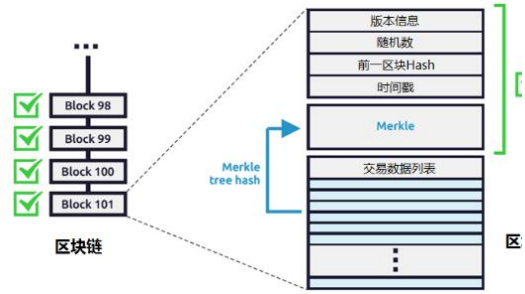
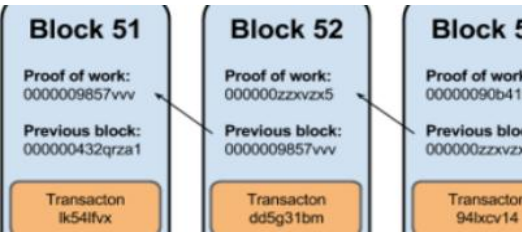
1. 区块链

以链式区块组织账本数据实现账本数据的不可篡改&分布式的可信记账机制 共识机制，新型存储，制衡合约 链链互联 隐私保护

基于哈希值连接，区块链中的数据无法删除，区块链越长可信度越高(一笔交易之后 6 则可信)，每个区块包括区块头和交易数据，Merkle 树用于对交易数据列表进行快速寻址。和区块哈希值（不包含在区块内）不同，区块高度并不是唯一的标识符，因为可能区块链分叉。





区块的结构

大小	字段	描述
4字节	区块大小	用字节表示的该字段之后的区
80字节	区块头	组成区块头的几个字段
1-9 字节（可变整数）	交易计数器	交易的数量
可变的	交易	记录在区块里的交易信息

区块头的结构

大小	字段	描述
4字节	版本	版本号，用于跟踪软件/协议的更新
32字节	父区块哈希值	引用区块链中父区块的哈希值
32字节	Merkle根	该区块中交易的merkle树根的哈希值
4字节	时间戳	该区块产生的近似时间（精确到秒的Unix
4字节	难度目标	该区块工作量证明算法的难度目标
4字节	Nonce	用于工作量证明算法的计数器

Version	PrevBlock	MerkleRoot	Time	Bits	Nonc

Merkle 树:哈希两次。归纳区块中所有交易，生成整个交易集合数字指纹，提供了校验区块是否存在某交易的途径（Merkle Path Proof），至多 log2(N)次即可找出。叶节点需要偶数->复制末一个。比较，定位修改。全节点（矿工/非矿工），轻节点（SPV 支付验证'只存储头'/钱包'一个连接区块链的应用软件不存储账本'）

分布式系统：拜占庭故障包括崩溃故障，崩溃容错不容忍有恶意节点，拜占庭：独立 可通信 有恶意 信道可靠但可中断 要达成一致性（忠诚则目标一致）和正确性（少数坏不影响）PBFT RAFT

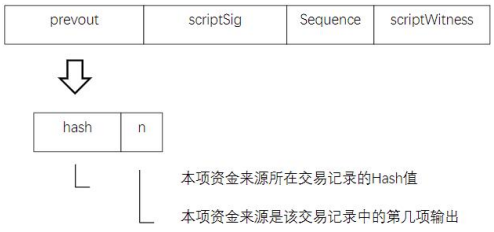
POW 共识：工作量证明 proof of work：高强度哈希(SHA256)进行算力竞争解决记账权，达成共识，在开放系统，动态增减，海量节点的场景下，可以解决记账权的问题。胜出的节点建立区块并广播，其他校验新区块，根据父区块 hash 去找，链到主链/分叉/独立区块。

挖矿：挖矿节点必须有钱包的功能（有自己 160 位密码地址，私钥），打包成区块的时候额外加一个交易 coinbase，utxo 招领地址写自己，记账成功的话则成功拿到。

UTXO：每个比特币用户用非对称密钥和 sha160 产生的地址（比特币网络交易唯一标识），私钥用于数字签名。Unspent Transaction Output 每个地址的资金余额就是散布在账本中所有 UTXO 的总和，用时把自己名下的 UTXO 作为交易输入。

CTxIn 认领脚本 scriptSig

P2PK PaytoPublicKey P2PKH Pay to Public KeyHash P2SH PaytoScriptHash



CTxOut 招领脚本 scriptPubKey

Value（支付金额）	scriptPubKey（招领脚本，即领用条件）

比特币虚拟机：运行以上俩脚本，并且具有主要验证签名是否正确的指令集。

比 tb 区 kl 网络通信（P2P）：种子结点 伙伴列表 发现伙伴

区块链技术是一种以非对称加密技术对交易进行数字签名，通过工作量证明等共识机制进行记账节点协调，数据以链式区块形式组织存储的分布式账本技术。

优：可信永久去中心平等性 **缺**：确认时间长（所以可以在特定情况采用链外的支付通道以及闪电网络）链上数据容量有限 区块链分叉以及攻击 底层协议可能存在缺陷 智能合约可能存在漏洞 信息匿名性低 不同链难互通 开发社区生态（抄来抄去）链上链下数据协同

区 kl 形态：无许可/许可，联盟链/公/私

2. 以太坊

是一个平台和一种编程语言，使开发人员能够建立和发布下一代分布式应用以太坊有自己的加密货币 Ether，用于支付以太坊网络上的某些活动

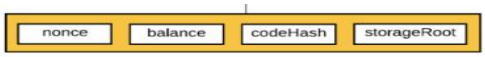
智能合约：智能合约是内嵌于区块中的自定义程序逻辑，当满足一定条件，区块中的程序逻辑会被触发执行合同约定的指令（比如资产清算、赔偿、交割等）

超级账本：是一个为了提高跨行业的区块链技术的开源合作项目（hyperledger）

数字货币钱包：“钱包”是指用于存储和管理用户密钥的容器。例子有全节点，spv 轻钱包，中心化钱包（依赖自身中心化服务器）冷钱包 硬件钱包

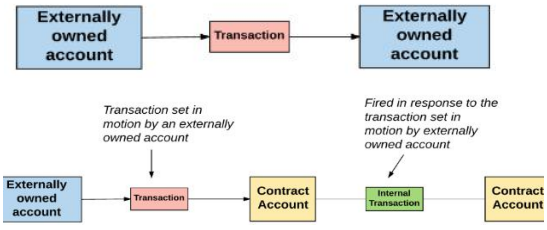
以太坊的组成：通过 P2P（use gossip to find 节点） 网络通信，交易（转账交易，合约交易），状态机（以太坊状态转移由 EVM 处理（以太坊虚拟机，处理字节码，solidity 可以转化为 bytecode））区块链账本，共识算法（由 pow 转为之后的 pos）客户端，【分布式应用软件（Dapp）作为应用层】

节点和账户：以太坊由大量的节点组成，节点有账户与之对应，节点每一个都运行着以太坊虚拟机，节点之间使用共识机制保证可靠性。外部账户（区块链外部主体创建，由公私钥对控制，没有合约代码）和合约账户（合约交易创建，唯一标识的智能合约，地址就是代码哈希）除了以上存储的不同，两种账户都存储以下 4 个：



区别：是否有成本（外部无，合约有 因为要使用网络存储）外部可以启动交易，合约只能收到交易做出相应（相应可以发起交易）两个外部账户交易只能是以太币转账，合约账户由智能合约代码控制

外部账户可以给其他外部账户发送消息（等同于转账）也可以向合约账户发起合约交易(可以触发合约代码)，



世界状态：所有账户的状态合集，以太坊本质上就是一个基于交易的状态机，以太坊的初始状态我们成为 Genesis，状态转换最小单元是交易。

以太坊交易举例：简单支付交易(两外部账户直接发消息)存证交易(把转账余额变为 0 即可，并加上需要存证的数据)合约部署交易(把对方地址写为 0)合约调用交易(复杂支付交易(有关 token)查询交易

Gas：在以太坊上，任何引起状态转移的操作都要消耗 gas，余额不够则回退操作，但是不归还 gas，因为之前的操作确实完成过，消耗了资源。

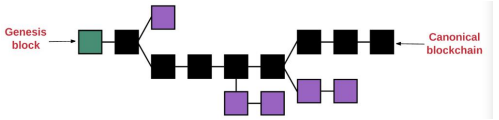
```
class BlockHeader {}    //最长可达800字节
] byte[] parentHash;          //前导块（父块）的块头Hash值
] byte[] unclesHash;          //块身中uncleList的Hash值
] byte[] coinbase;            //表示本区块的Coinbase和手续费应该给谁，共160位地址。
] byte[] stateRoot;           //执行完本块所含全部交易后的状态树Hash值。
] byte[] txTrieRoot;  //块身中所有交易记录的树根Hash值。
] byte[] receiptTrieRoot; //各个交易收据所构成树根的Hash值。
] byte[] difficulty;  //挖矿难度，用以调整发块周期长度
] long timestamp;     //时戳
] long number;        //本块在区块链中的高度，即区块链中处于本块之前的区块个数。
] byte[] gasLimit;    //本块所含所有交易所提供“汽油”量即手续费的上限总和。
] long gasUsed;       //本块所含所有交易实际消耗“汽油”量的总和。
] byte[] extraData;   //有关本区块的任意额外数据，不超过32字节。
] byte[] nonce;       //
```

- 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费单价
- maxFeePerGas - 支付交易的最大费用单价（包括 baseFeePerGas 和 maxPriorityFeePerGas)

GasPrice 表示发送者预付的 gas 价格，Fee=Gas*GasPrice。不仅计算需要，存储也需要付费。

以太坊（pow）平均出块时间是 15s

以太坊区块链应对分叉：GHOST 一挑选计算量最大的那条路径



交易后会有一个回执包含交易信息，用掉多少 gas 什么的还包括日志，日志是链上世界对链下世界的信息传递，可以在智能合约定义 event，执行 event 就有日志。

Merkle Patricia Trie：更新的时候先找到，在从下向上更新，就是字典序，有点像那个霍夫曼树，越往下就是把前面节点的字母加起来

以太坊的网络：主网 测试网

以太坊的虚拟机：EVM：基于栈的解释器，执行账户合约字节码，所有指令运算必须是确定性的，高精度浮点运算不可以！

图灵完备：数据操作规则可以是一门编程语言，也可以是计算机里具体实现的指令集。当这套规则可以实现图灵机模型里的全部功能时，就称它具有图灵完备。支持条件语句和循环语句，但是为了防止出现图灵停机问题，我们引入 gas 防止死循环

EVM 处理大概流程：接受交易信息，判断是部署还是执行，执行/部署完，判断 EVM 是否正常停机，否则回滚。如果是正常，则回执。

以太坊的共识机制：POW->POS

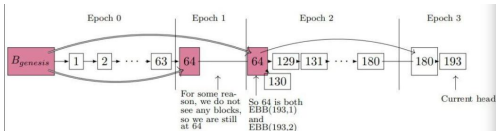
(POS)网络中的用户通过质押一定数量的以太坊成为验证者。每次系统从这些验证者从随机选择出区块创建者，其余验证者去验证创建出的区块是否合法。验证者会获得出块奖励，没有被选中的区块不进行验证则会被扣除一定质押币，如果进行错误验证则会被扣除全部质押币,权益证明在每隔一定区块的地方设置一个检查点，对前面的区块进行验证，2/3 验证者通过则验证通过，验证通过则该区块所在链成为最长合法链(不能被回滚)。

以太坊信标链：Beacon Chain： 每 12 秒为一个时段 slot，每 32 个 slot 为一个纪元 epoch，每个 slot 产生一个新区块，但有可能没有，信标链的 Genesis 区块在 slot 0。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选中成为区块的 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。

检查点：checkpoint：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。一个区块可以是多个纪元的检查点。

当投票时，当前 epoch 的 checkpoint 称为



targetcheckpoint，前一个 checkpoint，称为 sourcechp，，

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.

Attestation 投票接纳过程：generation,(传播)propagation，(聚合)aggregation，propagation，inclusion 接纳

针对 THE DAO:要求矿工彻底解除盗窃并且归还 TheDAO 所有以太币，这样就能自动归还给代币持有人，从而结束该项目

3. Fabric

联盟区块链、许可区块链，是一个面向企业应用的区块链软件，不是公链，不发行加密货币

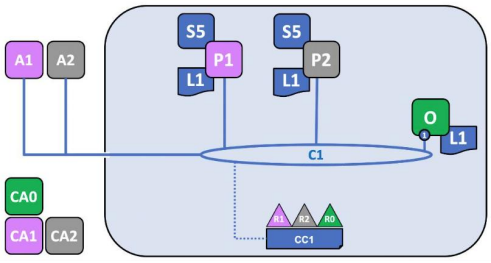
服务：成员服务（注册，审计）区块链服务（节点共识管理，账本分布式计算，账本存储，p2p 协议）合约代码服务（提供智能合约服务）

CA 体系：(公钥基础设施)PKI 由向各方(如服务的用户及提供者）发布数字证书的证书授权中心组成，各方使用证书在与其环境交换的消息中对自己进行身份验证：Fabric 有一个独立的 Fabric CA 模块，用来管理证书服务;CA 证书:一个自签名 X.509CA 证书列表来组成信任根(root of trust)，Fabric CA 提供客户端和 SDK 两种方式与 CA 进行交互，每个 Fabric CA 都有一个根 CA 或者中间 CA。为了保证 CA 的安全性，中间 CA 可以采用集群方式搭建

Fabric 的区块链服务：用于维护分布式账本，P2P[gRPC+Gossip]，共识机制[solo kafka,raft].分布式账本[包含世界状态和交易日志]，账本存储[数据库相关]

节点:client[至少连一个 peer 或者 orderer]，orderer 编排节点[接受包含背书签名的交易，排序打包并广播给 peer]peer 对等节点[执行链码实现对账本的读写，所有的 peer 都是 commiter 提交节点负责维护数据和账本副本]endorser 背书节点[只有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 节点可以担任多个角色>

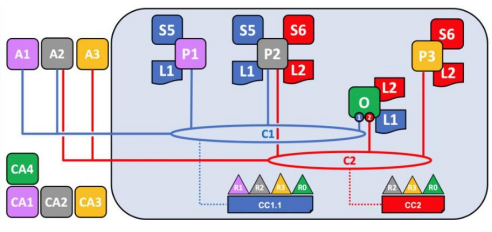
角色 1:提交节点:接收生成的区块,提交到 Peer 节点账本副本中;角色 2:背书节点:有一定前提，需要被指定才可以，智能合约的背书策略明确了在交易被接受并且记录到提交节点的账本之前，需要哪些组织的 Peer 节点为交易签名。



Fabric 区块链网络：

多个组织聚集在一起形成一个通道（channel），在该通道上，事务在链码上调用，并且权限由最初配置通道时商定的一组策略确定。

Channel:是构建在 Fabric 区块链网络上的由若干个节点所组成的一条链，实现了数据的私有和保密一个 Peer 节点至少接入一个通道，可以接入多个通道。每个通道拥有自己单独的账本，且仅对通道成员节点共享,Orderer 节点拥有的通道成为系统通道，该通道账本拥有网络上所有账本数据。



peer 的其他两种角色:主节点(负责将排好序的交易分发到其他提交节点)锚节点(负责不同组织的 peer 通信，可以有多个)

fabric 之内的全部交易都是合约交易，因为没有虚拟币，所以不牵扯转账

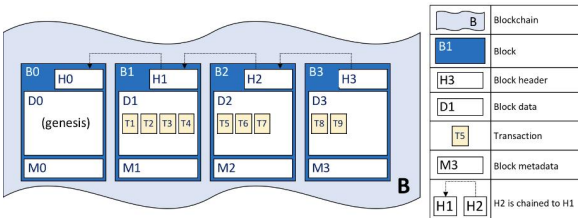
Fabric 账本:[世界状态(数据库/记录当前状态，可篡改)+区块链(交易日志/不可篡改)]

世界状态:key+value(key 唯一，value 随意大小和内容)

区块链结构:区块头，区块数据，区块元数据(与区块同时产生，所以不包含在 *hash* 之中)

区块头结构：区块编号，当前区块的哈希值(当前区块中包含的区块数据(交易)的哈希值)，前一个区块头的哈希值 current block data hash->current block hash ;previous black hash->copy of hash from previous block

交易的结构：头 H,签名 S，提案 P(是对 chaincode 的调用),相应 R，



背书 E，所有的交易都要调用 chaincode 因为 fabric 不是公链，不存在转账交易.

当 client 发起交易的时候,只填写了一个 H S P,传到 peer 节点之后，运行 chaincode,保存 h 保存 s p 加一个 r 填入执行结果,然后 peer 再发出去 发到 organization 检查去背书,不合格的话交易失败 但这个包还是会还给 client,就算是失效的也要打包放在区块链里（有效则更新世界状态），因为这就是设计理念（bitmap)然后签名，在 e 字段加上自己的签名 之后交易顺利完成 全部填满了。这个交易此时只是有效，还没有进入区块链这个交易回到 client 再发到区块链里 有orderer 进行排序 之后打包放在区块链即完成。(执行智能合约时世界状态未更新）

链码：链上代码是宿主机在容器之中运行的代码，图灵完备，防止死循环引入了计时器，一个链码可以包含多个智能合约。

背书：每个链码都有一个背书策略与之相关联，该背书策略适用于此链码中定义的所有智能合约。交易必须由 Fabric 网络中受信任的组织验证。之后才称之为**有效。**

交易验证：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。检验两个东西：1.是否根据背书策略得到足够背书 2.是否签署之前与当前未更新的世界状态匹配。

链码的部署：打包链码---在 peer 上安装链码---批准组织的链码定义===将链码定义提交到通道

链码打包：链码要打包成一个 tar 文件，可以用 Node Fabric SDK，或第三方工具如 GNU tar。可以由通道内一个组织或每个组织

链码安装：在每个将要执行链码的 peer 节点上安装链码包

批准链码定义：每个要使用链码的通道成员都需要批准其组织的链码定义，批准由 peer administrator 发起，通过后提交给 ordering 服务，然后分发给所有 peer 节点。

提交链码定义：链码定义的提交 transaction 由 peer administrator 发起，经通道中的 peer 节点背书通过后提交给 ordering 服务，然后把该链码的定义提交给通道。

Fabric 共识：当区块中交易的顺序和结果满足明确的策略标准检查时，最终会达成共识。

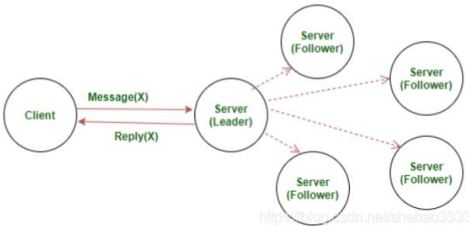
Fabric 共识由①提案和背书②排序③验证和提交组成，Fabric 的设计依赖于确定性的共识算法，账本不会像其他分布式的以及无需许可的区块链中那样产生分叉。

fabric 是许可链！！！以太坊部署合约一个节点就够了 只要你愿意付这个部署费和存储费用就行，而 fabric 更加复杂 打包和安装在自己控制的节点就能做好，但是后两步就不行了，因为要大家（将要使用这个 chaincode 的人）都认同，所以流程很严格

1. 提案和背书：首先客户端连接到一个 peer 节点，引用一个链码（提交一个 proposal）之后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。
2. 交易排序并打包成区块：虽然 Peer 节点执行智能合约并处理交易，而排序节点不会这样做。到达排序节点的每个授权交易都被机械地打包在一个区块中，排序节点不判断交易的内容。
3. 验证和提交：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。之后提交到账本上。

三种排序：solo kafka raft（raft 是分布式故障崩溃容错共识算法）

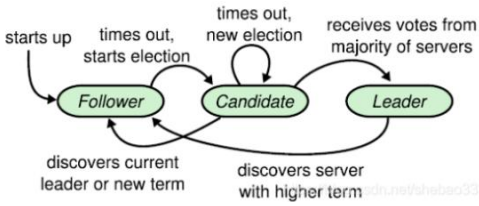
拜占庭容错：崩溃节点和恶意节点，节点数 3n+1，出错的不超过 n，崩溃容错：只有崩溃节点，不包括恶意，节点 2n+1，出错最多不超过 n。



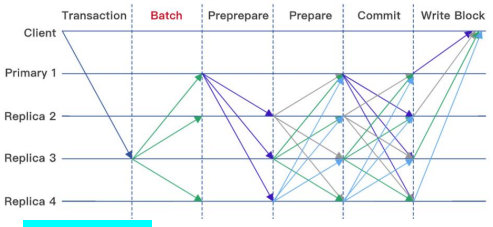
Raft 协议：是一个管理复制日志的共识算法，复制日志是复制状态机的组成部分，

客户端给主导节点发送,主导节点追加日志并给其他 follower node 更新，其他返回确认消息，主导节点收到一定数量的确认消息之后，主导提交日志，跟随节点也提交日志。

以下三个状态，用心跳包不断检测 leader 的状态，不对的话就选举。



以下是 pbtf（3f+1）共识，一个 primary 其余全是 replica-backup Primary 接受到 2f+1 投票即可完成共识，通知 all 节点，反馈 client



4. 区块链发展

共识机制比较以及新型共识机制：pow 竞争性哈希计算确定记账|不可逆|电能 成本 速度慢； pos 根据资产的多寡调整获得记账权的概率|不可逆 速度快 低耗能|不公平 寡头优势；Dpos 选中一小群节点作为代表进行 pos 速度更快 更民主|没有考虑账户的重要性。以下为新时代共识机制

共识机制	适用场景	性能效率	资源消耗	容错率
POS-POW混合共识	公链	低	中	50%
授权拜占庭容错（DBFT共识）	公链/联盟链	高	低	33%
BFT-DPOS混合共识	公链/联盟链	高	低	33%
VRF共识之Algorand算法	公链/联盟链	高	低	33%

隐私保护：区块链交易具有匿名特性，在一定程度上保护了个人的隐私，但是还是存在泄露可能（交易全公开），所以提出以下 2 途径：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。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。

侧链和跨链：联盟链：Fabric 的跨链技术（借助服务节点）公有链则采用侧链。多链和多通道：Fabric 定义了链、Peer、通道、共识服务的概念，

分片技术：解决扩展性问题，提高网络吞吐量

雷电闪电网络，链外计算 都是为了提高速度并且不损失安全性

区块链的应用：数字货币，libra，数字票据，证券交易，保险，供应链金融，资产通政化，供应链金融，跨境转账和支付，版权保护，法律信息，信息溯源