分布式共识算法

- ## 相关概念
  - 在软件系统里，要**保障系统的可靠性**，软件系统必须有多台机器能够拥有一致的数据副本，才有可能对外提供可靠的服务。
  - 在软件系统里，要**保障系统的可用性**，必须考虑动态的数据如何在不可靠的网络通信条件下，依然能在各个节点之间正确复制的问题。
  - **状态转移：**以同步为代表的数据复制方法，较符合人类思维的可靠性保障手段，但通常要以牺牲可用性为代价，建设分布式系统的时候，往往不能承受这样的代价。
  - **操作转移：**分布式系统里主流的数据复制方法，通过某种操作，令源状态转换为目标状态。能够使用确定的操作，促使状态间产生确定的转移结果的计算模型，在计算机科学中被称为状态机State Machine。
  - **状态机模型：**根据状态机特性，要让多台机器最终状态一致，只要确保初始状态是一致的，并且接收到的操作指令序列也是一致的即可。将任何行为理解为将操作日志正确地广播给各个分布式节点。广播指令与指令执行期间，不要求所有节点每一条指令都同时开始、同步完成，只要求在此期间的内部状态不能被外部观察到。
  - **Quorum 机制：**一旦系统中过半数的节点中完成了状态的转换，就认为数据的变化已经被正确地存储在系统当中，这样就可以容忍少数（通常是不超过半数）的节点失联，使得增加机器数量对系统整体的可用性变成是有益的。
  - **协商共识：**让系统各节点不受局部的网络分区、机器崩溃、执行性能或者其他因素影响，都能最终表现出整体一致的过程。能够让分布式系统内部暂时容忍存在不同的状态，但最终能够保证大多数节点的状态达成一致；同时，能够让分布式系统在外部看来始终表现出整体一致的结果。
- ## Paxos
  - **定义：**一种基于消息传递的协商共识算法，是当今分布式系统最重要的理论基础。
  - **分布式系统节点分类**
    - **提案节点Proposer：**提出对某个值进行设置操作的节点，设置值这个行为称为提案Proposal，值一旦设置成功就不会丢失也不可变。Paxos 是典型的基于操作转移模型来设计的算法，“设置值”应该类比成日志记录操作。
    - **决策节点Acceptor：**是应答提案的节点，决定该提案是否可被投票、是否可被接受。提案一旦得到过半数决策节点的接受，即称该提案被批准，提案被批准即意味着该值不能再被更改，也不会丢失，且最终所有节点都会接受该它。
    - **记录节点Learner：**不参与提案，也不参与决策，单纯地从提案、决策节点中学习已经达成共识的提案，譬如少数派节点从网络分区中恢复时，将会进入这种状态。
    - **原则：**所有的节点都是平等的，都可以承担以上某一种或者多种的角色，为了便于确保有明确的多数派，决策节点的数量应该被设定为奇数个，且在系统初始化时，网络中每个节点都知道整个网络所有决策节点的数量、地址等信息。
  - **分布式系统值达成一致影响因素**
    - 系统内部各个节点通信是不可靠的，不论对于系统中企图设置数据的提案节点抑或决定是否批准设置操作的决策节点，其发出、收到的信息可能延迟送达、也可能会丢失，但不去考虑消息有传递错误的情况。
    - 系统外部各个用户访问是可并发的，如果系统只会有一个用户，或者每次只对系统进行串行访问，那单纯地应用 Quorum 机制，就已经足以保证值被正确地读写。
  - **分布式系统加锁分析：**对同一个变量的并发修改必须先加锁后操作，在分布式的环境下要考虑到可能出现的通信故障，如果一个节点在释放锁之前发生崩溃失联，这将导致整个操作被无限期的等待所阻塞，因此算法必须提供一个其他节点能抢占锁的机制，以避免因通信问题而出现死锁，因此分布式环境中的锁必须是可抢占的。
  - **阶段**
    - **准备Prepare**
      - 承诺不会再接受提案 ID 小于或等于 n 的 Prepare 请求。
      - 承诺不会再接受提案 ID 小于 n 的 Accept 请求。
      - 不违背承诺的前提下，回复已经批准过的提案中 ID 最大的提案设定的值和提案 ID，如果该值从来没有被提案设定过则返回空值。如果违反此前的承诺，收到的提案 ID 并不是决策节点收到过的最大的，那允许直接对此 Prepare 请求不予理会。
    - **批准Accept**
      - 1-1、如果提案节点发现所有响应的决策节点此前都没有批准过该值（即为空），那说明它可以随意地决定要设定的值，将自己选定的值与提案 ID，构成一个二元组“(id, value)”，再次广播给全部的决策节点（称为 Accept 请求）
      - 1-2、如果提案节点发现响应的决策节点中，已经有至少一个节点的应答中包含有值了，必须无条件地从应答中找出提案 ID 最大的那个值并接受，构成一个二元组“(id, maxAcceptValue)”，再次广播给全部的决策节点（称为 Accept 请求）
      - 2、每一个决策节点收到 Accept 请求时，会在不违背作出的承诺的前提下，接收并持久化对当前提案 ID 和提案附带的值。如果违反此前做出的承诺，即收到的提案 ID 并不是决策节点收到过的最大的，那允许直接对此 Accept 请求不予理会。
      - 3、当提案节点收到了多数派决策节点的应答（称为 Accepted 应答）后，协商结束，共识决议形成，将形成的决议发送给所有记录节点进行学习。
  - **缺陷：**Basic Paxos 只能对单个值形成决议，并且决议的形成至少需要两次网络请求和应答（准备和批准阶段各一次），高并发情况下将产生较大的网络开销，极端情况下甚至可能形成活锁。
- ## Multi Paxos
  - **改进：**增加了“选主”的过程，提案节点通过定时轮询（心跳），发现没有主提案节点存在时就会在心跳超时后使用准备、批准的两轮网络交互过程，向所有其他节点广播竞选主节点的请求，希望各节点对由该节点作为主节点这件事情协商达成一致共识。如果得到了决策节点中多数派的批准便竞选成功。之后除非主节点失联之后发起重新竞选，否则从此往后，就只有主节点本身才能够提出提案。
  - **过程：**提案节点接收到客户端的操作请求，将请求转发给主节点来完成提案，而主节点提案的时无需再次经过准备过程，可理解为选主过后，相当于是处于无并发的环境当中进行的有序操作，此时要对某个值达成一致，只要进行一次批准的交互。
  - **广播的数据区别：**二元组变成三元组(id, i, value)，给主节点增加了任期编号，且保证严格单调递增，以应付主节点陷入网络分区后重新恢复，但另外一部分节点仍然有多数派且已经完成了重新选主的情况，此时以任期编号大的主节点为准。

- **分布式系统中如何对某个值达成一致问题划分**
  - **如何选主Leader Election：**即使用Basic Paxos的准备、批准阶段完成。
  - **如何把数据复制到各个节点上Entity Replication**
    - **正常情况下：**客户端向主节点发起一个操作请求，主节点将变更的值写入自己的变更日志，接着把变更的信息在下一次心跳包中广播给所有的从节点，从节点收到信息后，将操作写入自己的变更日志，然后给主节点发送确认签收的消息，主节点收到过半数的签收消息后，提交自己的变更、应答客户端并且给从节点广播可以提交的消息，从节点收到提交消息后提交自己的变更，数据在节点间的复制宣告完成。
    - **异常情况下：**网络出现了分区，部分节点失联，但只要仍能正常工作的节点的数量能够满足多数派（过半数）的要求，分布式系统就仍然可以正常工作。
  - **如何保证过程是安全的Safety**
    - **协定性Safety：**所有的坏事都不会发生
    - **终止性Liveness：**所有的好事都终将发生，但不知道是啥时候。
    - 譬如以选主问题为例，Safety 保证了选主的结果一定是有且只有唯一的一个主节点；而 Liveness 则要保证选主过程是一定可以在某个时刻能够结束的。

Gossip 协议

- **背景：**Paxos、Raft、ZAB 等分布式算法经常会被称作是“强一致性”的分布式共识协议（尽管系统内部节点可以存在不一致的状态，但从不一致的情况并不会被外部观察到）。还有另一类“最终一致性”的分布式共识协议，这表明系统中不一致的状态有可能会在一定时间内被外部直接观察到，比如DNS系统和Gossip协议。
- **工作过程**
  - 如果有某一项信息需要在整个网络中所有节点中传播，那从信息源开始，选择一个固定的传播周期，随机选择它相连接的 k 个节点（称为 Fan-Out）来传播消息。
  - 每一个节点收到消息后，如果这个消息是它之前没有收到过的，将在下一个周期内，选择除了发送消息给它的那个节点外的其他相邻 k 个节点发送相同的消息，直到最终网络中所有节点都收到了消息。
- **优势**
  - 对网络节点的连通性和稳定性几乎没有任何要求，一开始就将网络某些节点只能与一部分节点部分连通而不是以全连通网络为前提。
  - 能够容忍网络上节点的随意地增加或者减少，随意地宕机或者重启，新增加或者重启的节点的状态最终会与其他节点同步达成一致。
  - 把网络上所有节点都视为平等而普通的一员，没有中心化节点或者主节点的概念，这些特点使得 Gossip 具有极强的鲁棒性，而且非常适合在公众互联网中应用。
- **缺陷**
  - 消息通过多个轮次散播而到达全网，必然存在全网各节点状态不一致的情况。
  - 由于随机选取发送消息的节点，尽管可以在整体上测算出传播速率，但对于个体消息来说，无法准确地预计到需要多长时间才能达成全网一致。
  - 由于随机选取发送消息的节点，不可避免的存在消息重复发送给同一节点的情况，增加了网络的传输的压力，也给消息节点带来额外的处理负载。