paxos made live中也讲过的那个？通过master定期加大proposalID来解决的那个呢。感觉phxPaxos通过乐观锁来解决更有效些，也更优雅。

2018-04-14

回复 喜欢


 terrytan

...

如果连续拒绝，直到该follower机与Master失去联系的话，那提交proposal那台机器的proposalld会连续增加，导致最终成功的时候，改proposalID上升的非常快

2017-09-25

回复 喜欢


 王军

...

博主写的很棒很容易懂。所以想再请教一个问题：
《Paxos made live》中Master Lease一节中，提到的老master重新上线后，重新发起选举这个行为，导致Master频繁切换。
其实作者是不是想表达，因为老Master没有遵守Master租约，擅自发起了Master选举。然后作者的解决方法是不是通过一轮新Master的prepare时候的顺便把prepare序号提升，让老Master prepare失败，从而得知自己不是Master了？

2017-01-30

回复 喜欢


 loloa

...

老Master不知道现在Master是谁，但是其他机器知道，那些机器可以拒绝并告诉老Master现在的Master是谁，这个在multi paxos implement那篇论文有说

2019-04-19

回复 喜欢


 zhanjia

...

非常感谢回答，我还看到关于微信PAXOS的另外一篇文章
mp.weixin.qq.com/s/...
里面关于LEARNER，我想问一下这个LEARNER触发的时机是什么？是ACCEPTOR发现自己确认的最大提议号落后于当前最新的议题号吗？这个过程是不是可以异步触发，也就是ACCEPTOR先直接REJECT当前的议题号再启动LEARN的过程. 如果不巧有超过半数的ACCEPTOR都落后了是不是会BLOCK新的PROPOSAL直到超过半数的ACCEPTOR赶上，还是说干脆直接REJECT新的PROPSAL？（虽然发生的概率非常小）

2016-12-02

回复 喜欢


 LynnCui 作者

...

理论上不会出现多数派都处于落后状态的，出现这种情况应该属于拜占庭问题，如某台机器数据被清空了，这个时候必须得BLOCK住，没有其他办法。LEARNER与ACCEPTOR的逻辑在实际的工程实现中都是完全异步的。

2016-12-06

回复 喜欢


 zhanjia

...

感谢大神分享PAXOS的实战经验，看了这个感觉和RAFT的LEADER选举非常相似，我理解这里说的VERSION相当于RAFT 的CURRENT TERM，不知道是否正确？我还是有一个问题，如果一个节点在MASTER任期内被孤立了（和其他节点通信割裂），那其他节点选出一个新的节点会不会出现双MASTER的情况？希望大神指点

2016-12-01

回复 喜欢


 LynnCui 作者

...

这里master选举已经是一致性协议之外的状态机了，所以这里的version跟raft一致性协议里面的东西没有可比性，这个version只是针对这个get and set形式的状态机做一个乐观锁保证原子性。如果硬要跟raft对比的话，用写关于leader那篇文章才合适，可以认为raft里面的item和paxos里面的prepare ballot相似。关于第二个问题，不会出现双master，因为其他节点能选出master的前提必然是租约过期了，而租约过期前提是原先的master没续租成功。

2016-12-01


回复 喜欢

 xiaodan zhuang

...

真棒，有个小拼写错误，小贴纸之前一段：理解乐观锁的读者应该可以很快脑补出version的作用，这里应该是『乐观锁』。

20




LynnCui 作者

谢谢哈，我修改一下。

2016-11-15

回复 喜欢




鸡毛飞上天

c比master发起者a提前知道a是master是为啥呀。a发起决议，a的决议被选择，a知道自己的决议被选择前，b发起决议，b决议内容必须和被选择的a决议内容一样，b知道a是master，是这样吗，，麻烦老师啦。。

2016-11-06

回复 1 喜欢



LynnCui 作者

对的。


2016-11-07

回复 喜欢

点击查看全部评论 >

 欢迎参与讨论

文章被以下专栏收录



分布式一致性与高可用实践

微信后台在分布式一致性与高可用上的实践经验介绍

推荐阅读



Golang 实现 Paxos 分布式共识算法

多颗糖 发表于强力研分布...



一些关于 Distributed System 的话题（二）Paxos

邓明华 发表于Chaos

paxos共识算法

Google Chubby的作者Mike Burrows对Paxos的评价极高：“这个世界上只有一种一致性算法，那就是 Paxos”。其实也不为过，像非常有名的 Raft 算法、Zab 算法等都是基于 Paxos 的简化和改进。 ...

Tang老鸭

分布式技术探索——尝试 paxos

上篇文章中我们介绍了一致性算法。解决了集群中增减机器导致的数据迁移问题，以及数据均匀分布问题。本文将进一步保障整个服务的高可用的一致议paxos。它主要解决的问题我不是老欧