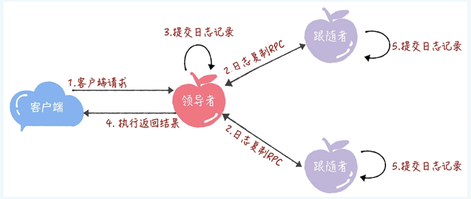
- Raft
  - 领导者选举
    - 节点状态
      - 领导者
      - 跟随者
      - 候选者
    - 任期
      - 任期编号
      - Raft算法中的任期，不只是时间段，而且，任期编号大小会影响领导者选举和请求处理
        - 在 Raft 算法中约定，如果一个候选人或者领导者，发现自己的任期编号比其他节点小，那么它会立即恢复成跟随者状态。比如分区错误恢复后，任期编号为 3 的领导者节点 B，收到来自新领导者的，包含任期编号为 4 的心跳消息，那么节点 B 将立即恢复成跟随者状态。
        - 还约定如果一个节点接收到一个包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。
    - 通讯
      - RPC
    - 规则
      - 领导者的心跳信息，能阻止跟随者发起新的选举
      - 指定时间内，没收到领导者的信息，跟随者推举自己为候选人，发起选举
      - 赢得大多数选票的候选人，将晋升为领导者
      - 在一次选举中，每一个服务节点，最多会对一个任期编号，投出一张选票 ，先来先服务的投票原则
      - 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。
      - 当任期编号相同时，日志完整性高的跟随者，将拒绝投票给日志完整性低的获选人
    - 随机超时时间
      - 第一种含义，跟随者，等待领导者心跳信息超时，然后发起选举的随机时间间隔
      - 第二种含义，当没有候选人赢得过半票数，选举无效时，等待选举超时，然后再发起新一轮选举的时间间隔
    - 日志，一切以领导者为准
    - Raft 算法和兰伯特的 Multi-Paxos 不同之处，主要有 2 点
      - 在 Raft 中，不是所有节点都能当选领导者，只有日志较完整的节点（也就是日志完整度不比半数节点低的节点），才能当选领导者
      - 在 Raft 中，日志必须是连续的
  - 日志复制
    - 过程
      - 在 Raft 算法中，副本数据是以日志的形式存在的，领导者接收到来自客户端写请求后，处理写请求的过程就是一个复制和应用（Apply）日志项到状态机的过程。
      - 组成，由日志项组成
        - 指令
          - 一条由客户端指定的状态机需要执行的指令
        - 索引值
          - 整数索引，用来标识日志项的，单调递增、连续的整数
        - 任期编号
          - 创建这条日志的领导者的任期编号
      - 优化后的二阶段提交协议
        - 第一阶段：通过日志复制RPC，将日志项复制到集群中其他节点，接着如果领导者接收到大多数的“复制成功”响应后，它将日志项应用到它的状态机，并返回成功给客户端。如果领导者没有接收到大多数的“复制成功”响应，那么就返回错误给客户端。
        - 无第二阶段，因为领导者的日志复制 RPC 消息或心跳消息，包含了当前最大的，将会被提交（Commit）的日志项索引值。所以通过日志复制 RPC 消息或心跳消息，跟随者就可以知道领导者的日志提交位置信息。因此，当其他节点接受领导者的心跳消息，或者新的日志复制 RPC 消息后，就会将这条日志项应用到它的状态机。而这个优化，降低了处理客户端请求的延迟，将二阶段提交优化为了一段提交，降低了一半的消息延迟。
      - 例图
        - 

        - 接收到客户端请求后，领导者基于客户端请求中的指令，创建一个新日志项，并附加到本地日志中。
        - 领导者通过日志复制 RPC，将新的日志项复制到其他的服务器。
        - 当领导者将日志项，成功复制到大多数的服务器上的时候，领导者会将这条日志项应用到它的状态机中。此时日志视为commit
        - 领导者将执行的结果返回给客户端。
        - 当跟随者接收到心跳信息，或者新的日志复制 RPC 消息后，如果跟随者发现领导者已经提交了某条日志项，而它还没应用，那么跟随者就将这条日志项应用到本地的状态机中。
    - 保证日志的一致性
      - 首先，领导者通过日志复制 RPC 的一致性检查，找到跟随者节点上，与自己相同日志项的最大索引值。也就是说，这个索引值之前的日志，领导者和跟随者是一致的，之后的日志是不一致的了。
      - 然后，领导者强制跟随者更新覆盖的不一致日志项，实现日志的一致
      - 一切以领导者的日志为主，不会删除领导者的日志项
  - 成员变更
    - 成员变更的问题
      - 主要在于进行成员变更时，可能存在新旧配置的 2 个“大多数”，导致集群中同时出现两个领导者，破坏了 Raft 的领导者的唯一性原则，影响了集群的稳定运行。
    - 单节点变更方案
      - 单节点变更是利用“一次变更一个节点，不会同时存在旧配置和新配置 2 个‘大多数’”的特性，实现成员变更。