## 基础知识

货币的价值: 交换价值, 使用价值 货币的形式: 贵金属, 纸币, 数字货币

经济学:马克思主义:商品的使用价值和 jiao 换价值; 凯恩斯经济学: 市场有时候会失灵, 需要干预;哈耶克经济学:自由市场、自由经 营、自由竞争、自动调节、自动均衡, 货币非 国有化;现代货币理论 MMT: 货币的本质就是 欠条,即IOU (I Owe You)

Hash 算法: Hash 实际上是一种思想,包含很 多算法。

HASH 应用: 快速定位 (数据库中散列表) ;错 误校验;唯一性验证

HASH 常见算法: MD4、MD5、SHA1、SHA2 hash 算法: 逆向困难, 抗碰撞

SHA256 输出为 256 位,输出空间是 2^256, 是"亿亿亿亿亿亿亿亿亿"。

非对称加密两种算法: RSA 算法: 大素数分解

椭圆曲线加密: 离散对数难解

## 区块链技术原理

任何数字货币都会面临虚假货币和多重支付的 问题。比特币解决方法,虚假货币(数字签名), 多重支付 (分布式账本)。

比特币核心技术

1. 以链式区块组织账本数据实现账本数据的 不可篡改

2. 分布式的可信记账机制

区块链基于哈希值进行链接, 特点是区块链中 数据无法篡改或删除; 区块链越长可信度越高 区块链中每个区块包括区块头和交易数据两个 部分, 其中

1. 区块头由当前区块的元数据和前一区块的 Hash 值构成

2. Merkle 树用于对交易数据列表进行快速寻 址

区块大小

区块头

交易数量

交易信息

产生的时间/秒

## 区块的微观结构

4 字节 80 字节 1-9 字节 可变

区块头 交易计数器 交易

区块大小

区块头结构 版本号 4字节 版本 父区块头 hash 值 32 字节

32 字节 merkel 根 hash 值 4 字节 时间戳 4 字节 难度

4字节 nonce

pow 随机数 注意区块头中的前一个区块的 hash 值为其区块 头的 hash 值,不是整个区块。

## 区块的区分方式

1. 区块主标识符是它的加密哈希值, 一个通过 SHA256 算法对区块头进行二次哈希计算而得 到的数字指纹。

2. 第二种识别区块的方式是通过该区块在区 块链中的位置,即"区块高度 (block height)" 例如: 高度为0的区块就是创世区块。和区块 哈希值不同的是, 区块高度并不是唯一的标识 符, 因为有可能出现区块链分叉。

## merkel 树

Merkle 树是一种哈希二叉树, 它是一种用作快 速归纳和校验大规模数据完整性的数据结构。 这种二叉树包含加密哈希值。叶节点是数据块 的哈希值。非叶节点的哈希值是根据它下面子 节点的值哈希计算得到。在比特币网络中, Merkle 树被用来归纳一个区块中的所有交易, 同时生成整个交易集合的数字指纹, 且提供了

一种校验区块是否存在某交易的高效途径。 Merkle 树中使用两次 SHA256 算法计算结点的 哈希值、H~A~= SHA256(SHA256(交易 A))。 两次 SHA256 是为了提高安全强度。当 N 个数 据元素经过加密后插入 Merkle 树时, 你至多计 算 log2(N)次就能检查出任意某数据元素是否 在该树中, 这使得该数据结构非常高效。

merkel 树的价值: 快速比较大量数据、快速定 位修改、快速验证其中数据 (merkel 路径) 比特币节点

全节点: 存储着整个区块链, 承但对交易请求 进行验证 和执行,可以通过挖矿争取发布区块, 还承担着应别的节点之请向其发送区块和相关 交易信息的义务,同时也承担转发交易请求和 区块的义务。 轻节点:

1. 简单支付验证(SPV)节点: 只存储区块头, 不 存储区块块体,仍可以对到来的交易请求进行 验证。

交易验证过程: 为交易建立布隆过滤器, 只接 受指定目标地址的交易, 其他全节点探测到某 个交易符合 SPV 节点设置的布隆过滤器条件时, 以 Merkleblock 消息的形式发送该区块, Merkleblock 消息包含区块头和 一条连接目标 交易与 Merkle 根的 Merkle 路径。

 交易的存在性验证:SPV 节点通过该 Merkle 路径找到跟该交易相关的区 块, 并验证对应区 块中是否存在目标交易(Merkle Path Proof)。

 交易是否双化验证:SPV 节点检查这笔交易所 在区块之后的区块个数, 区块个数越多说明该 区块被全网更多节点共识, 一般来说, 一笔交 易所 属区块之后的区块个数达到6个时,说明 这笔交易是可信的。

2. 钱包:一个连接区块链的应用软件(app), 记录 与所有者有关的信息: 区块链地址、私钥、账 户余额、UTXO等,不存储账本。

Class transaction

] const std::vector<CTxIn> vin //输入 utxo l const std::vector<CTxOut> vout //输出 utxo ] const int32 t nVersion//版本

] const uint32 t nLockTime//锁定时间, 在此之 前交易不讲入区块

1 const uint256 hash //交易 hash. 不存储不发送 工作量证明

PBFT 要求节点数量有限, 点集相对稳定, 所以 不适用.

pow 机制: 划定固定时间段(10 分钟)

。相同或相似输入数据(组装的区块)

。 算力竞争选出获胜节点, 其它节点验证结果 后不再发送消息。

最长链原则, 从短期共识扩展到长期共识 pow 机制中难度的确定方式:在每个完整节点

中独立自动发生的。每2,016个区块中的所有 节点都会调整难度。难度的调整公式是由最新 2,016 个区块的花费时长与 20,160 分钟 (两周, 即这些区块以10分钟一个速率所期望花费的 时长) 比较得出的。难度是根据实际时长与期 望时长的比值进行相应调整的(或变难或变易) 简单来说, 如果网络发现区块产生速率比 10 分 钟要快时会增加难度。如果发现比10分钟慢时 则降低难度。

比特币分叉处理机制: 把当前同一个父区块下 的若干有效子区块都记录, 形成兄弟区块 (产 生分叉) •后续区块 (第3代、第4代.....) 到 达后, 依次加在前序区块后, 若没有其他竞争 性区块, 这一分支最长, 成为主链。

比特币挖矿的奖励机制: 挖矿节点必须有钱包

功能,有自己的160位密码地址、私钥。 打包生成区块时, 区块中额外加一个交易 coinbase 生成一个 UTXO. 包含当前奖励数量 的比特币 (现在是 6.25)。这个 UTXO 的招领地 址是自己的地址, 如果记账成功, 这个 coinbase 交易就生效, 否则不在链上。

## 挖矿机制的总结:

每个全节点依据综合标准对每个交易进行独立 验证,通过完成工作量证明算法的验算,挖矿 节点将交易记录独立打包进新区块, 每个节点 独立地对新区块进行校验并组装进区块链每个 节点对区块链进行独立选择, 在工作量证明机 制下选择累计工作量最大的区块链 UTXO 模型: 每个比特币用户有一个 160 位 (20

字节) 长度的地址,产生的过程:用户生产一 对非对称密钥, 公钥经 hash 计算(SHA160) 产生 160 位的地址, 私钥自己保存, 用于数字 签名, 这个 160 位地址, 是用户在比特币网络 中交易的唯一标识。与一般的银行账户模型不 同, 比特币不维护每个账户的资金余额, 而是 采用一种称为 UTXO 的模型。 比特币中有两类交易

常规交易, 有交易输入 (支付者地址和金额) 交易输出(收入者地址和金额);挖矿交易 (Coinbase),产生比特币,只有交易输出(挖 矿者地址和金额)。

每个地址的资金余额就是散布在账本中所有 UTXO 的总和, 使用时把自己名下的 UTXO 作 为交易输入,可能需要拼凑找零。

ctxout 结构 = nValue(比特币价值) scriptPubKey(招领脚本),不一定是公钥 scriptPubKey 可以分为

## P2PK,P2PKH,P2SH,P2WPKH,P2WSH 交易输入的结构

|COutPoint prevout 资金来源 ]] uint256 hash 存储了交易的 hash 值 ]] uint32 t n 是上述交易的第几个输出 1 CScript scriptSig //认领脚本

1 uint32 t nSequence //特殊作用

] CScriptWitness scriptWitness //见证脚本 比特币虚拟机

比特币节点软件的一个模块、堆栈结构 运行的程序: 认领脚本 scriptSig、招领脚本 scriptPubKey

指令集: 有限能力, 主要验证签名是否正确 认领脚本 scriptPubKey

P2PK, "PaytoPublicKey", 付给公钥。付给

给定 256 位公钥的主人。 P2PKH, "PaytoPublicKeyHash", 付给公钥的

Hash 值。这里所谓的 Hash, 是特指对于 256 位公钥的 160 位 Hash, 那就是对方的"地址"。 P2SH, "PaytoScriptHash", 付给脚本的 Hash 值。给定一个脚本的 Hash 值, 付给能提供这个 脚本的对象, 所提供脚本的 Hash 值必须与给定 的 Hash 值相同。这个脚本是收付双方预先约定 的, 是双方在"链外"约定的。

需要多方签名并采用 SegWitness 脚本的支付. 具体又有两种:

P2WPKH, "PaytoSegWitnessPublicKeyHash" 付给 SegWitness 形式的多个公钥 Hash 值。 P2WSH, "PaytoSegWitnessscriptHash", 付给 SegWitness 形式的脚本 Hash 值。

通信方式: p2p

如果存在种子节点,新节点先和和种子节点通 信,得到伙伴列表

新节点和上述伙伴通信, 对有反应的请求新的

伙伴列表, 最终获得最新的伙伴列表

## 区块链技术的优点

去中心化:避免垄断,点对点交易,去代理 数据公开: 无暗箱操作, 平等, 开放生态体系 可信:数据永久存储,记录可信

## 比特币的问题 隐私问题

**署名地址如何监管** 性能问题

> 交易确认时间长 区块容量有限

系统性风险

区块链分叉和51%攻击

## 什么是比特币分叉

比特币协议分叉不同于区块链共识中常规分叉 而是因为协议升级或分歧, 导致遵循不同协议 的软件所产生区块链不兼容, 形成各自生长的

## 支付通道

用于链外支付或其他交易,向性能要求高、交 易数量大等场景 涉及的链上交易:

## 注资交易 (Funding Transaction)

A、B 双方建立一个 2-of-2 多签名的联合地址, 由拟议中的付方发布一个交易将一笔钱打到这 个地址中;但这是 P2SH 支付,即支付给能够 正确提供清算脚本 Hash 值的收款方, 因为是 2-of-2 就必须有双方的签名才能花, 。这是后 面链外支付的资金来源。

1个付款方: 单向通道, 2个付款方: 双向通道 应承交易 (Commitment Transaction)

A、B 双方的链下交易, 可以有很多个, 不上链。 每次支付的资金都来自同一个 UTXO, 每次交 易的输出分成两部分,一是给收方的 UTXO, 其数值是付方至此为止承诺支付的总和, 二是 给付方自己的找零。

## 决算交易 (Settlement Transaction)

收款方把手中由付款方开具并签名的最后那个 应承交易上签上自己的名并把它发送到比特币 网上,从当初注资阶段生成的那个 UTXO 中把 钱划给自己,同时把剩余的钱(如果还有的话) 找还给付款方。这个操作也可以由付款方发起, 因为收款方每次收到应承交易都会签上自己的 名并发还给付款方。

# 退款交易 (Refund Transaction)

## 闪电网络, 大致有以下几种支付方式

转存交易 (DeliveryTransaction) , 任何一方都 可以从联合地址认领属于自己的资金, 把它转 到一个自己更方便花费的地址中。

可撤转存交易 (RevocableDeliveryTransaction), 可以撤销。

救交易 (BreachRemedy), 这是在对方违约情 况下加以补救并使对方受惩罚的交易

# 数字货币和区块链生态

联盟锌 (ConsortiumChain) 对产业或者国家的 特定清算和结算用途, 容易进行控制权限设定, 更高的可扩展性。

公链 (任何人) 任何人都可以参与, 容易部署 应用程序, 全球范围可以访问, 不依赖于单个 公司

私链(内部链)由单独的个人或者组织拥有,对 组织内部的审计和测试有用。

无许可区块链不一定是公链 Scrypt hash 算法介绍

Scrypt 算法由 FreeBSD 黑客 Colin Percival 开发

的, 原来是用于密码抗 rainbow table 攻击设计

Scrypt 计算所需时间长, 而且占用的内存也多 使得并行计算多个摘要异常困难。

Scrypt 也是一种符合区块链 PoW 共识机制的算 法。Scrypt 算法过程中也需要计算哈希值,但 是 Scrypt 计算过程中需要使用较多的内存资源。 可以抗 ASIC 挖矿 (ASIC resistance) ASIC (Application Specific Integrated Circuit ) 是专 用集成电路芯片

## 以太坊

## 以太坊的组成

P2P 网络 以太坊以 P2P 方式进行网络通信 通过 TCP 端口 30303 访问。

交易 Transaction 以太坊交易是网络消息, 包括转账交易、合约交易等。

状态机 State Machine

以太坊的状态转移由 以太坊虚拟机(EVM) 处 理, 这是一个执行 bytecode(机器语言 指令)基于栈的虚拟机、称为"智能合约"的 EVM 程序以高级语言(如 Solidity)编写, 并编译 为字节码以便在 EVM 上执行。

## 区块链账本

以太坊的区块链账本存储在每个节点上,该区 块链在 Merkle Patricia Tree 的序列化 哈希数据结构中包含交易和系统状态。

共识算法 以太坊 1.0 使用名为 Ethash 的工作 量证明算法,以太坊 2.0 过渡到称为 Casper 的 是 0。 权益证

明机制(Proof-of-Stake)。

客户端

以太坊有几个可互操作的客户端软件实现, 最 著名的是 Go-Ethereum(Geth)和 Parity 以太坊技术架构

应用层: 数字钱包, Dapp, 以太坊应用 合约层: 智能合约 evm 激励层:发行机制、分配机制

共识层: pow, pos, dpos, pow+pbft, Pbft, poa 协议层: http, rpc, les, eth, whisper

网络层: p2p 网络、数据传输机制、数据校验

数据层:数据区块、链式交易、交易池、merkel 树、非对称加密、event 事件 存储层: leveldb, log

## 以太坊中的两种账户以及存储结构

外部账户:由区块链外部主体创建,拥有一对 公私钥, 账户地址没有合约代码。 合约账户:由合约交易创建, 账户拥有合约代

码, 地址是代码的 hash 值。 •nonce: 对外部账户, 代表该账户发出的交易数

对合约账户,表示该账户创建的合约数量。 •balance: 该账户地址的 Wei, 1018 Wei =

• storageRoot: Merkle Patricia 树的根节点 hash 值 , 该树是该账户下存储信息的 hash

值,缺省为0. •codeHash: 该账户的 EVM 代码 hash 值, 对于 合约账户, 是合约代码存储的 hash 值, 对外 部账户, 是空字符串的 hash 值。

# 合约账户和外部账户之间的区别

外部账户

•创建外部账户没有成本

•可以启动交易 •两个外部账户间的交易只能 是以太币转账交易 •由公私钥对控制 合约账户

•创建合约账户需要成本, 因为使用了网络存 储 •只能在收到一个交易作出响应而发出一个 交易 •从一个外部账户到一个合约账户的交易 可以触发合约账户上的代码, 执行代码 中的各 种动作, 例如代币转账、创建新合约等 •合约 帐户没有私钥, 相反, 它们由智能合约代码的

## 以太坊世界状态

以太坊本质上是一个状态机, 有一个 genesis 作 为初始状态,交易是其状态转换的最小单元(原 子性,一致性),每次执行一条或者多条交。 之后发生状态转换。

世界状态中外部账户的地址就是公钥计算得到, 合约账户的地址以太坊合约的地址是根据创建 者 (sender) 的地址以及创建者发送过的交易数 量 (nonce) 来计算确定的。 sender 和 nonce 进 行 RLP 编码, 然后用 Keccak-256 进行 hash 计

## 以太坊中交易的类型

简单支付交易 - 以太币的账户间转账, 从 知己账户转到对方账户,不涉及智能合约,无 需 动用以太坊虚拟机, "耗油"也最少。

存证交易 - 在简单支付交易中把支付额 设置成 0、需要存证的内容写在 data 字段中、 就成 了存证交易, 当然, 也可以为存证机制专 门部署一个智能合约, 以后要存证时就调用这 个合约, 这样可以为存证增添一些附加的操作。 因为是0支付,对方账户就无关紧要,但不能

合约部署交易 - 将对方地址设置成 0, 就可 以将智能合约的程序部署到以太坊网络中, 实 际上是所有的验证节点上。一旦交易记录进入 区块链即部署生效,以太坊网络会在交易"收 据"中返回新建合约账户的地址。

合约调用交易 - 对智能合约的调用, 依具 体智能合约的不同, 又可以分为以下几种:

复杂支付交易, 更确切地说是数字资产转移 交易, 这是对智能合约中代表着数字资 产的 Token 的转移。如果用 Token 实现各种虚拟货 币, 这样的转移就成了采用某种自 定义虚拟货 币的支付。

查询交易,查询是区块链网络中常用的操作。 以太坊网络中提供了若干"系统合约",即由 系统提供、无需用户部署的合约, 其中之一就 是用于查询.

其它(智能合约调用)交易, 如在电子商务, 电子政务等领域的应用。

## 交易的数据结构

transaction

] byte[] hash; //对 RLP 编码之后的交易请求 Tx 的 Hash 值。 ] byte[] nonce; //实质上是 Tx 的序号, 用以防止

对同一交易请求的重复处理。

] byte[] value; //支付币值

] byte[] receiveAddress; // 接受者地址 ] byte[] gasPrice; //油价

] byte[] gasLimit; // 最大油量 ] byte[] data;

] Integer chainId; // 链 id

] ECDSASignature signature; // 签名 ] byte[] sendAddress // 发送者地址

其中 data 在支付交易中可以作为付款说明,如 果 value=0, 那么可以作为存证交易(但是对方 账户不能是 0) , 在部署交易中 (对方账户地 址为 0) 表示合约本身, 在合约调用交易中,

含有函数名和参数。

## 以太坊中的 GAS

以太坊中的状态转移运算需要消耗 gas, 比如数 学运算, 状态存储, gas 是计算以太坊资源消耗 的最小单位。常见的 gas 消耗表如下 stop: 0, add:3, mul: 5, sub:3, div: 5

在上述结构中的 gasprice 表示预付的 gas 价格, 发送方必须有足够的 ether 余额来支付 gas 费用 产生的 gas 费用作为给 miner 的激励。

注意存储所使用的 gas 应该是 32 的倍数。

### 交易的最新数据结构

•recipient - 接收者地址 (如果是外部账户, 就转币值, 如果是合约账户, 就执行该地址上 的合约)

- •signature 发送者签名
- •nonce 该账户发起的交易计数器
- •value 转账的币值(in WEI)
- •data 可放置任意数据
- •gasLimit 本交易可消耗的 gas 上限
- ·maxPriorityFeePerGas 给验证节点的最大小

•maxFeePerGas - 支付交易的最大费用单价(包 括 baseFeePerGas 和 maxPriorityFeePerGas) basefee 表示本区块中的所有交易的最小 gas 单 价, 由以太坊根据前一个区块的 gas 费用计算 得到, 调整方式如下。

区块号	包含的 gas 数	量   fee 变动	base fee
1	15M	0	100gwei
2	30M	0	100gwei
3	30M	12.5	112.5gwei
4	30M	12.5	126.6gwei
5	30M	12.5	142.4gwei
6	30M	12.5	160.2gwei
7	30M	12.5	180.2gwei
8	30M	12.5	202.7gwei
计曾斯	(舌亜)		

-- 个交易的结构如下. 求解其消耗的最多的 gas



(190 + 10) \* 21000 = 4200000gwei = 0.0042eth 发送者减掉 1.0042eth, 接收者账户收到 1eth, 种类 gas 的消耗中 0.00399eth 被烧掉了, 0.00021eth 作为奖励给 miner 节点

这里要注意几点,首先,这里的 gas 的单位都 是 gwei, 其次, 这里的 basefeepergas 是 190gwei

# 以太坊区块链结构

# 以太坊区块链结构



## pow 版本的区块结构

class Block {} //Block 类是一个数据结构,内含 下面这些结构成分:

] BlockHeader header; //块头

] List<Transaction> transactionsList //交易记录 ]List<BlockHeader> uncleList //叔伯块的块头表

class BlockHeader {} //最长可达 800 字节 ] byte[] parentHash; //前导块(父块)的块头 Hash ] byte[] unclesHash;//块身中 uncleList 的 Hash 值 1 byte[] coinbase;

//表示本区块的 Coinbase 和手续费应该给谁. 其 160 位地址。

1 byte[] stateRoot; //执行完本块所含全部交易后 的状态树 Hash 值。

] byte[] txTrieRoot;//块身中所有交易记录的树根 Hash 值。

1 byte[] receiptTrieRoot; //各个交易收据所构成 树根的 Hash 值。

] byte[] difficulty;

l long timestamp;

llong number; // 区块高度

] byte[] gasLimit; //提供的 gas 总和

llong gasUsed; //实际消耗的 gas 总和

]byte[]extraData; //有关本区块的任意额外数据. 不超过32字节。

] byte[] nonce;

以太坊中一个区块能够容纳的 gas 的上限是 30Mgas, 平均出块速度为 15s.

## 以太坊分支解决策略

使用 ghost 算法, 让计算量最大的路径作为主链

## 回执

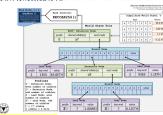
# 回执包含的信息

- 区块编号
- 区块哈希
- · 当前交易用掉的gas
- 当前交易执行后当前区块中使用的累计gas
- 执行当前交易时创建的日志

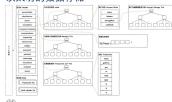
对智能合约来说,外部账户所发起的交易,是 链下世界的输入, 日志是智能合约向外部传递 信息的方式, 当智能合约执行该语句, 就会向 外界产生输出。

## 以太坊 merkel patricia tree 介绍

mpt 树是 merkel 树和前缀树的结合, 更新的时 候, 从根节点依次向下寻找对应的节点, 如果 value 为空,那么删除节点,不然更新节点,然 后反向使用 hash 函数更新 merkel 树节点的值。 以太坊的数据存储: Merkle Patricia Trie



# 以太坊的数据存储



## 以太坊 EVM 虚拟机

所有指令必须是确定行的, 不支持浮点运算。

## 以太坊虚拟机 (EVM)

运行智能合约的环境,运行在每一个节点上,类似于一个独立的沙盒,严格控制了访问权限,合约代码在EVM中运行时,是不能接触网络、文件或者其他进程的。

• 编译合约模块: 主要是对底层Solc编译器进行一层封装,提供RPC接口给外部服

- 务,对用Solidity偏写的智能合约进行偏译。编译后将会返回三进制码和相应的合 约abi、abi可以理解为合约的手册,通过ABI可以知道合约的方法名。参数、返回
- Ledger機块:主要是对区块链账户系统进行修改和更新,账户一共分为两种,分别是普通账户和智能合约账户,调用方如果知道合约账户地址则可以调用该合约,账户的每一次修改都会被持久化到区块链中。
- E/M執行機能(核心模块):主要功能是对交易中的智能合约代码进行解析和执行、每处分的键合约和原用合约所能分。而因为了重原效率。E/M执行模块的文分的键合约和原用合约所能分。而因为了重原效率。E/M执行使用分为编辑信括广进制码直接进行支援令。而订模式会对执行过程中的指令进行优化,如此连接的股份特别是一个划片、方便使用条效执行。

## EVM执行模块的大概流程

- 1. EVM接收到Transaction信息,然后判断 Transaction类型是部署合约还是执行合约, 如果 是部署合约,则新建一个账户来存储合约地址和编译后的代码;如果是执行合约或是调用合约,则使
- 执行上一条指令集之后,判断EVM是否停机,如果 停机则判断是否正常停机,正常停机则更新合约状态到区块链,否则回滚合约状态。如果不停机则继 续执行下一条指令集,重复2;
- 执行完的合约会返回一个执行结果,EVM会将结果存储在Receipt回执中,调用者可以通过Transaction的hash来查询其结果。

# Seeds season seess

OAMPR Turnetion

## pow 算法

## 以太坊PoW算法 ehash



# pos 机制 以太坊PoS共识

## 验证者(validator)以ETH的形式将资本抵押给以太坊上的 一个智能合约。抵押的ETH充当抵押品,如果验证者行为 不诚实或懒惰,可被销毁。然后,验证者负责检查通过网络传播的新块是否有效,并偶尔创建和传播新块。

- 要成为validator, 用户必须将32 ETH存入存款合约并运行 三个软件:执行客户端,共识客户端和验证器。在存入 ETH时,用户加入一个激活队列。一个用户可以发起多个 验证者,目前全球验证者数量40万个。
- 一日激活、验证者就会从以大访网络上的对等方接收新的 区块。重新执行区块中交付的交易,并检查区块签名以确 保区块有效。然后验证者在网络上发送支持该块的投票( 称为证明 attestation)

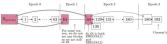
每 12 秒为一个时段 slot, 每 32 个 slot 为一个纪 元 epoch (6.4 分钟),每个 slot 产生一个新区块, 但有可能没有。每个时段 slot 有一组验证者组 成委员会, 其中一个验证者被随机洗中成为区 块的 proposer, 发出区块, 委员会的其他成员投 票 attestation,如 2/3 支持则该区块发布。每个 slot 至少有一个委员会, 成员至少 128 个 validators, 活跃 validator 集被随机选择成员组 成每个 slot 的委员会和 proposer, 一个 validator 可能被同时选为 proposer 和委员会成员。

每个 validator 在每个 epoch 只能属于一个委员 会。通常有超过8192个验证者:这意味着每个 slot 都有一个以上的委员会。所有委员会的规模 相同,至少有128名验证者。当验证者少于4096 个时,安全性会降低,因为委员会成员将少于 128 个。每个 epoch, 验证者被平均分配到各个 slot 中, 然后再细分为适当规模的委员会。洗牌 算法会增加或减少每个 slot 的委员会数量, 以 使每个委员会至少有128个成员。

检查点是 epoch 的第一个 slot 中的一个区块。 如果没有这样的区块,则检查点是前面最近的 区块。每个 epoch 始终有一个检查点区块。 个区块可以是多个纪元的检查点。

当验证者提交 LMD GHOST 投票时, 他投票当 前 epoch 的 checkpoint, 称为 target,•这个投票称

为 CasperFFG 投票,还需要包含前一个 checkpoint, 称为 source. •图中 Epoch 1 的验证 者投票的 source checkpoint 是 genesis 区块 target checkpoint 在 Slot 64 的区块.Epoch 3 中, 验证者投票的 source checkpoint 是 slot 64 区块, target checkpoint 是 180 区块。



Checkpoint 的 justified 和 finalized: 一个 epoch 结束时, 如果它的 checkpoint 获 2/3 验证者投票 支持, 其状态改为 justified。一个 justified 的 checkpoint 如果其后续的 checkpoint 状态改为 justified,则它的状态升级为 finalized。一般情 况下一个 checkpoint 经过 2 个 epoch 后 (12.8 分钟) 变为 finalized EVM

- 存在问题:
  EVM最核心在于安全,而非性能
  EVM或以支持物能外疫运算:
  如思呼学运算,天使部边合约内编码来实现 EVM的数学运算是任效的 无法运算float/double,储存运行"真"随 机算法、HTTP请求 0x8:bn256Pairing 0x9:blake2F
- 长被调用时,除出EVM ,用宿主机来调用高度优化的系统风象

合约部署过程: 用户编写合约, 编译得到的字 节码发送给节点, 节点验证之后创建合约账户, 部署合约, 之后进行广播, 争夺记账权, 交易 被确认。

合约执行过程:建立交易体,签名,将数据发 送给节点, 节点验证之后运行对应函数, 并返 回回执, 最后广播, 争夺记账权, 交易确认。

成员服务: Hyperledger Fabric 支持一个交易网 络, 在这个网络中, 所有参与者都拥有已知的 身份。•公钥基础设施 (PKI) 用于生成与组织、 网络组件以及终端用户或客户端应用程序相关 联的加密证书。可以在更广泛的网络和通道级 别上操纵和管理数据访问控制。•成员服务提供 者 (MSP) 分发证书、验证证书和用户授权背 后的所有加密机制和协议抽象出来。MSP 可以 定义它们身份概念,同样还可以定义管理(身份 验证)和认证(签名生成和验证)这些身份的规则。 principal 实际上是封装在x.509证书中的一个数 字身份和属性.

## 成员服务: CA证书

- · 一个自签名(X.509) CA 证书列表来组成信任根(root of trust)
- 一个X.509证书列表来代表证书验证时需要考虑的中间证书,这些证书应该由某一个信任根据发;中间证书是可选的参数 ー个X.509证书列表,并拥有从某一信任根起可验证的 CA 证书路径。米代表该 MSP的管理是证书,拥有管理员证书别代表拥有申请改变该MSP配置的权力/例如 超 CA 中间产价
- 一个组织单位列表,此列表应出现在该MSP的有效成员的X.509证书中;这是一个可选的配置参数,举例来说,可用于多组织使用相同信任根和中间CA,并给其 一个证书撤销列表(CRLs),其中每一个对应一个列出的(根或中间)MSP CA;这是
- 一个自签(X.509)证书列表,用来组成TLS证书的信任根(TLS root of trust)
- 一个X.509证书列表来代表证书验证时需要考虑的TLS中间证书,这些证书应该由某一个TLS信任根领发;TLS中间证书是可选的参数

区块链服务区块链服务用于维护分布式账本。 区块链服务包括 P2P 协议、分布式账本和共识 机制管理。1.P2P 协议 Fabric 网络中、Peer 和 Orderer 采用 gRPC (Google RPC) 对外提供远程 服务, 供客户端进行调用。网络中的节点之间 通过 Gossip 协议来进行状态同步和分发。2.共 识机制 Fabric 允许根据实际业务需要选择合适 的共识机制,目前支持 SOLO、Kafka、Raft 三 种共识机制。3.分布式账本分布式账本包括两 个组件: 世界状态 (world state) 、交易目志, 分布式账本是世界状态数据库和交易日志历史 记录的组合。世界状态 (world state) 组件记录

的是最新的分布式账本状态, 交易日志组件记 录的是世界状态的更新历史。4.账本存储支持 LevelDB(由 Google 公司研发的键值对嵌入式数 据库管理系统)和 CouchDB (由 Apache 软件基 金会开发的一个面向文档的开源数据库管理系 统).

# 节点

Client: 最终用户, 至少连接的一个 peer 节点或 一个 Orderer 节点, 一般只保存与自己有关的账 户数据

Orderer: 编排节点, 接收包含背书签名的交易 对未打包的交易进行排序生成区块, 并广播给

Peer: 对等节点, 负责通过执行链码 (chaincode) 实现对账本的读写操作, 所有的 Peer 节点都是 提交节点 (Committer) , 负责维护状态数据和 账本的副本

Endorser: 背书节点, 部分 Peer 节点根据背书 策略的设定会执行交易并对结果进行签名背书 充当了背书节点 (Endorser) 的角色。背书节点 是动态的角色,每个链码在实例化的时候都会 设置背书策略, 指定哪些节点对交易背书后才 是有效的。只有在应用程序向节点发起交易背 书请求的时候该 Peer 节点才是背书节点, 否则 它就是普通的记账节点。

## peer 节点分类

提交节点 (Committer),每个Peer 节点都是-个提交节点。他们会接收生成的区块,在这些 区块被验证之后会以附加的方式提交到 Peer 节点的账本副本中。

背书节点 (Endorser) ,每个安装了智能合约执 行引擎的 Peer 节点都可以作为一个背书节点。 然而, 想要成为一个真正的背书节点, 节点上 的智能合约必须要被客户端应用使用, 来生成 一个被签名的交易响应。

主节点。当组织在通道中具有多个 Peer 节点的 时候,会有一个主节点,它负责将交易从排序 节点分发到该组织中其他的提交节点。

锚节点。如果一个 Peer 节点需要同另一个组织 的 Peer 节点通信的话, 它可以使用对方组织通 道配置中定义的锚节点。一个组织可以拥有 0 个或者多个锚节点、并且一个锚节点能够帮助 很多不同的跨组织间的通信。

## 区块头的结构

编号+交易 hash 值+前一个区块头的 hash 值 交易验证流程

# 智能合约提取一组名为交易提案的输入参数,

并将其与程序逻辑结合起来使用以读写账本。 对世界状态的更改被捕获为交易提案响应(或 简称交易响应),该响应包含一个读写集,其 中既含有已读取的状态, 也含有还未书写的新 状态 (如果交易有效的话) 。注意, 在执行智 能合约时世界状态没有更新! • 一项交易被分发 给网络中的所有节点, 各节点通过两个阶段对 其进行验证。首先,根据背书策略检查交易, 确保该交易已被足够的组织签署。其次、继续 检查交易,以确保当该交易在受到背书节点签 名时它的交易读集与世界状态的当前值匹配, 并且中间过程中没有被更新。如果一个交易通 过了这两个测试、它就被标记为有效。所有交 易,不管是有效的还是无效的,都会被添加到

界状态。 部署 chaincode

打包链码: 可以由一个组织完成, 也可以由每 个组织完成。·在 peer 节点上安装链码:每个将 使用链码来背书交易或查询账本的组织都需要 完成此步骤。•批准组织的链码定义:每个将使

区块链历史中, 但是仅有效的交易才会更新世

用链码的组织都需要完成此步骤。链码定义需 要得到足够数量的组织的批准, 以满足通道的 生命周期终止策略 (默认情况下为大多数), 然后才能在通道上启动链码。•将链码定义提交 到通道:一旦通道上所需数量的组织获得批准, 提交事务需要由一个组织提交。提交者首先从 已批准的组织中足够多的对等方收集背书, 然 后提交事务以提交链码定义

## 交易流程

客户端应用程序通过通道内一个可信的 peer 节 点上的 Fabric Gateway 服务把交易提案发送给 该节点,该节点执行交易提案或转给其他 peer 节点去执行, 并把交易提案发给一组节点去背 书。•根据交易所调用的链码背书策略、相关的 背书节点对该交易提案进行背书签名。•背书节 点不将提案中的更新应用于其账本副本。相反, 背书节点将向客户端应用程序返回一个提案响 应。•已背书的交易提案最终将在第二阶段经过 排序生成区块,然后在第三阶段分发给所有节 点进行最终验证和提交。

交易排序并打包成区块

排序节点将区块分发给连接到它的所有 Peer 节点开始。并不是每个 Peer 节点都需要连接到 一个排序节点, Peer 节点可以使用 gossip 协议 将区块关联到其他节点。 共识算法

## 容错共识对比

出错当古 出错节占 。 崩溃节点 。 崩溃节点 。 恶意节点 节点总数2n+1时,出错节点数 · 节占总数3n+1时, 出错节占数 不超过n 不超过n 实现: Raft、Kafk · 算法: PBFT、Hotstuff

Raft: Fabric Raft 排序节点工作流程 1.交易自动 路由到通道的当前主导 orderer 节点 2. 主导 orderer 节点验证交易后, 将收到的交易传入区 块切割模块, 创建候选区块 3.产生新的区块, 主导 orderer 节点将其应用于本地的 Raft 有限状 态机 (FSM) 4.有限状态机将尝试复制到足够数 量的 orderer 节点,以便提交区块 5.区块被写入 接收节点的本地账本

Pbft:Quorum 组成 (3f+1 个节点) 。Client。Replica。 每个共识阶段称为一个 view--个 primary 其他 replica 都是 backup•共识过程。Request: client 发 起共识请求 pre-prepare: primary 验证请求、打 包消息、发给其他节点。Prepare:其他节点验证 消息、投票。Commit: primary 收到 2f+1 个节点 赞成票后,完成共识,通知所有节点,反馈 client

- 1. 分布式算法假设: 数据包可信, 节点独立运 行使用网络通信, 可能有恶意节点, 信道可靠 2. eth 中的合约部署交易, storageroot 存储部署 的合约的地址,codehash 存储代码的 hash 值
- 3. fabric 性能方面: 比特币和以太坊的性能依 赖于 miner, 但是 fabric 的性能不仅仅依赖于 order 节点, 也以来 peer 节点, 效率高: 打包的 时候不需要检查; 共识在此前已经形成; chaincode 的执行可以并行化
- 4. 零知识证明: 零知识证明: 证明者能够在不 向验证者提供任何有用的信息的情况下, 使验 证者相信某个论断是正确的
- 5. 同态加密: 对经过同态加密的数据进行处理 得到一个输出,将这一输出进行解密,其结果 与用同一方法处理未加密的原始数据得到的输 出结果是一样的