专题21：Raft 协议（史上最全、定期更新）

本文版本说明：V2

> 此文的格式，由markdown 通过程序转成而来，由于很多表格，没有来的及调整，出现一个格式问题，尼恩在此给大家道歉啦。
>
> 由于社群很多小伙伴，在面试，不断的交流最新的面试难题，所以，《Java面试红宝书》， 后面会不断升级，迭代。
>
> 本专题，作为 《Java面试红宝书》专题之一， 《Java面试红宝书》一共**30个面试专题，后续还会增加**

《Java面试红宝书》升级的规划为：

后续基本上，**每一个月，都会发布一次**，最新版本，可以扫描扫架构师尼恩微信， 发送 “领取电子书” 获取。

尼恩的微信二维码在哪里呢？ 请参见文末

面试问题交流说明：

如果遇到面试难题，或者职业发展问题， 或者中年危机问题，都可以来 疯狂创客圈社群交流，

加入交流群，加尼恩微信即可，

入交流群，加尼恩微信即可， 发送**“入群”**

raft算法之所以容易理解，其一是他将一致性问题划分成几个子问题，这几个子问题都是独立、可理解和解释的。从传统的思维来讲，对于一个复杂的系统或者工程，都是大化小，分解实现，然后去尝试融合解决整体逻辑。

1、Raft 详解

`Raft 算法`是分布式系统开发首选的`共识算法`。比如现在流行 Etcd、Consul、Nacos。

如果`掌握`了这个算法，就可以较容易地处理绝大部分场景的`容错`和`一致性`需求。比如分布式配置系统、分布式 NoSQL 存储等等，轻松突破系统的单机限制。

Raft 算法是通过一切以领导者为准的方式，实现一系列值的共识和各节点日志的一致。

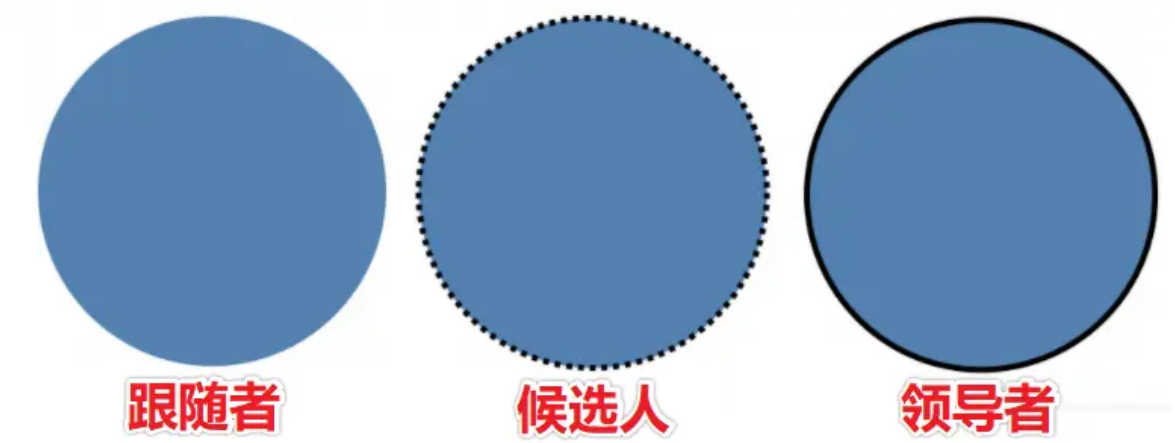
1.1 Raft 角色

跟随者（Follower）：`普通群众`，默默接收和来自领导者的消息，当领导者心跳信息超时的时候，就主动站出来，推荐自己当候选人。

候选人（Candidate）：`候选人`将向其他节点请求投票 RPC 消息，通知其他节点来投票，如果赢得了大多数投票选票，就晋升当领导者。

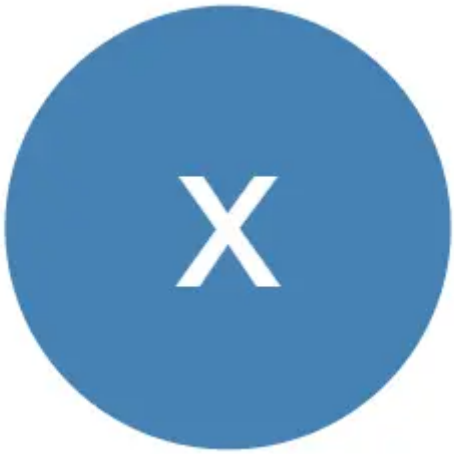
领导者（Leader）：`霸道总裁`，一切以我为准。处理写请求、管理日志复制和不断地发送心跳信息，通知其他节点"我是领导者，我还活着，你们不要"发起新的选举，不用找新领导来替代我。

如下图所示，分别用三种图代表跟随者、候选人和领导者。

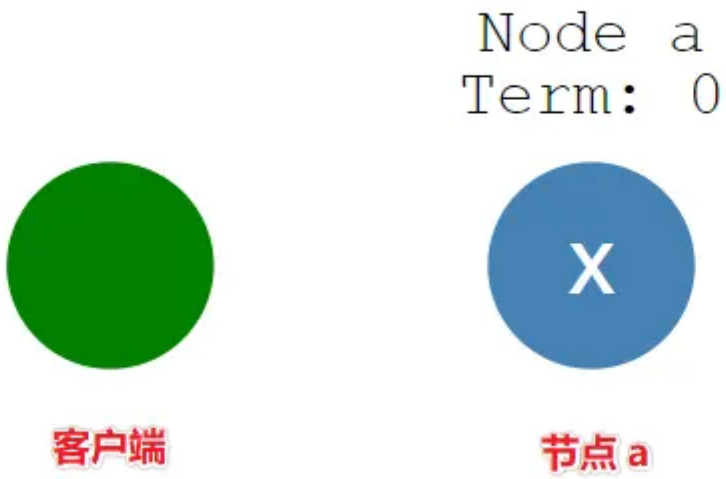


1.1 数据库服务器

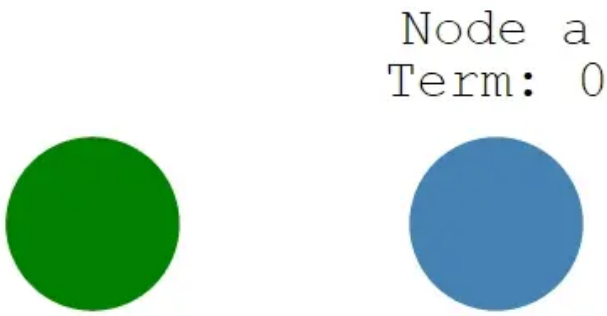
现在我们想象一下，有一个单节点系统，这个节点作为数据库服务器，且存储了一个值为 X。



左边绿色的实心圈就是客户端，右边的蓝色实心圈就是节点 a（Node a）。Term 代表任期，后面会讲到。



客户端向单节点服务器发送了一条更新操作，设置数据库中存的值为 8。单机环境下（单个服务器节点），客户端从服务器拿到的值也是 8。一致性非常容易保证。



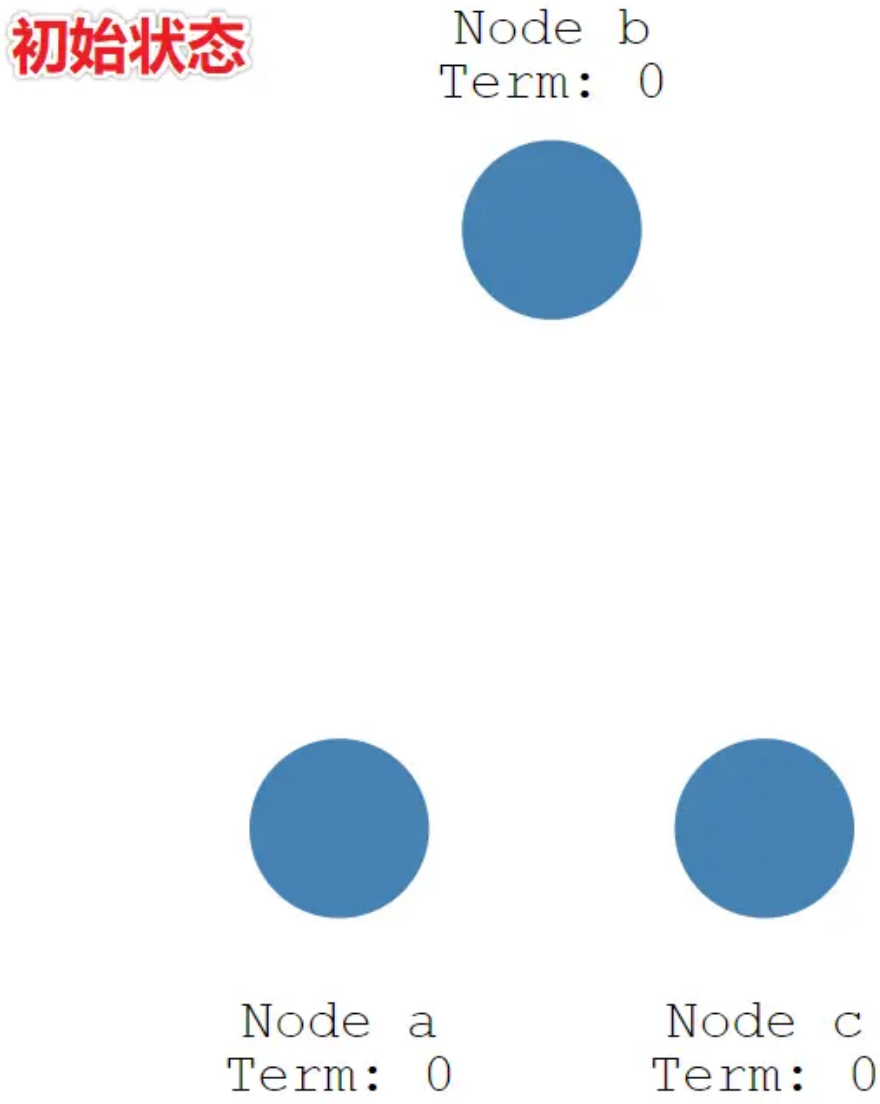
但如果有多个服务器节点，怎么保证一致性呢？比如有三个节点：a，b，c。如下图所示。这三个节点组成一个数据库集群。客户端对这三个节点进行更新操作，如何保证三个节点中存的值一致？这个就是分布式一致性问题。Raft 算法就是来解决这个问题的。当然还有其他协议也可以保证，本篇只针对 Raft 算法。

在多节点集群中，在节点故障、分区错误等异常情况下，Raft 算法如何保证在同一个时间，集群中只有一个领导者呢？下面就开始讲解 Raft 算法选举领导者的过程。

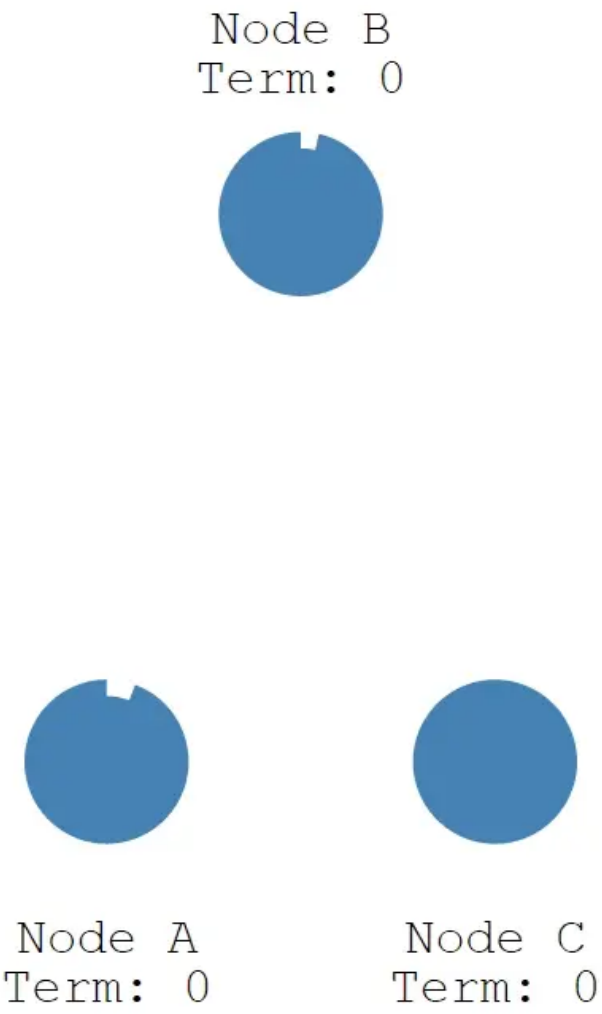
1.2 初始状态

初始状态下，集群中所有节点都是跟随者的状态。

如下图所示，有三个节点(Node) a、b、c，任期（Term）都为 0。



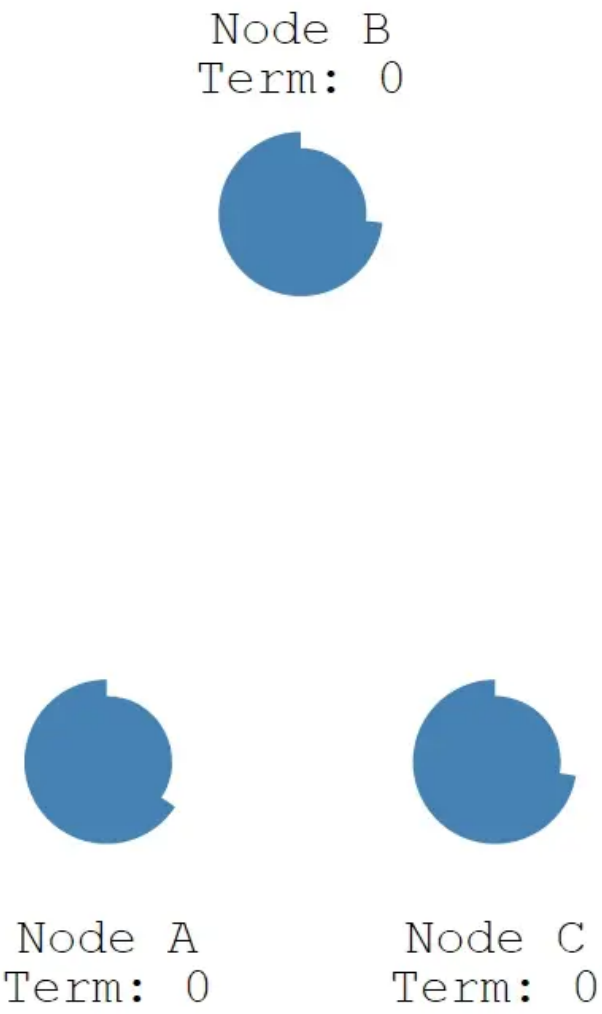
Raft 算法实现了随机超时时间的特性，每个节点等待领导者节点心跳信息的超时时间间隔是随机的。比如 A 节点等待超时的时间间隔 150 ms，B 节点 200 ms，C 节点 300 ms。那么 a 先超时，最先因为没有等到领导者的心跳信息，发生超时。如下图所示，三个节点的超时计时器开始运行。



1.3 发起投票

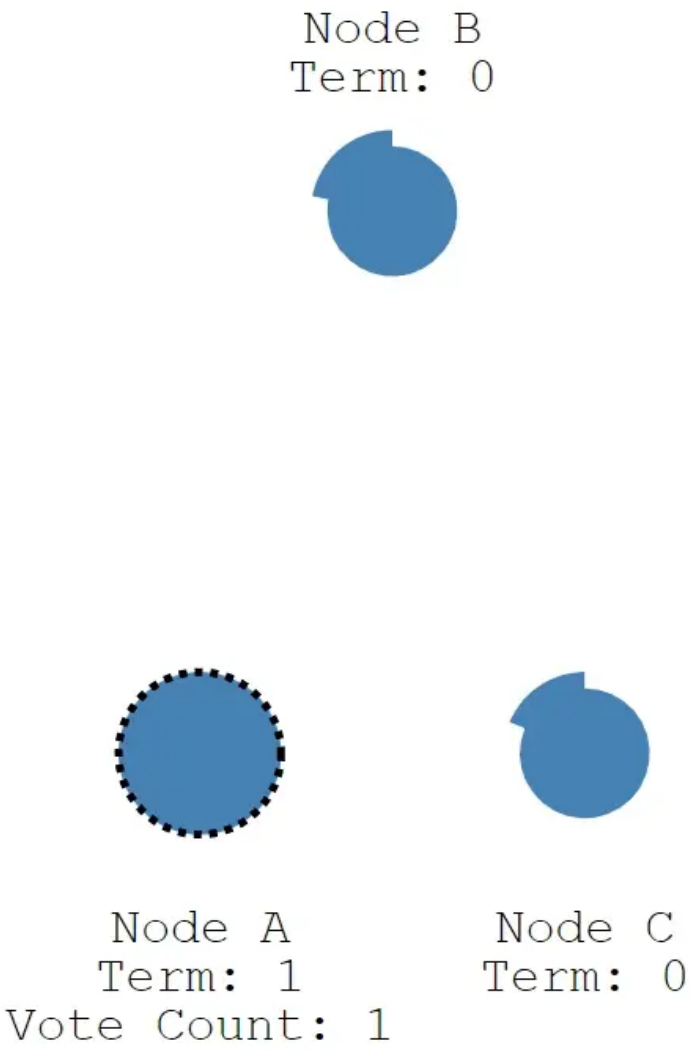
当 A 节点的超时时间到了后，A 节点成为候选者，并增加自己的任期编号，Term 值从 0 更新为 1，并给自己投了一票。

- Node A：Term = 1, Vote Count = 1。
- Node B：Term = 0。
- Node C：Term = 0。



1.4 成为领导者的简化过程

我们来看下候选者如何成为领导者的。



- **第一步**：节点 A 成为候选者后，向其他节点发送请求投票 RPC 信息，请它们选举自己为领导者。
- **第二步**：节点 B 和 节点 C 接收到节点 A 发送的请求投票信息后，在编号为 1 的这届任期内，还没有进行过投票，就把选票投给节点 A，并增加自己的任期编号。
- **第三步**：节点 A 收到 3 次投票，得到了大多数节点的投票，从候选者成为本届任期内的新的领导者。
- **第四步**：节点 A 作为领导者，固定的时间间隔给 节点 B 和节点 C 发送心跳信息，告诉节点 B 和 C，我是领导者，组织其他跟随者发起新的选举。
- **第五步**：节点 B 和节点 C 发送响应信息给节点 A，告诉节点 A 我是正常的。

1.5 领导者的任期

英文单词是 term，领导者是有任期的。

- **自动增加**：跟随者在等待领导者心跳信息超时后，推荐自己为候选人，会增加自己的任期号，如上图所示，节点 A 任期为 0，推举自己为候选人时，任期编号增加为 1。
- **更新为较大值**：当节点发现自己的任期编号比其他节点小时，会更新到较大的编号值。比如节点 A 的任期为 1，请求投票，投票消息中包含了节点 A 的任期编号，且编号为 1，节点 B 收到消息后，会将自己的任期编号更新为 1。
- **恢复为跟随者**：如果一个候选人或者领导者，发现自己的任期编号比其他节点小，那么它会立即恢复成跟随者状态。这种场景出现在分区错误恢复后，任期为 3 的领导者受到任期编号为 4 的心跳消息，那么前者将立即恢复成跟随者状态。

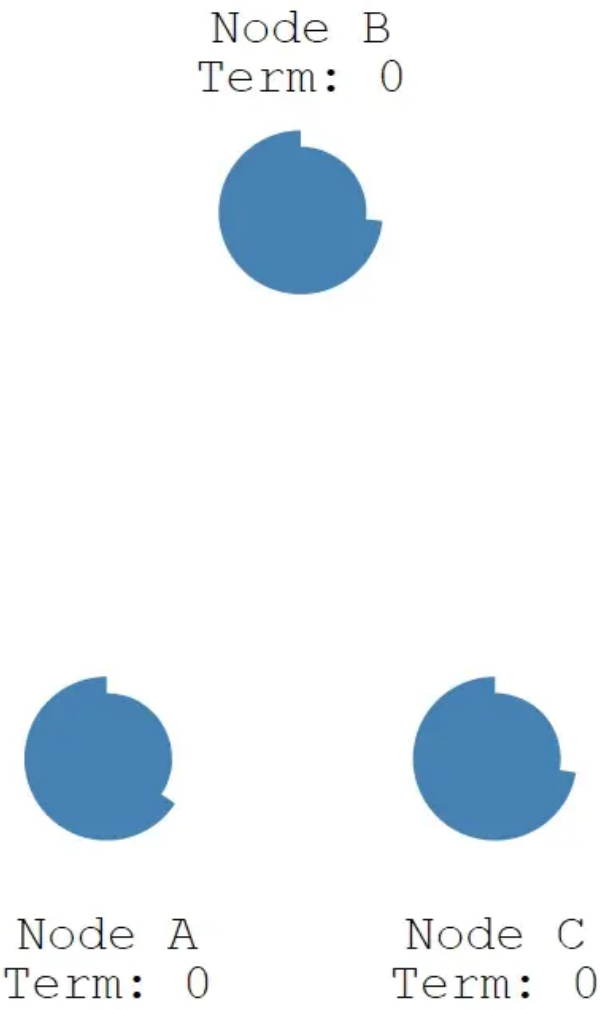
- **拒绝消息**：如果一个节点接收到较小的任期编号值的请求，那么它会直接拒绝这个请求，比如任期编号为 6 的节点 A，收到任期编号为 5 的节点 B 的请求投票 RPC 消息，那么节点 A 会拒绝这个消息。

- 一个任期内，领导者一直都会领导者，直到自身出现问题（如宕机），或者网络问题（延迟），其他节点发起一轮新的选举。
- 在一次选举中，每一个服务器节点最多会对一个任期编号投出一张选票，投完了就没了。

假设一个集群由 N 个节点组成，那么大多数就是至少 N/2+1。例如： 3 个节点的集群，大多数就是 2。

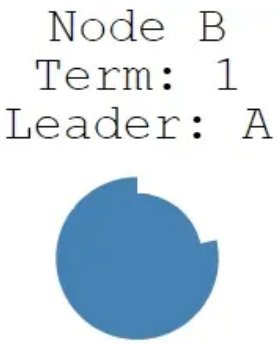
1.6 防止多个节点同时发起投票

为了防止多个节点同时发起投票，会给每个节点分配一个随机的选举超时时间。这个时间内，节点不能成为候选者，只能等到超时。比如上述例子，节点 A 先超时，先成为了候选者。这种巧妙的设计，在大多数情况下只有一个服务器节点先发起选举，而不是同时发起选举，减少了因选票瓜分导致选举失败的情况。

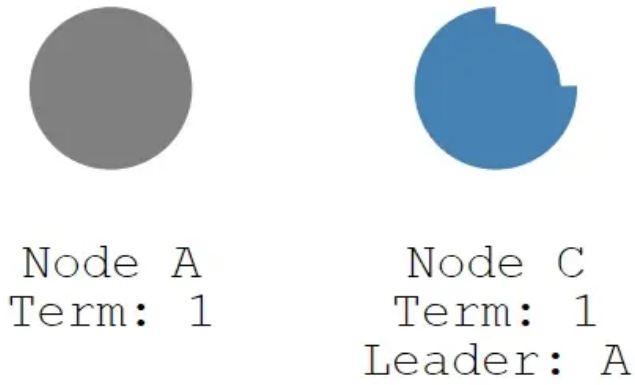


1.7 触发新的一轮选举

如果领导者节点出现故障，则会触发新的一轮选举。如下图所示，领导者节点 B 发生故障，节点 A 和节点 B 就会重新选举 Leader。



领导者节点 A 故障



- 第一步： 节点 A 发生故障，节点 B 和节点 C 没有收到领导者节点 A 的心跳信息，等待超时。
- 第二步：节点 C 先发生超时，节点 C 成为候选人。
- 第三步：节点 C 向节点 A 和 节点 B 发起请求投票信息。
- 第四步：节点 C 响应投票，将票投给了 C，而节点 A 因为发生故障了，无法响应 C 的投票请求。
- 第五步：节点 C 收到两票（大多数票数），成为领导者。
- 第六步：节点 C 向节点 A 和 B 发送心跳信息，节点 B 响应心跳信息，节点 A 不响应心跳信息。

1.8 Raft 算法的几个关键机制

Raft 算法通过以下几个关键机制，保证了一个任期只有一位领导，极大减少了选举失败的情况。

- 任期
- 领导者心跳信息
- 随机选举超时时间
- 先来先服务的投票原则
- 大多数选票原则

2. Raft算法的典型应用

Raft算法的典型应用包括：

- 领导选举
- 日志复制

对于一个集群只有一个leader（领导），那么我们就很容易理解。只要领导操作同步到对应的followers（跟随者），数据必然一致。当leader宕机，需要进行领导选举。

> 日志复制其实就是同步操作数据的过程。leader将操作日志同步到其他节点。

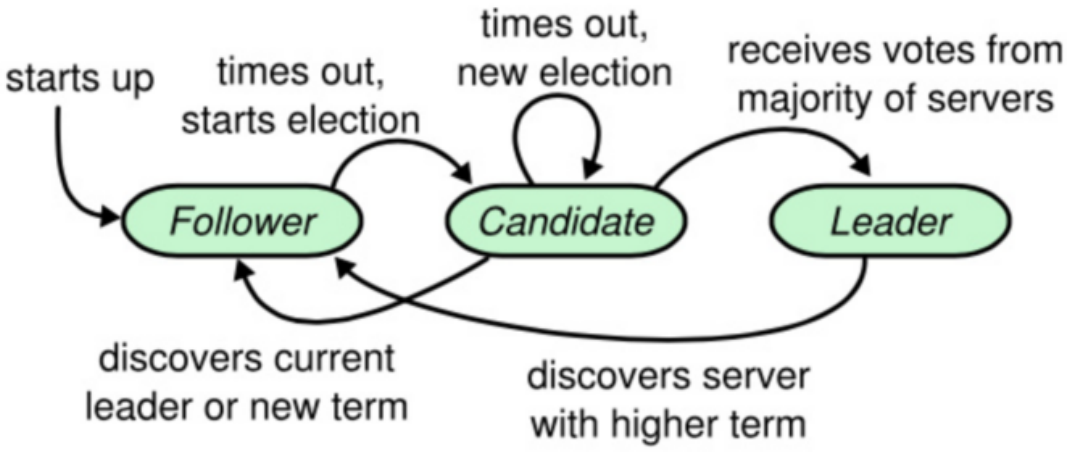
安全性：如何安全的同步，在不同的情况，我们都能保证一致性，这也就是安全性需要考虑的问题。

> 其实就是如此，raft首先假设了领导选举。然后实现了日志复制，最后在安全问题上解决上面的漏洞问题。

1.领导选举

目的：当集群初始化或者领导gg的时候选出一个新的领导。毕竟一个集群不能没有领导，如果没有，那么这个集群就不可用了。

触发机制：通过心跳。



这幅图展现了跟随者、候选者和领导者之间的状态转换关系，下面主要介绍他们的转换流程。

跟随者

如果他能持续从领导者或者候选者接收到有效的RPCs，那么他的状态就不会变。一直保持跟随者身份。但如果没有持续接受，也就是在一段时间没收到有效的RPCs，那就**选举超时**，他会变成候选者。

候选者

要变为候选者，也就意味着他要开启一轮新的选举。那么跟随者会增加自己的任期号，转为候选者。

成为候选者后，自己会并行发送一个为自己投票的RPCs请求 给其他服务器。

成为候选者的整个过程也会保持当前状态，知道满足下面三个条件

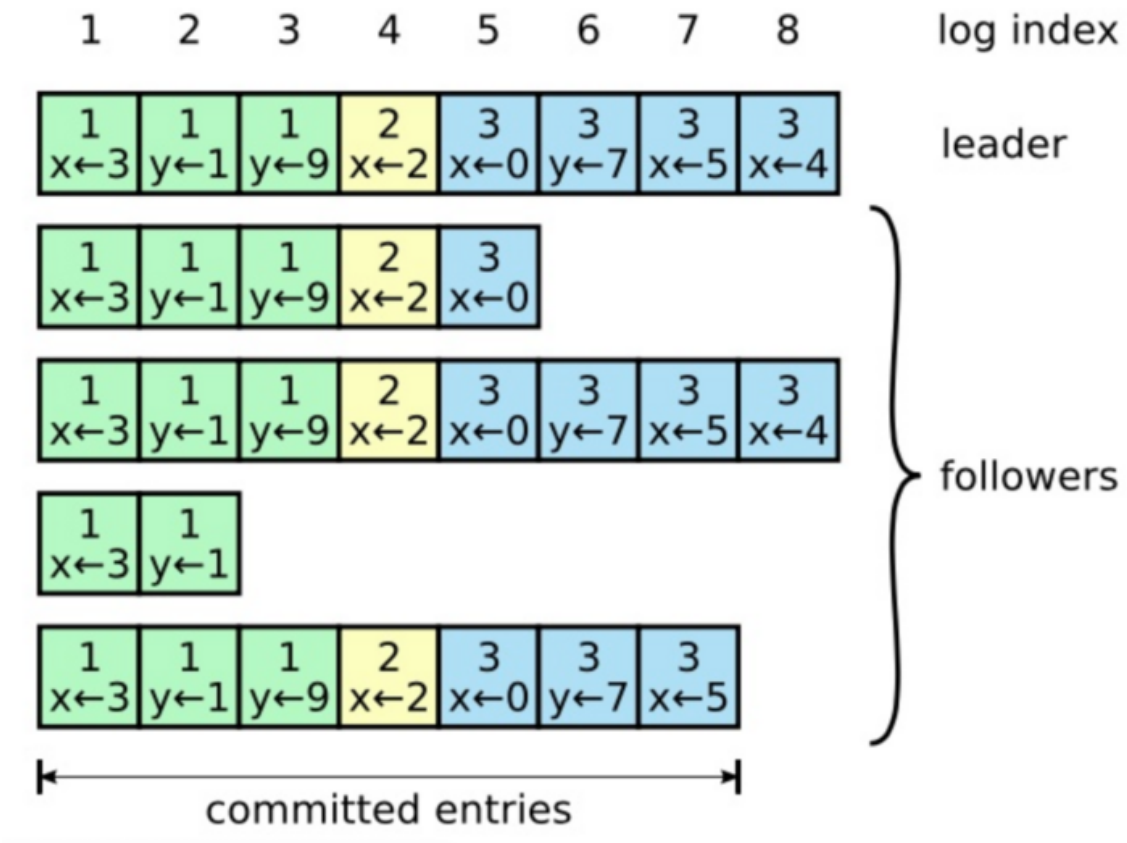
- 他赢得选举，转变为领导制
- 其他节点赢了，他转为跟随者
- 一段时间没有任何人获胜。说明选票被瓜分，重复执行。

这里需要注意的点：

1.对于同一任期号，每个节点一会投一张票。比如服务器A作为候选者生成了编号为5的任期号，那么如果接收到其他节点的编号为5的任期号的投票请求，他会忽略。这个过程遵循的是先来先投的原则。
2.候选者接收到领导者的声明。会判声明中RPC的任期号，如果比自己的还小，那么他还会保持候选者身份，否则转为跟随者。
3.对于上面第三点，重复执行，Raft采用随机超时选举时间进行了优化。降低选票瓜分的可能性。

2.Raft日志复制

直接去理解日志复制，是很容易的，客户端的一条指令到达，领导者会为这条指令创建一条日志条目，并且并行发送到其他跟随者。当日志被安全复制（所谓安全复制后面会有），领导会将日志应用到状态机（比如如果是mysql的insert，那么就是执行insert操作），然后响应客户端。

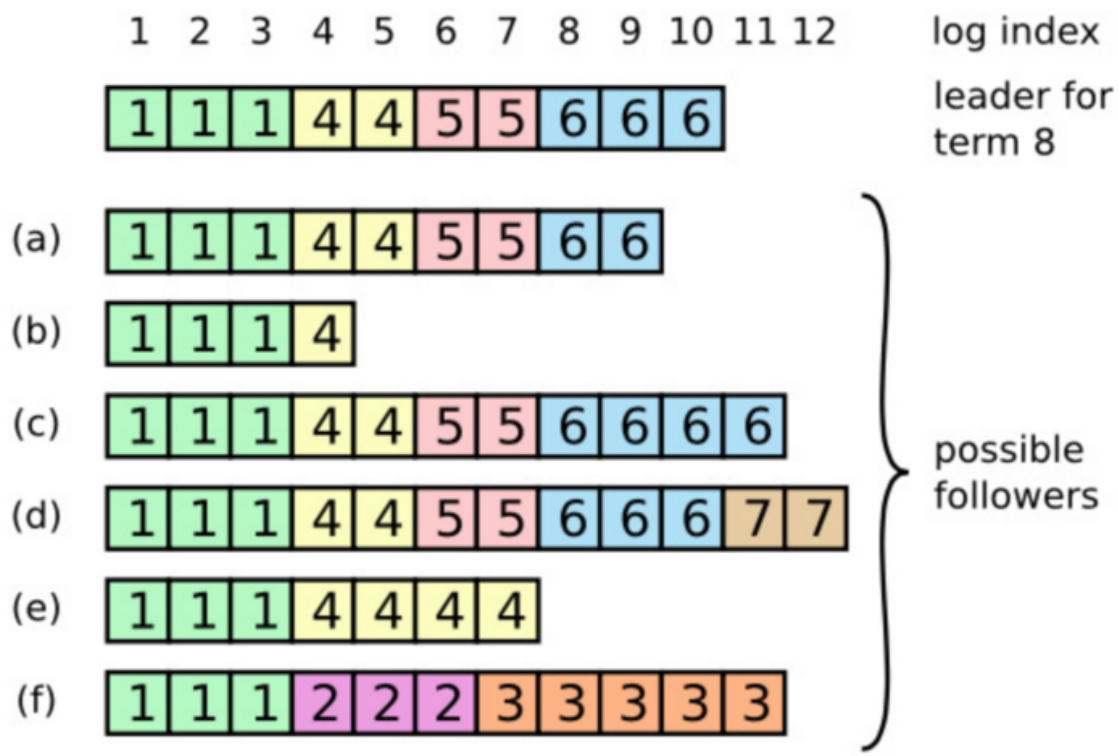


如上图，每条日志都会有对应的任期号，和指令。
每个日志都会有对应的索引。

raft日志匹配特性
1.如果在不同的日志中的两个条目拥有相同的索引和任期号，那么他们存储了相同的指令。
2.如果在不同的日志中的两个条目拥有相同的索引和任期号，那么他们之前的所有日志条目也全部相同。

第一点：一个任期只有一个领导人，并且领导人在一个任期中对于同一索引日志，只会创建一条日志，是不会改变的，是确定的。这就保证第一点成立。
第二点：要想全部相同，就要保证跟随者得到的日志是领导者发送的顺序附加上去的。领导者在发送新的日志时，会附加这条日志之前日志的索引和任期号。如果跟随者发现数据匹配，才会附加上去，否则拒绝。就是一个个状态保证了日志的匹配特性。

> 对于日志不一致的现象，raft是通过跟随者强制复制领导者的日志来保证的。



如上图，对于a-f，最终都会和leader同步，也就是说，d会丢弃日志。f的对应日志也会被丢弃和覆盖。

> 其实就是通过日志覆盖解决。但是对于日志覆盖，我们就会想到一个问题，会不会覆盖已经提交的日志（日志对应指令已经返回给客户端）。那当然不会，如果真有这样，就会有不一致，或者指令丢失现象。

那么如何去做覆盖跟随者日志

其实就是跟随者在append日志的时候，会进行错误校验。

在候选者成为领导者的时候，会为每个跟随者初始化一个nextIndex数组，数组的值初始化为自己最后条日志+1，其实就是理想化状态，默认认为日志都已经同步成功。但是理想总会因为各种原因导致不正确，就用上面那张图。leader会初始化所有nextIndex为11，但是在同步日志的过程中。f节点会出现校验错误的响应。因为f节点10索引对应的日志和leader10索引对应的日志不相同（主要是根据任期号判断）

这里再强调一下为什么根据任期号就可以判断日志是否一致，就是上面所说的日**志匹配原则。**

3.Raft算法的安全性

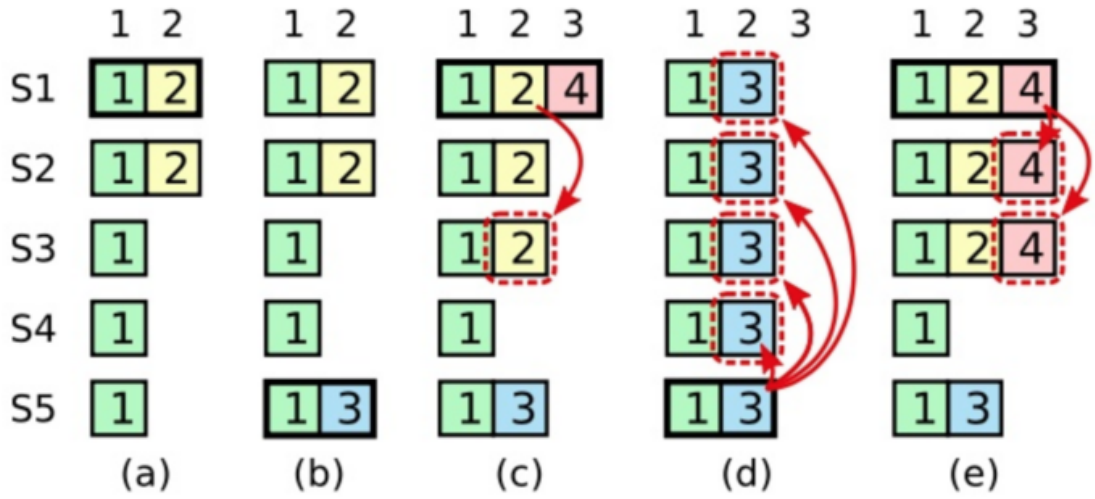
我们明白了如何选举和日志复制，但是没有考虑安全性问题。其实我上慢提到，比如一个宕机很久的跟随着会被选为领导者，进行日志覆盖操作会有丢失问题。

> 其实解决这个办法很简单，就是在领导选举的时候，只能让安全的节点当leader，所谓安全，就是对应节点拥有当前领导者已经提交的所有日志。Raft就是这么做的。

Raft中节点在投票的时候，会判断被投票的候选者对应的日志是否至少和自己一样新。如果不是，则不会给该候选者投票。

日志比较的方法：
1.最后一条日志的任期号。如果大说明新。如果小，说明不新。如果相等。跳到2
2.判断索引长度。大的更新。

还有一个问题，就是领导人不能保证一个已经在大多数节点存在的日志是否已经提交。



a、b、c、d、e代表不同的任期阶段

（a）S1是leader。同步任期2的数据给S2

（b）S1宕机，S5当选（S3、S4、S5投票），产生任期3的日志

（c）S5宕机，S1恢复当选（同步任期2的数据给S3）。

（d）S1宕机，S5当选（因为他的任期日志比其他的都新），复制了任期3的所有数据。

假如说在c阶段，S1提交了任期2的数据，那么如果出现d，则会导致任期2数据被覆盖，丢失。也就是说，S1在任期4时候，不能保证已经在大多数节点存在的日志（任期2的日志）是否提交。

> 所以raft永远不会通过计算副本数目的方式（大多数存在）去提交一个之前任期内（任期2）的日志条目。只有领导人当前任期里的日志条目通过计算副本数目可以被提交（e阶段）。这样之前任期的数据也会被提交。

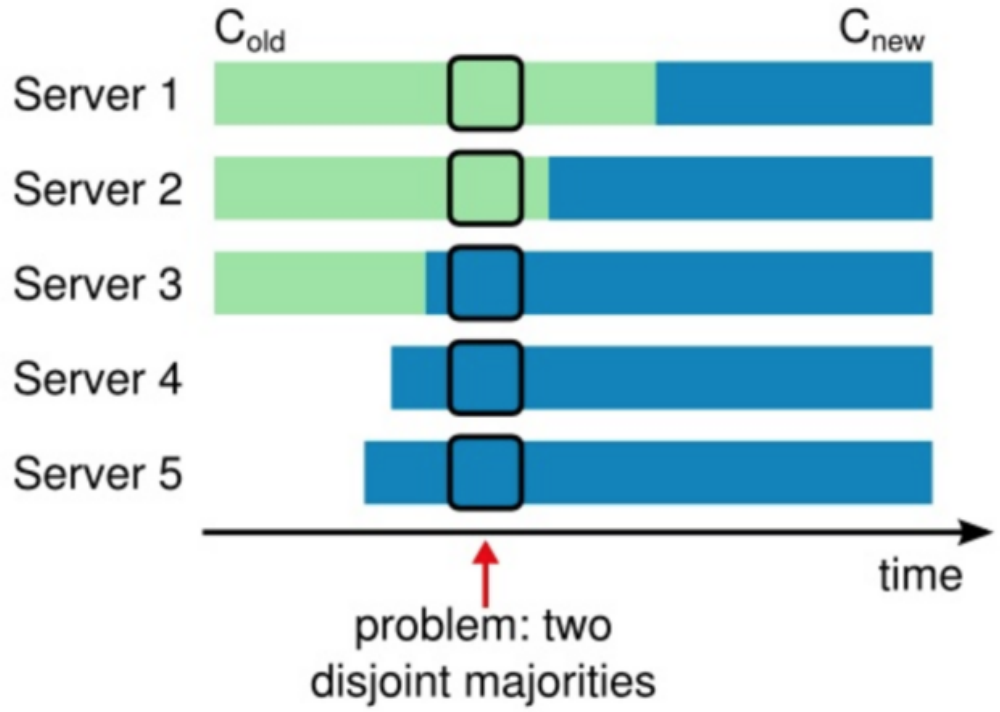
那这里我理解一点就是，加入S1当选为leader，如图c状态，那么，如果不再有新的日志出现，任期2对应的日志就不会提交。那么会导致客户端对应的任期2请求失败。

跟随者和候选人宕机

这个就比较容易理解了，宕机的话RPCs就会失败，Raft通过无限重试却解决这个问题。
所以对于每个RPCs，做到幂等和无限重试，在节点恢复后，就还是会保证一致性状态。

Raft集群成员变化

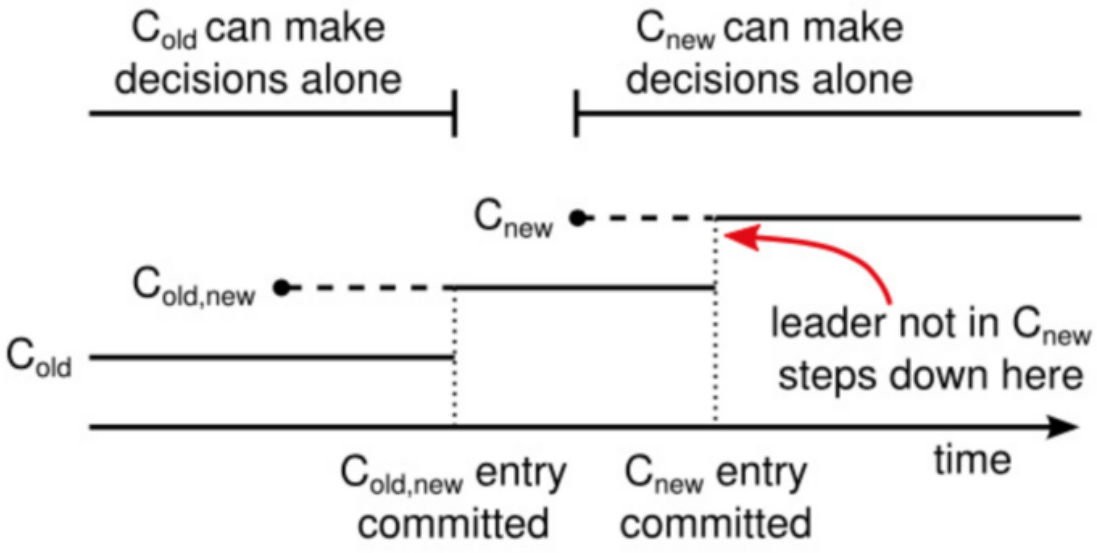
对于集群成员配置变化，如果直接更新每台机器配置，那么就会有安全性问题。以为对于同一时刻，不同节点使用的不同的配置去执行算法逻辑，这就是不安全的。



> 如图，蓝色代表新的配置。绿色代表老的配置。Old状态有三台机器Server1、2、3。 New加入两台server4、5。

那么随着配置时间应用的不同，可能会导致选举出两个leader。
比如server1、2使用老配置，那么1和2都有可能被当选为leader。3、4、5使用心得配置，他们之中的一个也会被当选为leader。

其实这个问题原因就是节点使用了不同配置执行算法逻辑。为了解决这个问题，raft采用两阶段方法（其实只需要保证不会让新或者旧配置单独作出决定就行）



raft吧配置当作普通日志形式去提交。

为了实现两阶段，引入了C(old、new)配置。
还有一点就是，一点一个新的配置日志增加到对应的节点日志中，那么该节点就会立刻使用这条新的日志配置。

> 对于C（old、new）配置，其实就是只有同时满足old和new配置的时候才会生效。

这样理想状态下，如果拥有C（old、new）配置的节点当选为leader。并且提交了该配置，那么说明C（old、new）配置已经在大多数节点应用。下次选举的产生的leader日志中必然会有该配置。这个时候在创建一条新的C（new）配置提交，即可。

硬核推荐：尼恩Java硬核架构班

又名疯狂创客圈社群 VIP

详情：

https://www.cnblogs.com/crazymakercircle/p/9904544.html

架构班（社群 VIP）的起源：

最初的视频，主要是给读者加餐。很多的读者，需要一些高质量的实操、理论视频，所以，我就围绕书，和底层，做了几个实操、理论视频，然后效果还不错，后面就做成迭代模式了。

架构班（社群 VIP）的功能：

提供高质量实操项目整刀真枪的架构指导、快速提升大家的:

- 开发水平
- 设计水平
- 架构水平

弥补业务中 CRUD 开发短板，帮助大家尽早脱离具备 3 高能力，掌握：

- 高性能
- 高并发
- 高可用

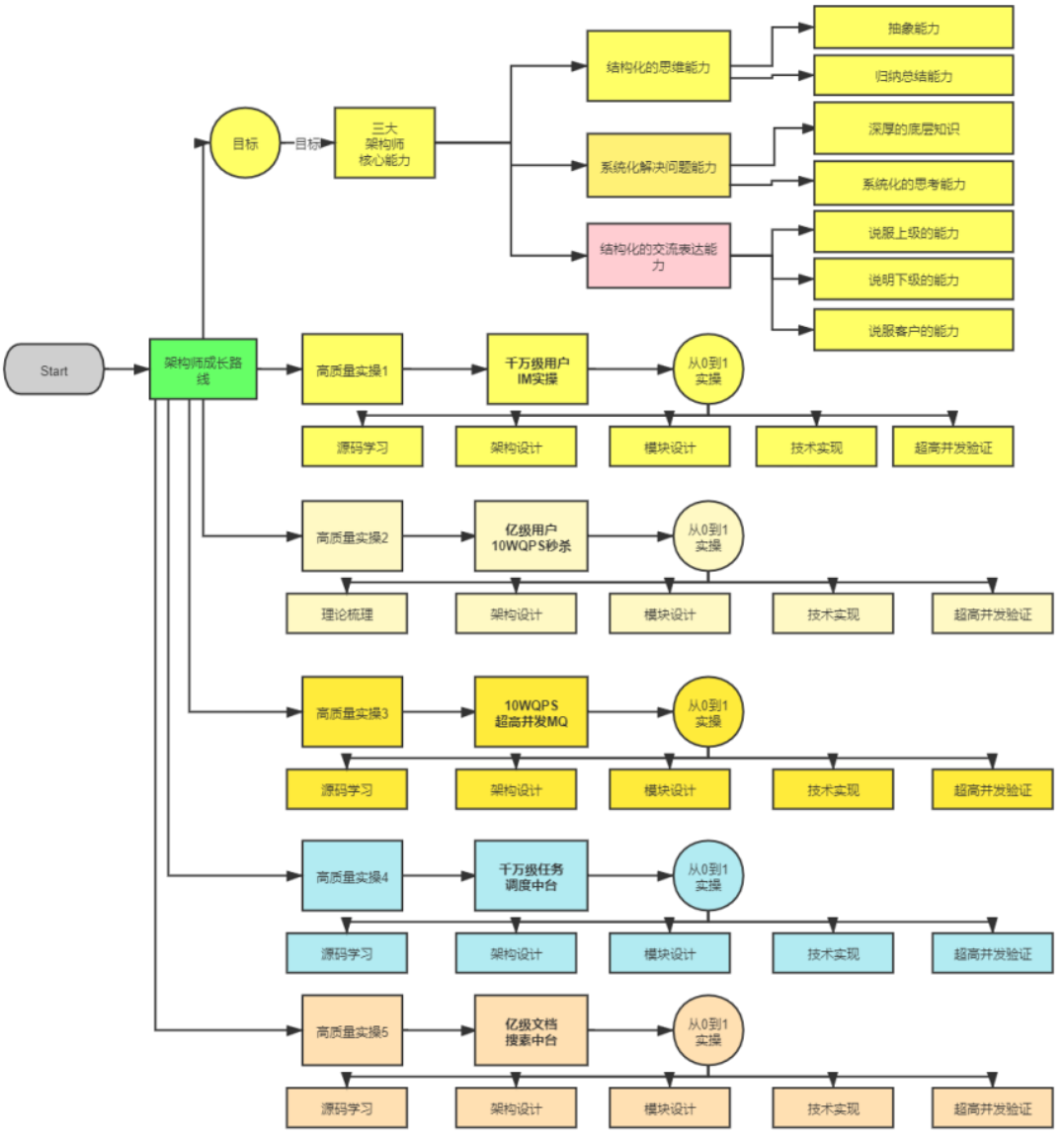
作为一个高质量的架构师成长、人脉社群，把所有的卷王聚焦起来，一起卷：

- 卷高并发实操
- 卷底层原理
- 卷架构理论、架构哲学
- 最终成为顶级架构师，实现人生理想，走向人生巅峰

架构班（社群 VIP）的目的：

- 高质量的实操，大大提升简历的含金量，吸引力，增强面试的召唤率
- 为大家提供九阳真经、葵花宝典，快速提升水平
- 进大厂、拿高薪
- 一路陪伴，提供助学视频和指导，辅导大家成为架构师
- 自学为主，和其他卷王一起，卷高并发实操，卷底层原理、卷大厂面试题，争取狠卷 3 月成高手，狠卷 3 年成为顶级架构师

N 个超高并发实操项目：简历压轴、个顶个精彩



【样章】第 17 章:横扫全网 Rocketmq 视频第 2 部曲: 工业级 rocketmq 高可用(HA)底层原理和实操

工业级 rocketmq 高可用底层原理，包含：消息消费、同步消息、异步消息、单向消息等不同消息的底层原理和源码实现；消息队列非常底层的主从复制、高可用、同步刷盘、异步刷盘等底层原理。

工业级 rocketmq 高可用底层原理和搭建实操，包含：高可用集群的搭建。

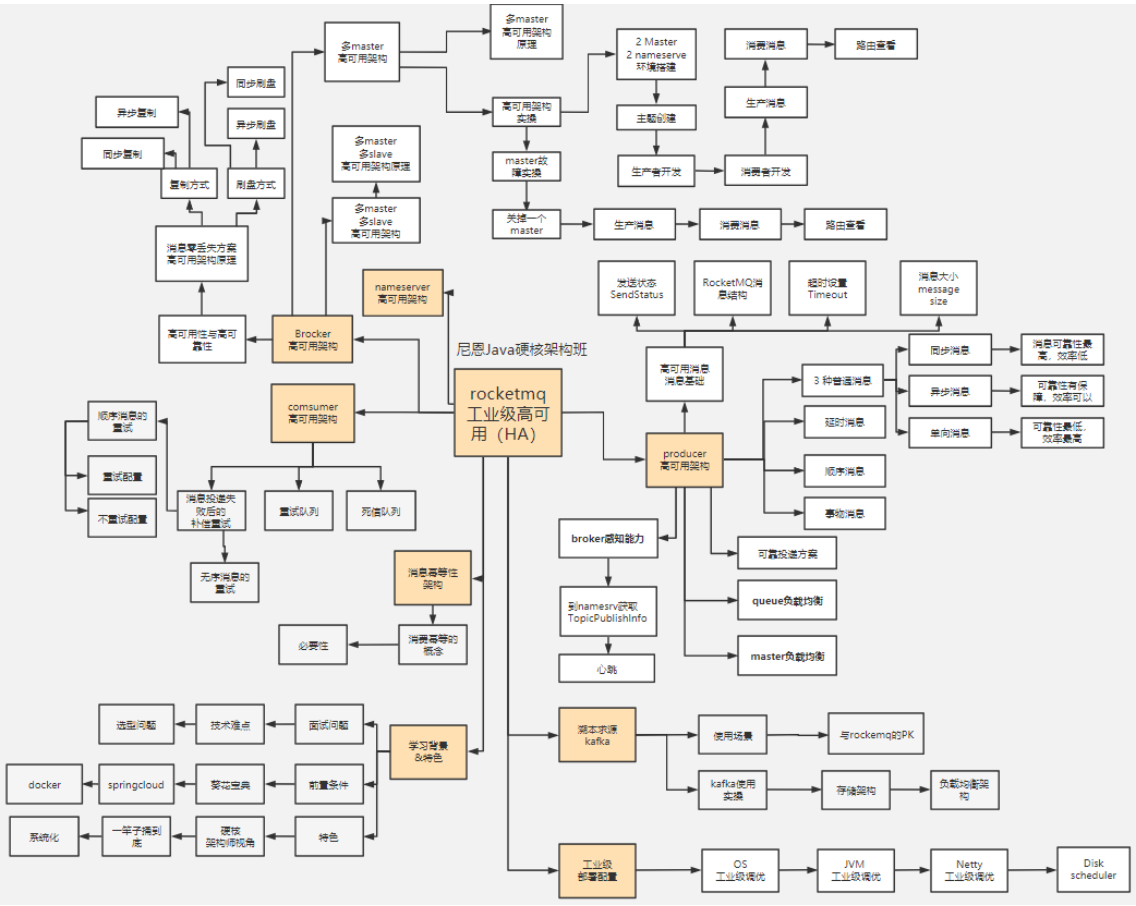
解决以下难题：

1、技术难题：RocketMQ 如何最大限度的保证消息不丢失的呢？RocketMQ 消息如何做到高可靠投递？

2、技术难题：基于消息的分布式事务，核心原理不理解

3、选型难题： kafka or rocketmq ，该娶谁？

下图链接：https://www.processon.com/view/6178e8ae0e3e7416bde9da19



成功案例：2 年翻 3 倍，35 岁卷王成功转型为架构师

详情：http://topcoder.cloud/forum.php?mod=forumdisplay&fid=43&page=1

最新 最后发表 热门 精华

- 成功案例：[1057号卷王] 3年小伙拿到外企offer，薪酬涨了200% — 卷王1号 超级版主 前天 17:41
- 成功案例：[645号卷王] 4年经验卷王逆袭，被毕业后，反涨24W — 卷王1号 超级版主 2022-9-21
- 成功案例：[878号卷王] 小伙8年经验，年薪60W — 卷王1号 超级版主 2022-8-13
- 年薪70W案例：通过尼恩的指导，小伙伴年薪从40W涨到70W — 卷王1号 超级版主 2022-2-11
- 成功案例：[493号卷王] 5年小伙拿满意offer，就业寒冬季逆涨30% — 卷王1号 超级版主 前天 17:43
- 成功案例：[250号卷王] 就业极寒时代，收offer 涨25% — 卷王1号 超级版主 前天 17:38
- 成功案例：[612号卷王] 就业极寒时代，从外包到自研 — 卷王1号 超级版主 前天 17:15
- 成功案例：[913号卷王] 热烈祝贺6年经验卷王，年薪40W — 卷王1号 超级版主 2022-9-21
- 成功案例：[959号卷王] 4年经验卷王，喜获百度、Boss直聘等N个优质offer，最高涨100% — 卷王1号 超级版主 2022-9-21
- 成功案例：[529号卷王] 5年经验卷王喜收2大offer，最高涨5K — 卷王1号 超级版主 2022-9-21
- 成功案例：[811号卷王] 热烈祝贺7年经验卷王，薪酬涨30% — 卷王1号 超级版主 2022-9-21
- 成功案例：[287号卷王] 不惧大寒潮，卷王逆市收4 offer，涨30%，可喜可贺 — 卷王1号 超级版主 2022-5-30
- 成功案例：[1002号卷王] 5月份"被毕业"，改简历后，斩获顶级央企Offer，涨薪7000+ — 卷王1号 超级版主 2022-7-5
- 成功案例：[7号卷王] 热烈祝贺小伙伴涨薪120% — 卷王1号 超级版主 2022-8-13
- 成功案例：[134号卷王] 大三小伙卷1年，斩获顶级央企Offer，成功逆袭 — 卷王1号 超级版主 2022-7-6
- 成功案例：[1008号卷王] 5年经验卷王收42W offer，月涨8000，可喜可贺 — 卷王1号 超级版主 2022-5-30
- 成功案例：[453号卷王] 非全日制 6年卷王喜提3 offer，年薪30W，可喜可贺 — 卷王1号 超级版主 2022-5-21
- 成功案例：[924号卷王] 6年卷王喜提4 offer，最高涨薪9000，可喜可贺 — 卷王1号 超级版主 2022-5-21
- 成功案例：[15号卷王] 4年卷王入职 微软，涨薪50%，可喜可贺 — 卷王1号 超级版主 2022-5-12
- 成功案例：[527号卷王] 4年卷王喜提2 offer，涨薪50%，可喜可贺 — 卷王1号 超级版主 2022-5-13
- 成功案例：[788号卷王] 3年卷王喜提优质Offer，涨薪60% — 卷王1号 超级版主 2022-5-11
- 成功案例：热烈祝贺：非全日制卷王，喜提2个心仪offer，面3家过2家 — 卷王1号 超级版主 2022-4-21

最新 最后发表 热门 精华

- 成功案例：[693号卷王] 二线城市6年卷王喜提4大优质Offer，含央企offer，最高薪酬35W — 卷王1号 超级版主 2022-4-16
- 成功案例：[85号卷王] 双非2本小伙，春招大捷，喜提9个offer，最高薪酬近30万 — 卷王1号 超级版主 2022-4-14
- 成功案例：[741号卷王] 卷王逆袭！6年小伙从很少面试机会到搞定35K*14薪Offer — 卷王1号 超级版主 2022-4-12
- 成功案例：[642号卷王] 热烈祝贺，6年卷王喜提优质国企offer — 卷王1号 超级版主 2022-4-7
- 成功案例：[796号卷王] 热烈祝贺，36岁卷王喜提52万优质offer — 卷王1号 超级版主 2022-3-25
- 成功案例：[15号卷王] 小伙卷1年，涨薪9K+，喜收ebay等多个优质offer — 卷王1号 超级版主 2022-3-24
- 成功案例：[821号卷王] 小伙狠卷3个月，喜提10多个offer — 卷王1号 超级版主 2022-3-21
- 成功案例：[736号卷王] 3年半经验收22k offer，但是小伙志存高远，冲击25k+ — 卷王1号 超级版主 2022-3-20
- 成功案例：热烈祝贺1群小卷王offer拿到手软，甚至拒了阿里offer — 卷王1号 超级版主 2022-3-16
- 简历案例：简历一改，腾讯的邀请就来了！热烈祝贺，小伙收到一大堆面试邀请 — 卷王1号 超级版主 2022-3-10
- 成功案例：祝贺我圈两大超级卷王，一个过了阿里HR面，一个过了阿里2面 — 卷王1号 超级版主 2022-3-10
- 成功案例：小伙伴php转Java，卷1.5年Java，涨薪50%，喜收多个优质offer — 卷王1号 超级版主 2022-3-10
- 成功案例：4年小伙狠卷半年，拿到 移动、京东 两大顶级offer — 尼恩 超级版主 2022-3-5
- 成功案例：[267号卷王] 助力3年经验卷王，拿到蜂巢的17k x 14薪的offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[143号卷王] 二本院校00后卷神，毕业没到一年跳到宇节，年薪45W — 卷王1号 超级版主 2022-2-27
- 成功案例：[494号卷王] 尼恩分布式事务助力卷王拿到 中信银行offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[76号卷王] 2线城市卷王，狠卷1.5年，喜收22K offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[429号卷王] 小伙伴在社群卷5个月，涨8k+ — 卷王1号 超级版主 2022-2-27
- 成功案例：[154号卷王] 双非学校毕业卷王，连拿 京东到家&滴滴 两个大厂Offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[232号卷王] 涨薪10K，继续卷向食物链顶端 — 卷王1号 超级版主 2022-2-27
- 成功案例：狠卷1年技术，喜收 腾讯、阿里、微软三大Offer，最高年薪56W — 卷王1号 超级版主 2022-2-27
- 成功案例：[449号卷王] 应届毕业卷王喜收 滴滴offer，年薪33W — 卷王1号 超级版主 2022-2-27

- 成功案例：[732号卷王] 尼恩助力3年经验卷王收获 京东offer，年薪35W — 卷王1号 超级版主 2022-2-27
- 成功案例：[558号卷王] 2年经验卷王，喜收 网易和阿里子公司两个优质offer — 卷王1号 超级版主 2022-2-27
- 成功案例：[569号卷王] 双非应届生卷王，喜收字节跳动实习offer — 卷王1号 超级版主 2022-2-25
- 成功案例：[420号卷王] 狠卷1年，卷王涨薪80%，涨薪12000元！ — 卷王1号 超级版主 2022-2-25
- 成功案例：[76号卷王] 通过尼恩1年半的指导，专科学历小伙伴从0.8K涨到22K — 卷王1号 超级版主 2022-2-10

- 成功案例：[551号卷王] 小伙伴学完后，成功进入大厂，并且推荐自己的朋友加VIP学习 — 卷王1号 超级版主 2022-2-10
- 成功案例：[214号卷王] 助力2年经验卷王，成功拿到17K月薪 — 卷王1号 超级版主 2022-2-10
- 成功案例：[92号卷王] 课程实操助力社群小伙伴喜收 喜马拉雅Offer — 卷王1号 超级版主 2022-2-10
- 成功案例：社群卷王小伙伴成功过了滴滴三面 获滴滴Offer — 卷王1号 超级版主 2022-2-10
- [612号卷王]滴滴小伙伴，蹲点考察半年，觉得靠谱后加入 疯狂创客圈 — 卷王1号 超级版主 2022-2-10

简历优化后的成功涨薪案例（VIP 含免费简历优化）

修改简历找尼恩（资深简历优化专家）

- 如果面试表达不好，尼恩会提供 简历优化指导
- 如果项目没有亮点，尼恩会提供 项目亮点指导
- 如果面试表达不好，尼恩会提供 面试表达指导

作为 40 岁老架构师，尼恩长期承担技术面试官的角色：

- **从业以来，“阅历”无数，对简历有着点石成金、改头换面、脱胎换骨**的指导能力。
- **尼恩指导过刚刚就业的小白，也指导过 P8 级的老专家，都指导他们上岸。**

如何联系尼恩。尼恩微信，请参考下面的地址：

语雀：https://www.yuque.com/crazymakercircle/gkkw8s/khigna
码云：https://gitee.com/crazymaker/SimpleCrayIM/blob/master/疯狂创客圈总目录.md