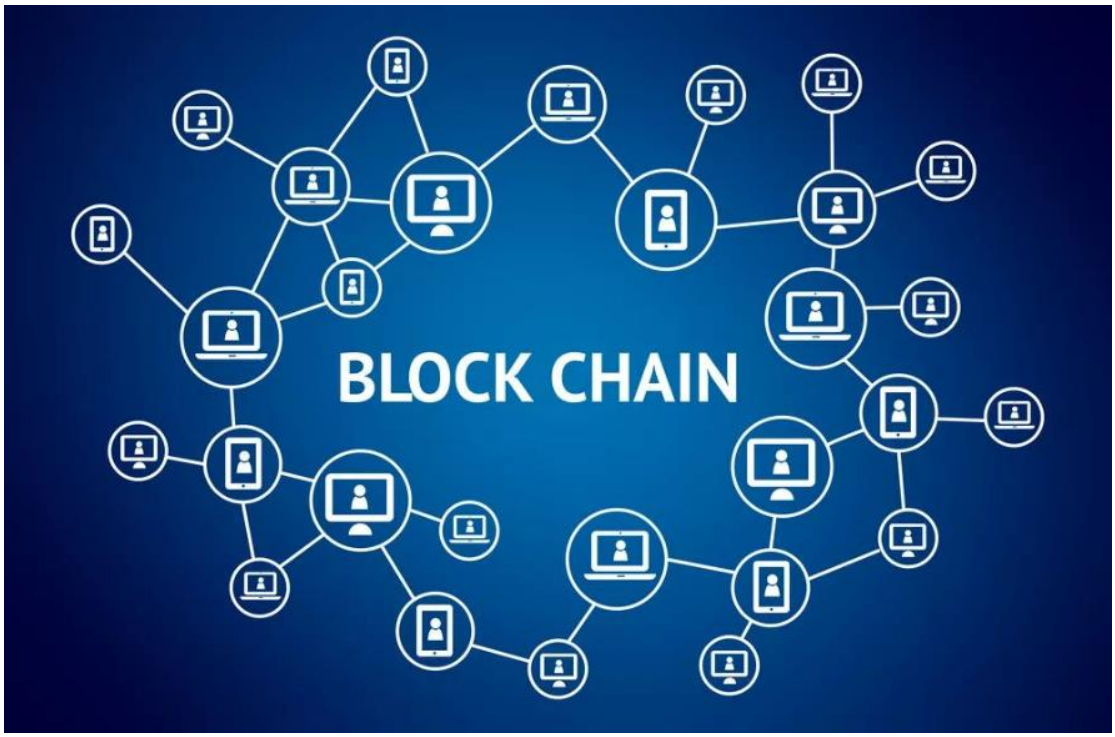
XDL 兄弟连教育
www.itxdl.cn



兄弟连. 孙建超

邮箱：sunjianchao@itxdl.cn

Raft——分布式一致性算法

一、Raft 介绍

1. Raft 是什么

- Raft 提供了一种在计算系统集群中分布状态机的通用方法，确保集群中的每个节点都同意一系列相同的状态转换
- 它有许多开源参考实现，具有 Go，C ++，Java 和 Scala 中的完整规范实现
- 一个 Raft 集群包含若干个服务器节点，通常是 5 个，这允许整个系统容忍 2 个节点的失效，每个节点处于以下三种状态之一
  - follower（跟随者）：所有节点都以 follower 的状态开始。如果没收到 leader 消息则会变成 candidate 状态
  - candidate（候选人）：会向其他节点“拉选票”，如果得到大部分的票则成为 leader，这个过程就叫做 Leader 选举(Leader Election)
  - leader（领导者）：所有对系统的修改都会先经过 leader

2. Raft 一致性算法

- Raft 通过选出一个 leader 来简化日志副本的管理，例如，日志项(log entry)只允许从 leader 流向 follower
- 基于 leader 的方法，Raft 算法可以分解成三个子问题
  - Leader election (领导选举)：原来的 leader 挂掉后，必须选出一个新的 leader
  - Log replication (日志复制)：leader 从客户端接收日志，并复制到整个集群中
  - Safety (安全性)：如果有任意的 server 将日志项回放到状态机中了，那么其他的 server 只会回放相同的日志项

3. Raft 动画演示

- 地址：http://thesecretlivesofdata.com/raft/
- 动画主要包含三部分：
  - 第一部分介绍简单版的领导者选举和日志复制的过程
  - 第二部分介绍详细版的领导者选举和日志复制的过程
  - 第三部分介绍如果遇到网络分区（脑裂），raft 算法是如何恢复网络一致的

4. Leader election (领导选举)

- Raft 使用一种心跳机制来触发领导人选举
- 当服务器程序启动时，节点都是 follower(跟随者) 身份

- 如果一个跟随者在一段时间里没有接收到任何消息，也就是选举超时，然后他就会认为系统中没有可用的领导者然后开始进行选举以选出新的领导者
- 要开始一次选举过程，follower 会给当前 term 加 1 并且转换成 candidate 状态，然后它会并行的向集群中的其他服务器节点发送请求投票的 RPCs 来给自己投票。
- 候选人的状态维持直到发生以下任何一个条件发生的时候
  - 他自己赢得了这次的选举
  - 其他的服务器成为领导者
  - 一段时间之后没有任何一个获胜的人

5. Log replication (日志复制)

- 当选出 leader 后，它会开始接收客户端请求，每个请求会带有一个指令，可以被回放到状态机中
- leader 把指令追加成一个 log entry，然后通过 AppendEntries RPC 并行地发送给其他的 server，当该 entry 被多数 server 复制后，leader 会把该 entry 回放到状态机中，然后把结果返回给客户端
- 当 follower 宕机或者运行较慢时，leader 会无限地重发 AppendEntries 给这些 follower，直到所有的 follower 都复制了该 log entry
- raft 的 log replication 要保证如果两个 log entry 有相同的 index 和 term，那么它们存储相同的指令
- leader 在一个特定的 term 和 index 下，只会创建一个 log entry

二、安全性

1. 选举限制

- 在一些一致性算法中，即使一台 server 没有包含所有之前已提交的 log entry，也能被选为主，这些算法需要把 leader 上缺失的日志从其他的 server 拷贝到 leader 上，这种方法会导致额外的复杂度
- raft 使用一种更简单的方法，即它保证所有已提交的 log entry 都会在当前选举的 leader 上，因此，在 raft 算法中，日志只会从 leader 流向 follower
- 为了实现上述目标，raft 在选举中会保证，一个 candidate 只有得到大多数的 server 的选票之后，才能被选为主
- 得到大多数的选票表明，选举它的 server 中至少有一个 server 是拥有所有已经提交的 log entry 的，而 leader 的日志至少和 follower 的一样新，这样就保证了 leader 肯定有所有已提交的 log entry

2. 提交之前任期内的日志条目

- 领导人知道一条当前任期内的日志记录是可以被提交的，只要它被存储到了大多数的服务器上。如果一个领导人在提交日志条目之前崩溃了，未来后续的领导人会继

续尝试复制这条日志记录

- 然而，一个领导人不能断定一个之前任期里的日志条目被保存到大多数服务器上的时候就一定已经提交了

3. 时间和可用性

- 领导人选举是 Raft 中对时间要求最为关键的方面。Raft 可以选举并维持一个稳定的领导人，系统需要满足以下时间要求
- 广播时间（broadcastTime） << 选举超时时间（electionTimeout） << 平均故障间隔时间（MTBF）
  - 广播时间指的是从一个服务器并行的发送 RPCs 给集群中的其他服务器并接收响应的平均时间
  - 选举超时时间就是选举的超时时间限制
  - 平均故障间隔时间就是对于一台服务器而言，两次故障之间的平均时间
- 选举超时时间要大于广播时间的原因是，防止跟随者因为还没收到领导者的心跳，而重新选主
- 选举超时时间要小于 MTBF 的原因是，防止选举时，能正常工作的 server 没有达到大多数
- 对于广播时间，一般在[0.5ms,20ms]之间，而平均故障间隔时间一般非常大，至少是按照月为单位。因此，一般选举超时时间一般选择范围为[10ms,500ms]。因此，当领导者挂掉后，能在较短时间内重新选主

三、Go 实现 Raft 选举

四、RPC 的案例

五、Raft 选举添加 RPC

六、Raft 分布式选举的实现