Raft 算法（1）：如何选举领导者

jask

2024-08-11

Raft 算法

Raft 算法属于 Multi-Paxos 算法，它是在兰伯特 Multi-Paxos 思想的基础上，做了一些简化和限制，比如增加了日志必须是连续的，只支持领导者、跟随者和候选人三种状态。

Raft 算法是现在分布式系统开发首选的共识算法。

一句话概括 Raft 算法，我觉得是这样的：从本质上说，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。这句话比较抽象，我来做个比喻，领导者就是 Raft 算法中的霸道总裁，通过霸道的“一切以我为准”的方式，决定了日志中命令的值，也实现了各节点日志的一致。

有哪些成员

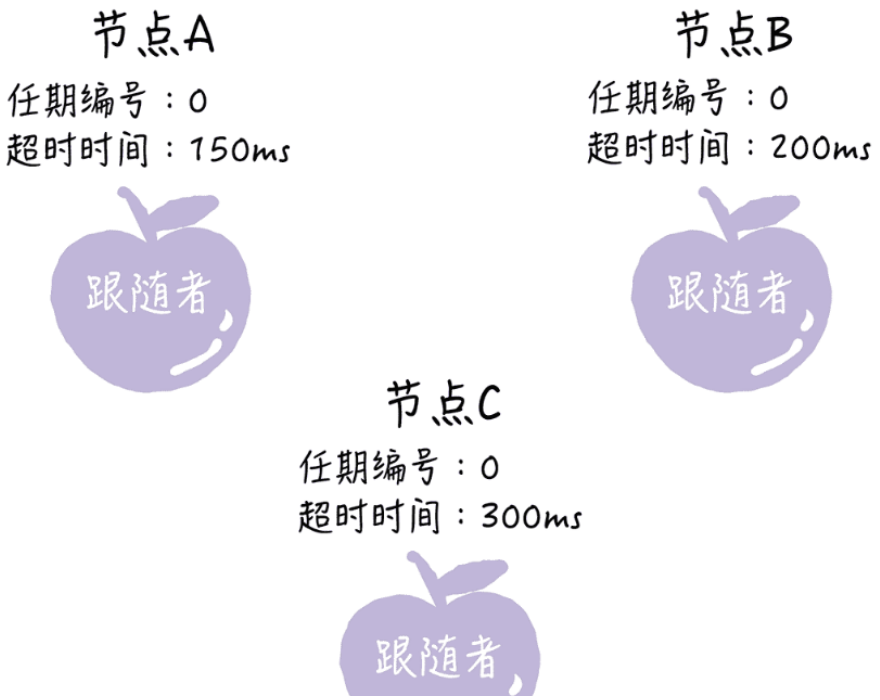
成员身份，又叫做服务器节点状态。
Raft 算法支持领导者（Leader），跟随者（Follower），候选人（Candidate）三种状态。

跟随者：就相当于普通群众，默默地接收和处理来自领导者的消息，当等待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：候选人将向其他节点发送请求投票（RequestVote）RPC 消息，通知其他节点来投票，如果赢得了大多数选票，就晋升当领导者。领导者：蛮不讲理的霸道总裁，一切以我为准，平常的主要工作内容就是 3 部分，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，你们现在不要发起新的选举，找个新领导者来替代我。”

Raft 算法是强领导者模型，集群中只能有一个“霸道总裁”。

选举领导者的过程

在初始状态下，集群中的所有节点都是跟随者的状态。



Raft 算法实现了随机超时时间的特性。也就是说，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。通过上面的图片你可以看到，集群中没有领导者，而节点 A 的等待超时时间最小（150ms），它会最先因为没有等到领导者的心跳信息，发生超时。

这个时候，节点 A 就增加自己的任期编号，并推举自己为候选人，先给自己投一张选票，再向其他节点发送请求投票 RPC 消息，请他们选举自己为领导者。

如果其他节点接收到候选人 A 的请求投票 RPC 消息，在编号为 1 的这届任期内，也还没有进行过投票，那么它将把选票投给节点 A，并增加自己的任期编号。

如果候选人在选举超市时间内赢得大多数的选票，它就会成为本届任期内的新的领导者。

节点 A 当选领导者后，他将周期性地发送心跳消息，通知其他服务器我是领导者，阻止跟随者发起新的选举，篡权。

可能的问题

节点间如何通讯？ 在 Raft 算法中，服务器节点间的沟通联络采用的是远程过程调用（RPC），在领导者选举中，需要用到这样两类的 RPC：

1. 请求投票（RequestVote）RPC，是由候选人在选举期间发起，通知各节点进行投票；
2. 日志复制（AppendEntries）RPC，是由领导者发起，用来复制日志和提供心跳消息。

日志复制 RPC 只能由领导者发起，这是实现强领导者模型的关键。

什么是任期? 议会选举中的领导者是有任期的，领导者任命到期后，要重新开会再次选举。Raft 算法中的领导者也是有任期的，每个任期由单调递增的数字（任期编号）标识，比如节点 A 的任期编号是 1。任期编号是随着选举的举行而变化的，这是在说下面几点。

跟随者在等待领导者心跳信息超时后，推举自己为候选人时，会增加自己的任期号，比如节点 A 的当前任期编号为 0，那么在推举自己为候选人时，会将自己的任期编号增加为 1。如果一个服务器节点，发现自己的任期编号比其他节点小，那么它会更新自己的编号到较大的编号值。比如节点 B 的任期编号是 0，当收到来自节点 A 的请求投票 RPC 消息时，因为消息中包含了节点 A 的任期编号，且编号为 1，那么节点 B 将把自己的任期编号更新为 1。

Raft 算法中的任期不只是时间段，而且任期编号的大小，会影响领导者选举和请求的处理。

在 Raft 算法中约定，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。比如分区错误恢复后，任期编号为 3 的领导者节点 B，收到来自新领导者的，包含任期编号为 4 的心跳消息，那么节点 B 将立即恢复成跟随者状态。还约定如果一个节点接收到一个包含较小的任期编号值的请求，那么它会直接拒绝这个请求。比如节点 C 的任期编号为 4，收到包含任期编号为 3 的请求投票 RPC 消息，那么它将拒绝这个消息。

选举有哪些规则? 1. 领导者周期性地向所有跟随者发送心跳消息（即不包含日志项的日志复制 RPC 消息），通知大家我是领导者，阻止跟随者发起新的选举。

2. 如果在指定时间内，跟随者没有接收到来自领导者的消息，那么它就认为当前没有领导者，推举自己为候选人，发起领导者选举。

3. 在一次选举中，赢得大多数选票的候选人，将晋升为领导者。

4. 在一个任期内，领导者一直都会是领导者，直到它自身出现问题（比如宕机），或者因为网络延迟，其他节点发起一轮新的选举。

5. 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，并且按照“先来先服务”的原则进行投票。比如节点 C 的任期编号为 3，先收到了 1 个包含任期编号为 4 的投票请求（来自节点 A），然后又收到了 1 个包含任期编号为 4 的投票请求（来自节点 B）。那么节点 C 将会把唯一一张选票投给节点 A，当再收到节点 B 的投票请求 RPC 消息时，对于编号为 4 的任期，已没有选票可投了

1. 当任期编号相同时，日志完整性高的跟随者（也就是最后一条日志项对应的任期编号值更大，索引号更大），拒绝投票给日志完整性低的候选人。比如节点 B、C 的任期编号都是 3，节点 B 的最后一条日志项对应的任期编号为 3，而节点 C 为 2，那么当节点 C 请求节点 B 投票给自己时，节点 B 将拒绝投票。

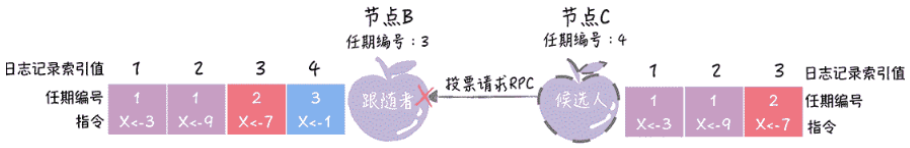


Figure 1: 投票规则

选举是跟随者发起的，推举自己为候选人；大多数选票是指集群成员半数以上的选票；大多数选票规则的目标，是为了保证在一个给定的任期内最多只有一个领导者。

如何理解随机超时时间? 在多个候选人同时发起选举，导致选票被瓜分的情况下，Raft 利用随即超时时间解决选举无效的问题。

在 Raft 算法中，随机超时时间是有 2 种含义的：

1. 跟随者等待领导者心跳信息超时的时间间隔，是随机的；

2. 当没有候选人赢得过半票数，选举无效了，这时需要等待一个随机时间间隔，也就是说，等待选举超时的时间间隔，是随机的。

总结

Raft 算法和兰伯特的 Multi-Paxos 不同之处，主要有 2 点。首先，在 Raft 中，不是所有节点都能当选领导者，只有日志最完整的节点，才能当选领导者；其次，在 Raft 中，日志必须是连续的。Raft 算法通过任期、领导者心跳消息、随机选举超时时间、先来先服务的投票原则、大多数选票原则等，保证了一个任期只有一位领导，也极大地减少了选举失败的情况。本质上，Raft 算法以领导者为中心，选举出的领导者，以“一切以我为准”的方式，达成值的共识，和实现各节点日志的一致。