

A	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
B	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
C	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
D	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...

Paxos理论介绍(4): 动态成员变更



LynnCui
软件·音乐

已关注

29 人赞同了该文章

发布于 2016-08-23 23:58， 编辑于 2016-10-26 17:06

多数派的本质

在讲解成员变更之前，我们先回顾一下前文介绍的Paxos理论第一篇文章 Paxos理论介绍(1): 朴素 Paxos算法理论推导与证明，（仔细回顾数学定义和投票约束章节）文中提到Bqrm为一轮成功投票所需要的投票者集合，而Paxos算法理论第二条约束要求任意两个Bqrm的交集不为空，于是乎我们可以理解为Bqrm就是一个多数派的意思，因为在一个固定的投票者集合里面，取多数派作为Bqrm，肯定是满足条件的。

而所有的理论介绍，都是基于投票者集合是固定的。一旦投票者集合出现变化，Bqrm的定义将不再是多数派，Bqrm的取值将变得异常困难，而无法定义Bqrm，Paxos算法的约束就无法达成一致性。也就是说，固定的成员是Paxos算法的根基。

人肉配置进行成员变更？

我们再进行第二篇文章 Paxos理论介绍(2): Multi-Paxos与Leader 的回顾，通过文章我们知道 Paxos是以独立的实例的方式推进，从而产生一个一致的有序系列，而每个实例都是单独运作的 Paxos算法。再根据上文，我们得出一个要求，在相同的实例上，我们要求各个成员所认为的成员集合必须是一致的，也就是在一次完整的Paxos算法里面，成员其实还是固定的。

每个成员如何得知这个成员集合是什么？通常我们是通过配置文件。在通过配置的变更能否满足以上的要求呢？我们知道Multi-Paxos在推进的过程中是允许少数派落后的，而在同一个实例里面，获知Value被chosen也是有先后的，那么配置的变更可能出现在以上任何的先后夹缝内，下图演示一个更换节点C为D的样例。





A	A,B,C					A,B,D	
	1	2	3	4	5	6	...
B	A,B,C			A,B,D			
	1	2	3	4	5	6	...
C	A,B,C					A,B,D	
	1	2	3	4	5	6	...
D				A,B,D			
	1	2	3	4	5	6	...

注：绿色代表已经获知chosen value的实例

可以观察到4这个实例，已经出现了成员混乱，(A,C),(B,D)都可以被认为是Bqrm，但明显这两个Bqrm没有交集，已经违反Paxos协议。

事实上我们追求的是找到一个切入时机，使得Paxos的运作程序都在这相同的时刻完成配置的原子切换，但明显在分布式环境里面能做原子切换的只有一致性算法，所以配置更新不靠谱。

题外话，如果真的要使用人肉配置更新，在工程上是有一些办法，通过一些工具加入人的细微观察来无限逼近这个正确性，但终究只能逼近。在理论层面我们会放大任何现实中可能不会出现的细微错误，比如时间的不同步，网络包在交换机无限停留，操作系统调度导致的代码段卡壳等等，这些都会导致这些人肉方法不能上升到理论层面。况且，我们接下来要介绍的动态成员变更算法也是非常的简单。所以这些细致的问题就不展开来聊了。

Paxos动态成员变更算法

这个算法在 Paxos Made Simple 的最后一段被一句话带过，可能作者认为这个是水到渠成的事情，根本不值一提。

Multi-Paxos决议出的有序序列，一般被用来作为状态机的状态转移输入，一致的状态转移得出一致的状态，这是Paxos的基本应用。那么非常水到渠成的事情就是，成员(投票者集合)本身也是一个状态，我们通过Paxos来决议出成员变更的操作系列，那么各台机器就能获得一致的成员状态。如下图。



A	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
B	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
C	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
D	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...

在4这个实例，我们通过Paxos算法来决议一个成员变更操作，所有的节点在实例4之后都能获取到成员从A,B,C变成了A,B,D，在理论上达到了原子变更的要求。

延缓变更生效

通常Paxos的工程化为了性能和进展性都会由一个Leader节点进行数据写入，而成员变更往往可能会导致Leader节点发生变化。那么变化的瞬间可能会出现大量当前Leader节点堆积请求的失败。

另外如果采用多个实例基于窗口滑动并行提交（这里暂时不展开讲，而且我觉得这个在实际工程实现中必要性也不是很大）的话，在成员变更操作被chosen后，之后的实例可能还在Paxos算法运行当中，那么这些实例的Bqrm可能已经被确定下来，所以我们不能再去改动这些实例的Bqrm。如上图例子，4的时候进行了成员变更，但是由于并行提交的关系，5和6可能都已经在提交当中了，那么他的Bqrm还是被确定下来为A,B,C，这时候我们不能去改这些实例的Bqrm为A,B,D。

为了解决这个问题，我们可以将成员生效时间延缓一下，在这个期间将请求引导到新的写入节点。

A	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
B	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
C	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...
D	A,B,C				A,B,D		
	1	2	3	4 (change)	5	6	...

我们可以定义一个延缓窗口为a，成员变更点为l，则生效点为l + a。这个a根据实际情况进行调节。如上图，成员变更点在实例4，但是生效点可以在实例6之后。我们规定旧的Leader在l + a之前仍然能正常写入数据，而新的写入节点必须从l + a开始写入数据，这样可以完成一个平滑过渡。

这里有个异常情况，如果成员变更后，旧的Leader不写入数据了，实例停留在(l + a)，而新的写



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)，我们完整的实现成员变更算法。

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 2016-08-23 23:58，编辑于 2016-10-26 17:06

「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](#) [Paxos](#) [微信](#)



欢迎参与讨论

10 条评论

默认 最新



OceanDeep

l还没选定的时候，l+a之后的记录可以选定吗？我觉得这个才是关键

2019-06-10

回复 1



yingrt

更正一下，老的Leader如果成功选定了l+a之后的记录，那么新Leader的Promise请求会收到该记录的Response，新的Leader也会在对应位置上选定同样的记录

2020-06-19

回复 喜欢



yingrt

我理解不需要额外对l+a之后的记录做限制

a的实现应该是这样：新的Leader对【实例l，以及所有l+a之后的实例】发起Promise请求，成功后新的Leader才可以对l+a之后的实例进行选定。

而老的Leader有可能已经正在对l+a之后的记录进行选定，如果选定成功，那新Leader的Promise请求不会成功，如果新Leader Promise成功，那么老的Leader就无法对l+a之后的记录进行选定。

2020-06-19

回复 喜欢



知乎用户zKZxv8

你好 如果在 (l, l + a) 期间 老leader A 挂了 这时候假如在AC上有达成的多数派决议，到配置生效ABD集群后不符合多数派 数据是否会丢失？

2019-03-13

回复 喜欢



yingrt

新的集群在l+a之后才生效，所以 (l, l+a) 之间的paxos实例认可的集群仍然是ABC，如果A和C同时挂掉，数据是有可能丢失的。
工程上这种情况应该让算法卡住，等A或者C恢复，在这之前不响应请求

2020-06-19

回复 喜欢



Feiyyr

你好~文章中提到“多实例基于窗口滑动并行提交”没有展开说，不是很能理解，请问楼主后面有打算写文章详细介绍一下吗？或者可否推荐一下相关论文或文章呢？

2016-12-17

回复 1



姜涛

如果不用窗口滑动，如果配置只生效，极端情况下有可能全局集群分裂





satanson

我关注的人

粗略的看了一下，是原文党，不是简单的博文转发，网上好多关于paxos和zab都是错误的，包括wiki也是只字片语

2016-08-28

回复

2



starrynight

你好，你们的库里面有一个角色是committer，我看到它最后还是借助proposer去进行propose的，你能解释一下这个committer是怎样一个角色吗？难道它就是leader吗？那产生这个committer的逻辑是在那一段啊？

2016-08-24

回复

1



LynnCui

作者


committer只是对propose操作的一个封装，无实际角色意义。phxpaxos对数据的写入持开放态度，不会做任何拒绝。单点写入我们推荐使用者搭配master功能进行更灵活的数据写入调度。如我们实际使用中发现有些场景必须需要多点同时写入的。

2016-08-24

回复

1

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



ZEMAX | 如何使用ZOS-API创建自定义操作数

达摩寺扫地僧



搭建 Doxygen 便捷编写环境 (VSCode)

端水大法师 发表于搭建优雅编...



vscode开发ROS1(22)-给ros添加界面(python)

穆士凝魂 发表于ROS



Palabos第一章：linux系统与终端基本操作

Mr.Lu