共识算法

1 共识算法简介
共识算法是指在分布式场景中，多个节点为了达成相同的数据状态而运行的一种分布式算法。 在分布式场景中，可能出现网络丢包、时钟漂移、节点宕机、节点作恶等等故障情况，共识算法需要能够容忍这些错误，保证多个节点取得相同的数据状态。

根据可容忍的故障类型的不同，可以将共识算法分为两类：

容忍宕机错误类算法，可以容忍网络丢包、时钟漂移、部分节点宕机这种节点为良性的错误。常见算法有 Paxos、Raft。
容忍拜占庭错误类算法，可以容忍部分节点任意类型错误，包括节点作恶的情况。常见算法有 PBFT、PoW、PoS等。
根据使用场景的不同，又可将共识算法分为公链共识、联盟链共识两类。

典型的分布式三节点场景N1，N2，N3，刚启动触发 leader election，节点状态分leader，candidater，follower，选主采用抢注机制，完成选主后为了保证leader是靠谱的，需要有keepalive机制，若leader保活超时会触发新一轮选举。

leader节点会负责资源分配，负载均衡（LB），同时会负责log同步，状态机对齐等共识决策，以此来达成数据一致性

1.1. 公链共识

公链的特点是节点数量多且节点分布分散，主要使用的共识算法有PoW和PoS，这两种共识的优点是可以支持的节点数量多，缺点是TPS较低和交易确认时间长。

1.2. 联盟链共识

联盟链的特点是节点之间网络较为稳定且节点有准入要求，根据需要容忍的错误类型可以选择Raft和PBFT类算法，这类算法的优点是TPS较高且交易可以在毫秒级确认，缺点是支持的节点数量有限，通常不多于100个节点。

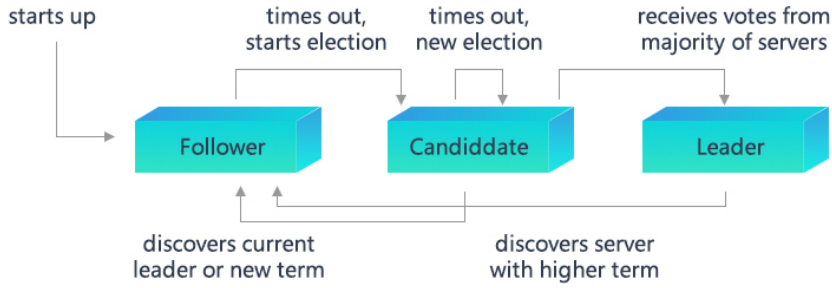
2.联盟链共识算法，RAFT一致性共识算法

Raft算法是目前使用最广泛的非拜占庭容错类共识算法。 Raft算法主要依靠 投票机制和 日志复制机制来实现节点间的共识。节点通过投票选出一个leader，由leader负责处理所有请求，再将请求以日志的方式复制到其他节点。

Raft保证了在一个由N个节点构成的系统中有(N+1)/2（向上取整）个节点正常工作的情况下的系统的一致性，比如在一个5个节点的系统中允许2个节点出现非拜占庭错误，如节点宕机、网络分区、消息延时。Raft相比于Paxos更容易理解，且被证明可以提供与Paxos相同的容错性以及性能，其详细介绍可见官网及动态演示。

ps：在Raft算法中，每个网络节点只能如下三种身份之一：Leader、Follower以及Candidate，其中:

Leader：主要负责与外界交互，由Follower节点选举而来，在每一次共识过程中有且仅有一个Leader节点，由Leader全权负责从交易池中取出交易、打包交易组成区块并将区块上链；
Follower：以Leader节点为准进行同步，并在Leader节点失效时举行选举以选出新的Leader节点；
Candidate：Follower节点在竞选Leader时拥有的临时身份。



Follower增加当前的Term，转换为Candidate；

Candidate将票投给自己，并广播RequestVote到其他节点请求投票；

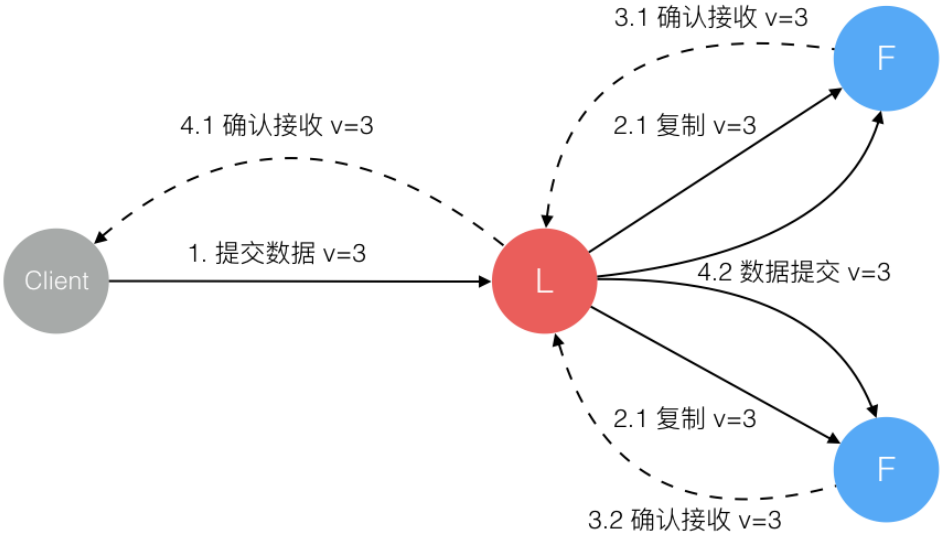
Candidate节点保持在Candidate状态，直到下面三种情况中的一种发生：

该节点赢得选举，,即自己获得了超过半数以上的服务器投票,成为 Leader；

在等待选举期间，Candidate收到了其他节点的Heartbeat，有其他节点获得了半数以上的投票；

经过Election Timeout后，没有Leader被选出。

Raft算法采用随机定时器的方法来避免节点选票出现平均瓜分的情况以保证大多数时候只会有一个节点超时进入Candidate状态并获得大部分节点的投票成为Leader。



Raft 协议强依赖 Leader 节点的可用性来确保集群数据的一致性。数据的流向只能从 Leader 节点向 Follower 节点转移。

当 Client 向集群 Leader 节点提交数据后，Leader 节点接收到的数据处于未提交状态（Uncommitted），

接着 Leader 节点会并发向所有 Follower 节点复制数据并等待接收响应，确保至少集群中超过半数节点已接收到数据后再向 Client 确认数据已接收。

一旦向 Client 发出数据接收 Ack 响应后，表明此时数据状态进入已提交（Committed），Leader 节点再向 Follower 节点发通知告知该数据状态已提交。

2.2 PBFT，实用拜占庭容错

简介

将所有的副本组成的集合使用大写字母R表示，使用0到|R|-1的整数表示每一个副本。为了描述方便，假设|R|=3f+1，这里f是有可能失效的副本的最大个数。尽管可以存在多于3f+1个副本，但是额外的副本除了降低性能之外不能提高可靠性

PBFT算法可以容忍小于1/3个无效或者恶意节点。最多能容忍的作恶/故障节点为(n-1)/3个。系统的总节点数为：|R| = 3f + 1。

证明过程

因为我们知道有f个作恶节点，所以我们必须在n-f个状态复制机的沟通内，就要做出决定。而且我们无法预测这f个作恶节点做了什么（错误消息/不发送），所以我们并不知道，这n-f个里面有几个是作恶节点，我们必须保证正常的节点大于作恶节点数。所以有 n-f-f > f，从而得出了n > 3f。

主节点确定机制：

主节点通过视图编号以及节点数集合来确定，即：主节点 p = v mod |R|。v：视图编号，|R|节点个数，p：主节点编号。

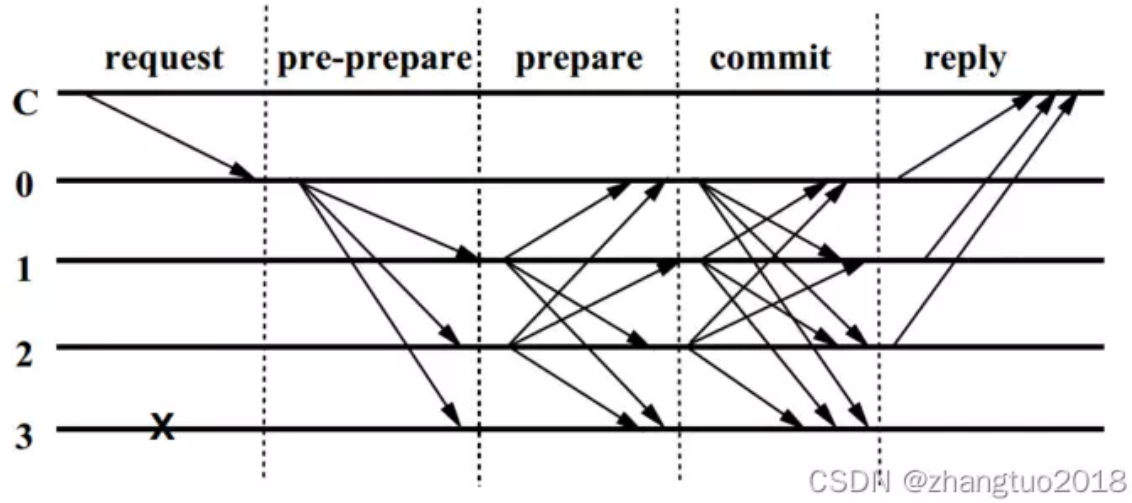
角色

Client： 客户端节点，负责发送交易请求。
Primary: 主节点，负责将交易打包成区块和区块共识，每轮共识过程中有且仅有一个Primary节点。
Replica: 副本节点，负责区块共识，每轮共识过程中有多个Replica节点，每个Replica节点的处理过程类似。
其中，Primary和Replica节点都属于共识节点。

基本流程

PBFT 算法的基本流程主要有以下四步：

1. 客户端发送请求给主节点
2. 主节点广播请求给其它节点，节点执行PBFT算法的三阶段共识流程。
3. 节点处理完三阶段流程后，返回消息给客户端。
4. 客户端收到来自 f+1 个节点的相同消息后，代表共识已经正确完成。



对比点	Raft	Pbft
适用环境	私链	联盟链
算法通信复杂度	O（n）	O（n^2）
最大故障和容错节点	故障节点：2f+1<=N	容错节点：3f+1<=N
流程对比	1.初始化leader选举（谁快谁当） 2.共识过程 3.重选leader机制	1.初始化leader选举（按编号依次轮流做主节点） 2.共识过程 3.重选leader机制

3 公链共识算法

3.1 POW，工作量证明

简单的理解就是通过一份证明来确定做过一定量的工作，好比说你是王者对吧，那么肯定是通过大量的rank的胜利来获得的。

3.1.1 名称解释

- 区块头结构

  区块头的大小为80字节，由4字节的版本号、32字节的上一个区块的哈希值、32字节的Merkle根哈希值、4字节的时间戳（当前时间）、4字节的目标值、4字节的随机数组成。

字段名	长度/字节	含义
version	4	版本号
prev_hash	32	前一区块哈希值
merkle_root	32	区块体中交易的 Merkle 树的根节点的哈希值
time	4	当前区块生成时间，unix 时间戳
target	4	当前目标值
nonce	4	随机数

难度值

比特币的区块大约每10分钟生成一个，如果要在不同的全网算力条件下，新区块的产生都基本保持这个速率，难度值必须根据全网算力的变化进行调整。简单地说，难度值被设定在无论节点计算能力如何，新区块产生速率都保持在每10分钟一个。

难度的调整是在每个完整节点中独立自动发生的。每2016个区块，所有节点都会按统一的公式自动调整难度，这个公式是由最新2016个区块的花费时长与期望时长（期望时长为20160分钟，即两周，是按每10分钟一个区块的产生速率计算出的总时长）比较得出的，根据实际时长与期望时长的比值，进行相应调整（或变难或变易）。也就是说，如果区块产生的速率比10分钟快则增加难度，比10分钟慢则降低难度。

这个公式可以总结为：新难度值=旧难度值×（过去2016个区块花费时长/20160分钟）

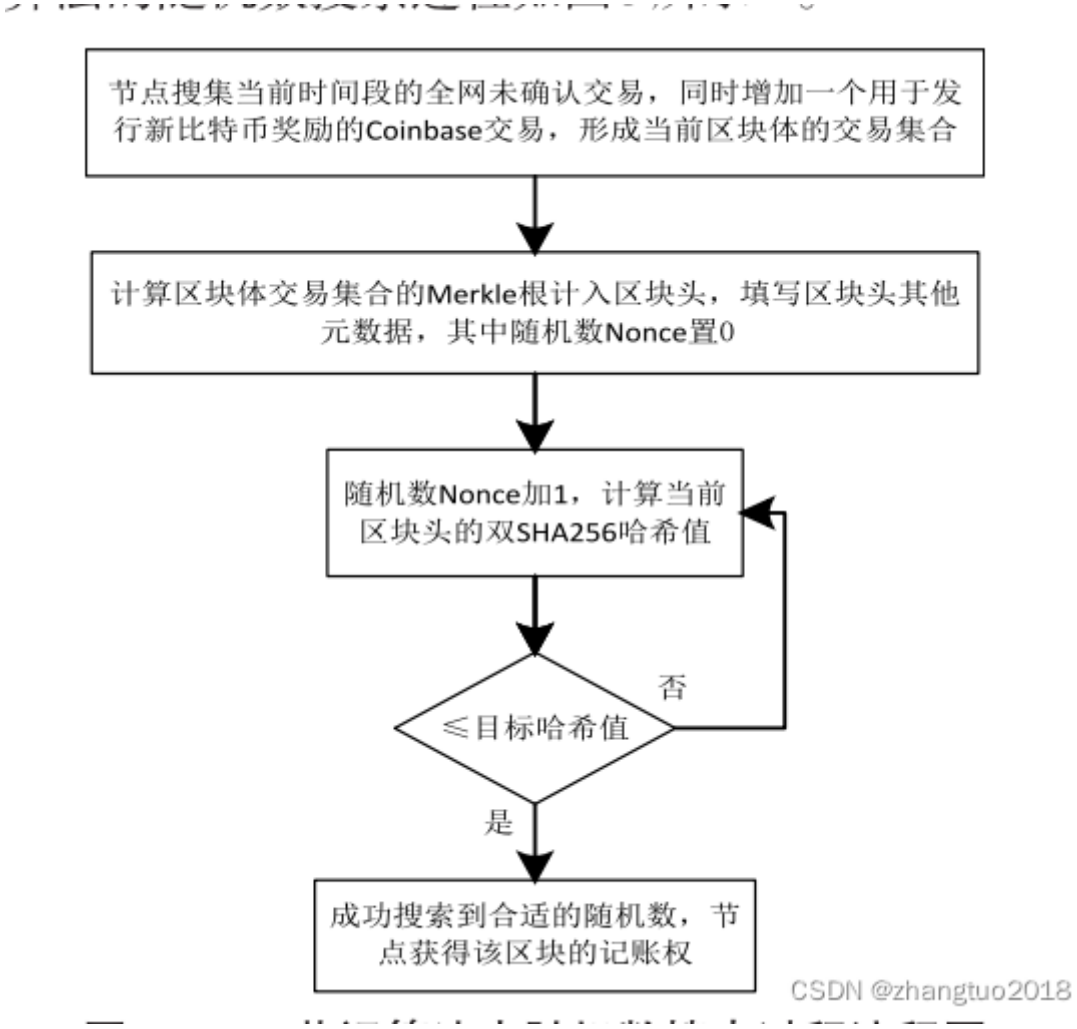
目标值

比特币工作量证明的目标值（Target）的计算公式：目标值=最大目标值/难度值

其中最大目标值为一个恒定值，难度为1时的目标值，即2^224：
0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

目标值的大小与难度值成反比。比特币工作量证明的达成就是矿工计算出来的区块哈希值必须小于目标值。

过程



POS，权益证明

PoS(PowofStake)即股权证明,是为解决 PoW 算法 大量浪费资源问题而提出的一种替代算法,该算法与 PoW 算 法的最大不同点在于区块的记账权由权益最高的节点获得, 而不是算力最高的节点.

共识过程中节点通过消耗的币龄来获取记账权,节点消耗的币龄越多,获得记账权的机会越 大.算法设定的主链原则为:消耗币龄最多的链为系统中正确有效的链.点点币中 PoS算法合格区块的判定公式为:

ProofHash < Target×CoinAge

其中,ProofHash对应一组数据的哈希,此处省略;CoinAge 为币龄（coin*age）;Target为当前目标值,同 PoW 一样,与难度成反比.

Target = PrevTarget×(1007×10×60 + 2 ×Tgap)/1009× 10×60

其中,PrevTarget为上一个区块的目标值; 2 ×Tgap 为前两个 区块的间隔时间.

当前两个区块时间间隔大于10 min时,当前目标值相比 于上一个区块目标值会提高,即当前区块难度会降低;

反之, 当前两个区块时间间隔小于10 min,当前目标值相比于上一 个区块目标值会降低,即当前区块难度会提高.

PoS 将算力竞争转化为权益竞争, 不仅节约算 力, 权益的引入也能够防止节点发动恶意攻击; 同 时促使所有节点有责任维护区块链的安全稳定运行 以保障自身权益; PoS 虽然降低了算力资源的消耗, 但是没有解决中心化程度增强的问题, 新区块的生 成趋向于权益高的节点。PoS 中需要拥有超全网一半 的权益发动 51%攻击, 但其成本高于拥有超全网一 半的算力[70], 另外创建区块需要消耗权益, 使得 PoS 持续进行 51%攻击的难度增加, 一定程度上降低了 安全风险

DPOS

它是 PoS的一种衍生算法,算法的思想是系统中持有权益的节点投票选举出一部分代表，再由这些代表轮流获取区块记账权，某种程度上类似于股份制公司的“董事会”。它的原理是让每一个持有比特股的人进行投票，由此产生101位代表 。我们可以将其理解为101个超级节点或者矿池，而这101个超级节点彼此的权利完全相等。

DPoS算法将节点分为两部分:参与投票的选民和被选举出的代表.

每个节点都可将自己持有的权益转化为选票投给 自己信任的节点,所持的权益越多,选票所占的权重也就越 高,获得票数最多的n 个节点当选为见证人(Witness)即 代 表.见证人的名单每经过一个固定周期都将重新选举更换一 次.见证人直接参与区块的共识和验证过程,在一个规定的 时间内它们随机排列并轮流对交易进行打包,生成新区块连 接到最长链上.每生成一个区块,见证人会获得 m%的交易 手续费,m 值由见证人们提议设定并由选民们投票决议.当 然,参与见证人的竞选也需缴纳大量的保证金,这样,见证人在系统中投入的资金最大,为保证自身的利益积极地维护系 统安全.

DPOS通过其选择区块生产者和验证节点质量的算法确保了安全性，同时消除了交易需要等待一定数量区块被非信任节点验证的时间消耗。通过减少确认的要求，DPOS算法大大提高了交易的速度。通过信任少量的诚信节点，可以去除区块签名过程中不必要的步骤。

表 3 共识算法性能特点总结

共识算法	核心思想	优点	缺点
PoW	拥有最高算力的节点易获得新区块的记账权和区块奖励，再由全网节点验证区块的正确性	算法简单，可操作性强，节点自由加入或者退出，可扩展性强	严重浪费电力等资源，依赖专业挖矿硬件资源，算力集中在几大矿场之间，有中心化风险且效率低下，交易吞吐量小
PoS	拥有最高权益的节点易获得新区块的记账权和区块奖励，再由全网节点验证区块的正确性	降低了 PoW 机制算力的浪费，节点不再依赖专业挖矿硬件资源，节点自由加入或者退出，可扩展性强	需要完善解决 PoS 机制存在的“无利害关系”问题，吞吐量小，一些平台手续费高
DPoS	由拥有权益的节点选出前 N 个超级节点作为新区块的记账节点，这些受理人轮流获得记账权和区块奖励	高吞吐量，更快地确认时间，降低能源损耗，节省了区块确认时间，减少了交易延迟，吞吐量得到极大提高，节点自由加入或者退出，可扩展性强	超级节点投票麻烦，投票期间需要锁定代币，不利于激发普通节点参与，固定数量超级节点的竞选规则令人担忧，去中心化的实现存在争议
PBFT	主节点排序请求，从节点响应请求，多数节点响应结果为最终结果	共识结果的一致性和正确性程度高，确认时间快	算法复杂度较高，通信量大；当节点数量过多时，运行效率较低