Raft

一、角色和基本定义

Leader: 处理来自客户端的请求,(如果客户端将请求发给了 follower，则 follower 将该请求的方向改到 leader)，负责日志的同步管理，通过向 follower 发送 heartbeat 来保持领导权。

Candidate：候选状态，当 raft 刚启动或者一个 follower 在规定时间内没有接收到来自 leader 的 heartbeat 时，follower 则会转换成 candidate 状态，并向其他 follower 发送邀票请求，如果一个 candidate 接收到来自一个 majority 的投票，则该 candidate 转换为 leader 状态

Follower：处于被动状态，响应 Leader 的日志同步请求，给 Candidate 进行投票，把请求到 Follower 的事务转发给 Leader；

Term：每个 term 都是一个不定期同一个 leader 任期的时间段，当 leader 值更新的时候，term 值递增。Raft 保证每个 term 都只会有一个 leader，

二、Leader Election

1、 RPCs(remote procedure calls):

Raft 算法中有以下两种 RPCs：

- RequestVote RPCs:由 candidate 发出，用来向其他的服务器申请投票。
- AppendEntries RPCS：由 leader 发出，用来要求其他服务器复制 log entries，同时 heartbeat 也是一种定期向 follower 发送的 AppendEntries RPCS，用来保证 leader 的领导权。

2、 当 raft 算法开始的时候，所有的服务器都处于 follower 的状态，每个 follower 都设置有一个倒计时称为 timeout，率先过完 timeout 时间的 follower 成为 candidate 并向其他的 follower 发送 RequestVote RPCs 申请投票（candidate 会首先给自己投票），其他 follower 如果在该 term 还没有给其他的 candidate 投票则会给该 candidate 投票，此时可能出现三种情况：

1) 该 candidate 在此轮竞选中胜出，成为 leader：
   如果一个 candidate 收到集群中大多数（majority）的服务器的投票，则成为 leader 状态，并开始向其他所有的服务器发送 heartbeat 来申明它的领导权防止其他的 leader election 产生。
2) 其他的服务器已经率先成为了 leader：如果一个 candidate 在等待投票的过程中收到来自其他的自称为 leader 的服务器的 AppendEntries RPCS，则如果该 RPCs 的 term 值小于该 candidate 的 term 值，则 candidate 拒绝该 RPCs 并继续等待投票；如果该 RPCs 的 term 值大于或等于该 candidate 的 term 值，则该 leader 是合法的，candidate 自动退回 follower 状态。
3) 没有服务器成为 leader：如果多个 follower 同时成为 candidate，并在竞选的过程中都没有得到大多数的投票（成为 splitVote），则没有服务器胜出成为 candidate。当这种情况发生的时候，则更新 term 值，开始新一轮的 leader election，所有的服务器重新等待 timeout 成为 candidate 并重新竞选。但是同样的 splitVote 的情况仍然很有可能发生。于是 raft 采用 random timeout 机制，每个服务器的 timeout 值都是在一个区间内（如 150ms-300ms）随机生成的一个数值，这样多个服务器同时 timeout 成为 candidate 的可能性就会变得很小，

而如果有 splitvote 的产生，随着 term 的增加，还会同时 timeout 的几率就会变得越爱越小，因此有效的防止了 splitvote 的产生，使得 leaderelection 不会产生活锁，保证了 leaderelection 的顺利进行。

三、Log Replication

日志复制（Log Replication）主要作用是用于保证节点的一致性，这阶段所做的操作也是为了保证一致性与高可用性；当 Leader 选举出来后便开始处理客户端的请求，所有更新操作请求都必须先经过 Leader 处理，这些请求或说成命令也就是这里说的日志，我们都知道要保证节点的一致性就要保证每个节点都按顺序执行相同的操作序列，日志复制（Log Replication）就是为了保证执行相同的操作序列所做的工作；在 Raft 中当接收到客户端的日志（事务请求）后先把该日志追加到本地的 Log 中，然后通过 heartbeat 把该 Entry 同步给其他 Follower，Follower 接收到日志后记录日志然后向 Leader 发送 ACK，当 Leader 收到大多数（n/2+1）Follower 的 ACK 信息后将该日志设置为已提交并追加到本地磁盘中，通知客户端并在下个 heartbeat 中 Leader 将通知所有的 Follower 将该日志存储在自己的本地磁盘中。

四、时间限制

为保证 raft 程序的 progress，须有以下时间限制：

broadcastTimeout <=electionTimeout<=MTBF

其中：

- broadcastTime：是指一台服务器向集群中的其他媚态服务器发送 RPCs 并且收到回应的时间。
- electionTimeout：一轮竞选的最长时间，不管产生是否产生 leader，盖伦竞选都会结束
- MTBF：平均每台服务器在宕机之前能够正常工作的时间。