全球区块链生态技术大会

Global Blockchain Eco-Technology Conference

区块链核心技术横向剖析

谢文杰

智链ChainNova CTO

主办方 Geekbang. 极客邦科技 InfoQ

区块链前哨
BLOCKCHAIN OUTPOST

正本清源，打造链圈
第一技术公众号

掌握前沿区块链资讯
深度分析区块链技术
致力于区块链技术普及

扫码关注区块链前哨

主讲人简介

谢文杰

智链ChainNova CTO，联合创始人

原百度公有云高级产品经理，原金山云产品专家，原百度移动事业部技术经理
IBM外部技术Champion，CSDN区块链金牌讲师

主持过多个大型系统设计，从零构建金山云云计算产品，搜狐WebIM亿级PV技术产品创造者。

01.

区块链技术基石

不同区块链平台都依赖的几个基础技术

2 智能合约及数据模型

智能合约的发展历史及主流的两类数据模型

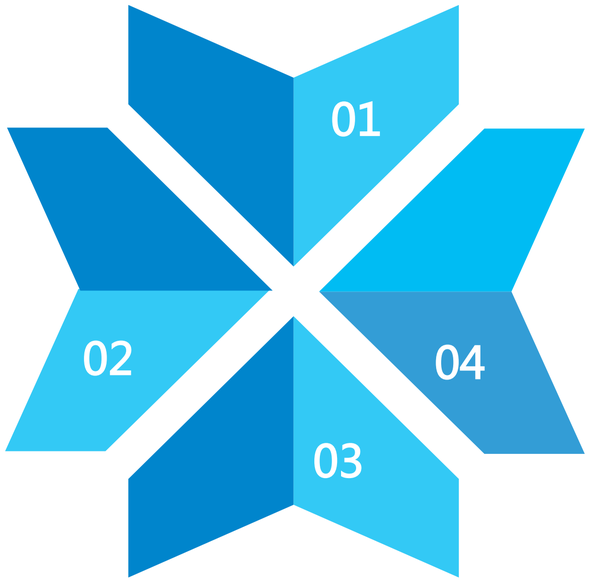
3 经典项目概览

经典项目彼此技术上的独特性

01 基础技术概览

Hash和非对称加密等一系列密码学算法
是区块链技术体系最基础的构成

密码学算法



Merkle Tree

Merkle Tree及其一系列衍生改进是区块链
数据结构的一大基础构成

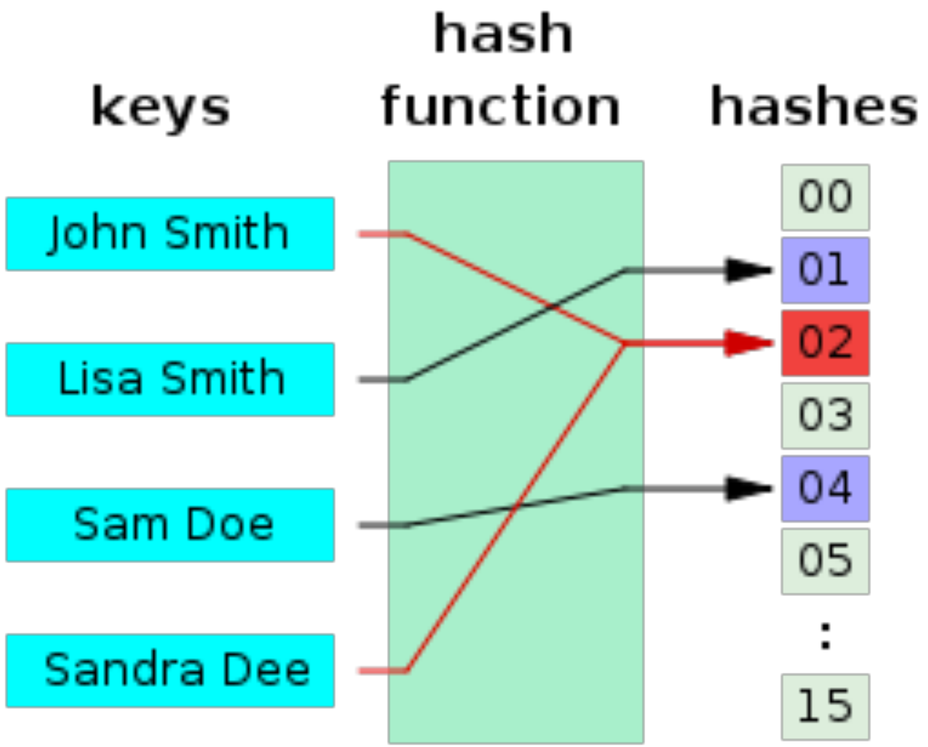
PoX

区块链保证不可篡改的核心即共识算法，简称
PoX（Proof of something）

链式快照

Blockchain即Block+Chain，Chain即区块
链的链式快照，链式快照的数据模型是区块
链数据结构的另一大基础构成

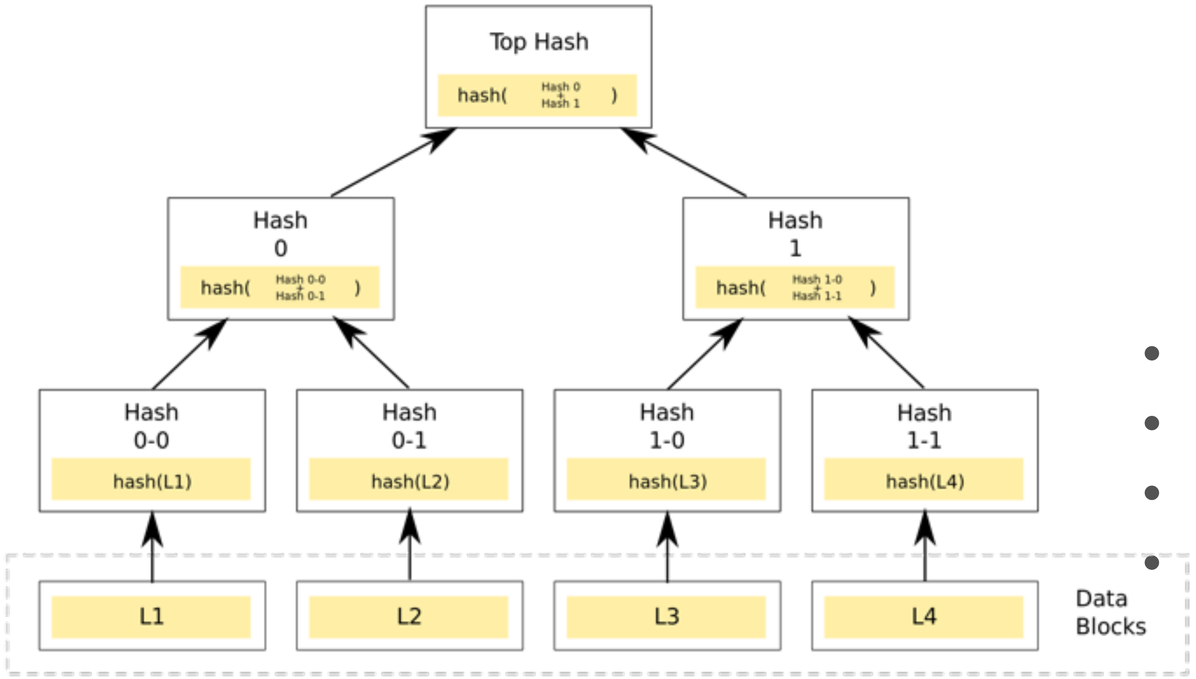
01/1 Hash算法



https://en.wikipedia.org/wiki/Hash_function

- 区块链第一课，基础的基础
- 哈希又称散列算法，它是一种数据映射关系
- 任意长的数据经过哈希运算后，得到的是一个固定长度的数据
- 特点：确定性、均匀性、不可逆……
- 应用场景：快速查找、重复检查、数据校验、数字签名……

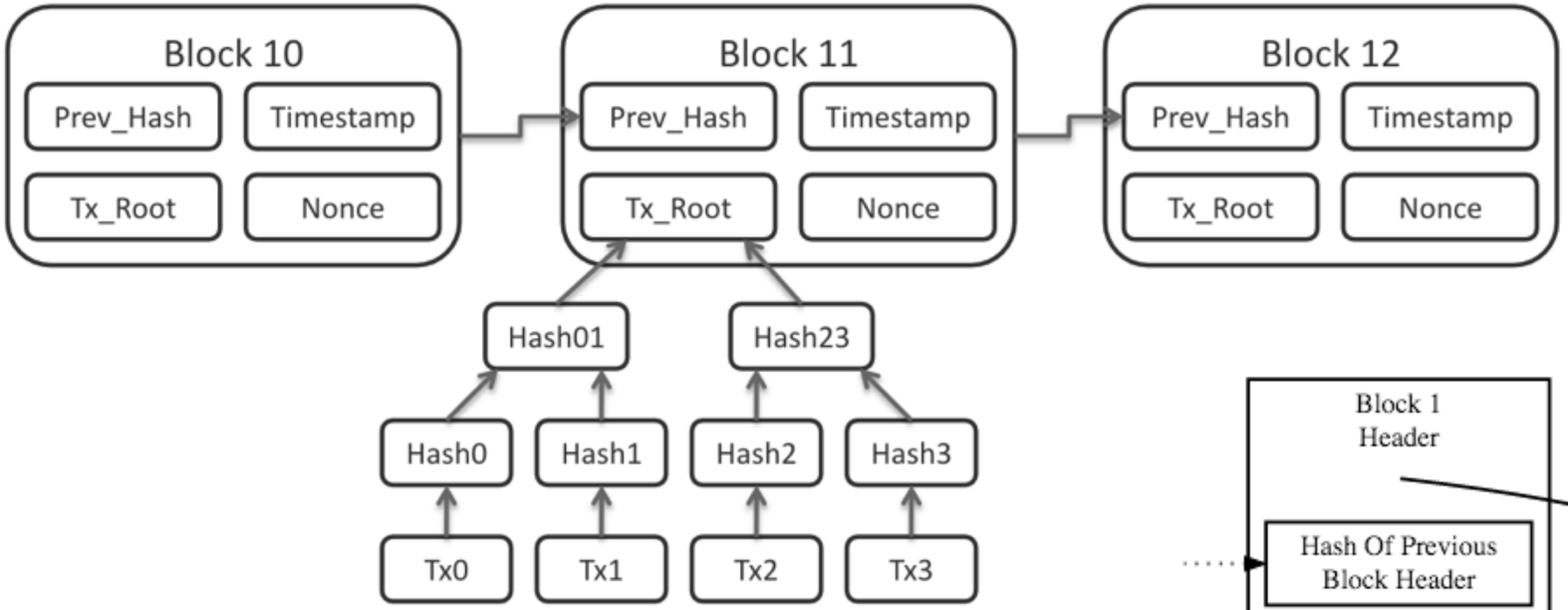
01/2 Merkel Tree



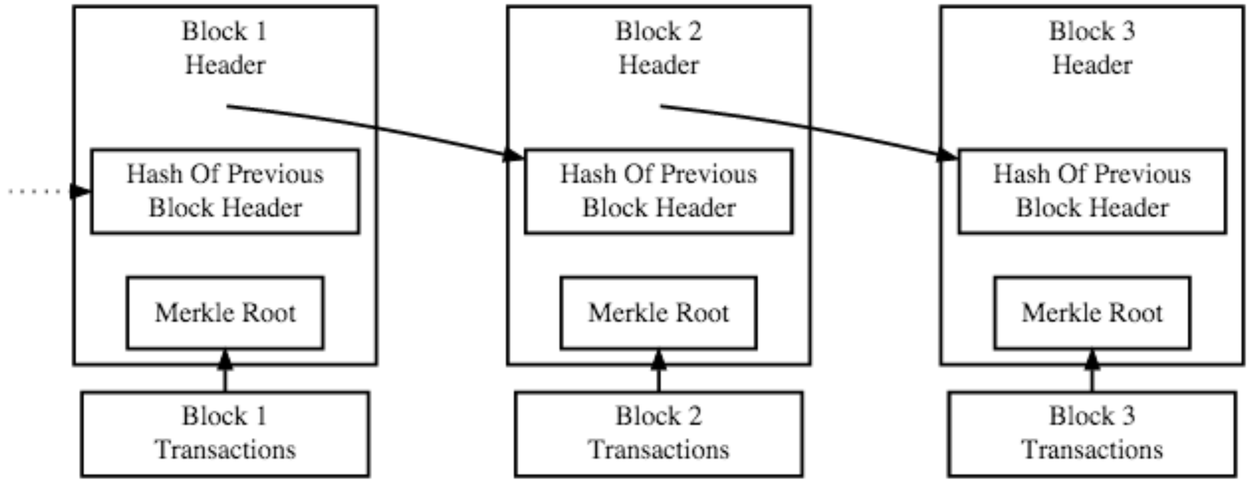
- 区块链数据结构基础之一
- 以发明人Ralph Merkle命名，1979年
- 快速比对数据、快速定位变化……
- 应用场景：Blockchain、Git、BitTorrent、Cassandra、Dynamo、ZFS、btrfs……

https://en.wikipedia.org/wiki/Merkle_tree

Merkel Tree在bitcoin里的应用



https://en.wikipedia.org/wiki/Blockchain

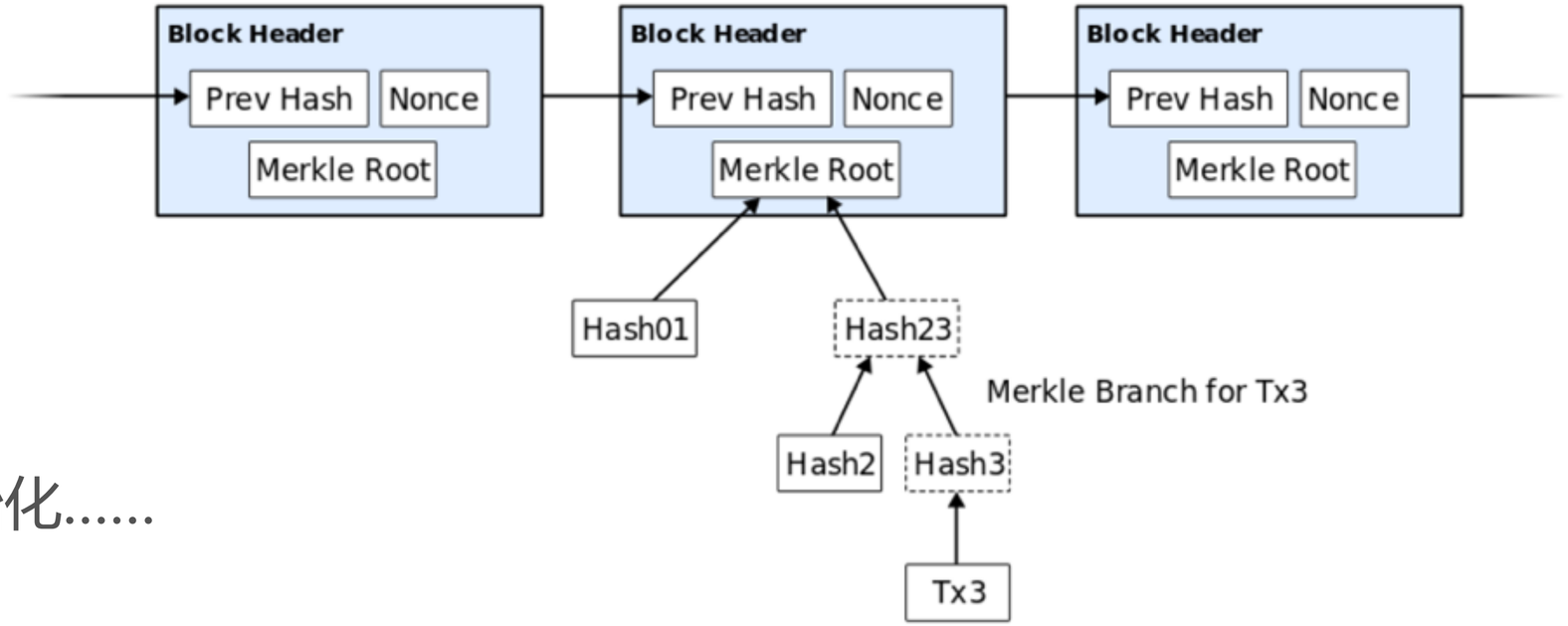


Simplified Bitcoin Block Chain

https://bitcoin.org/en/developer-guide#block-chain-overview

- 交易数据即以此构建

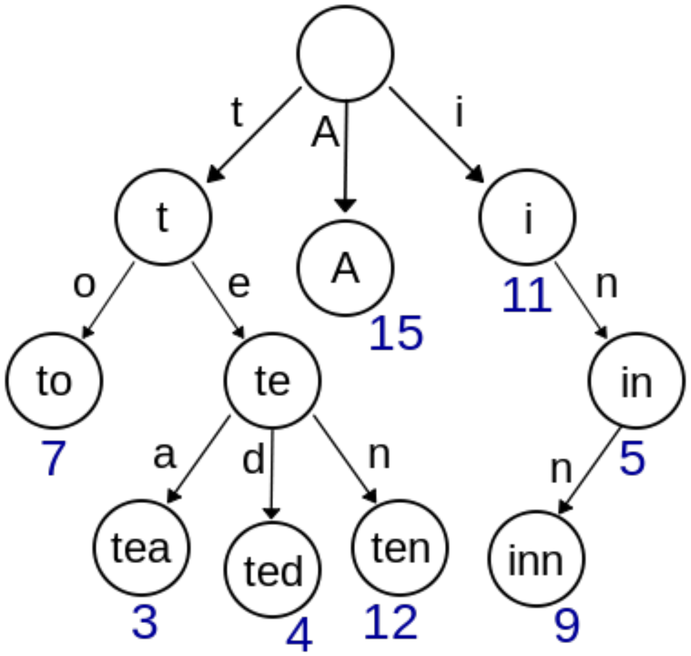
应用收益：SPV（轻量化验证）



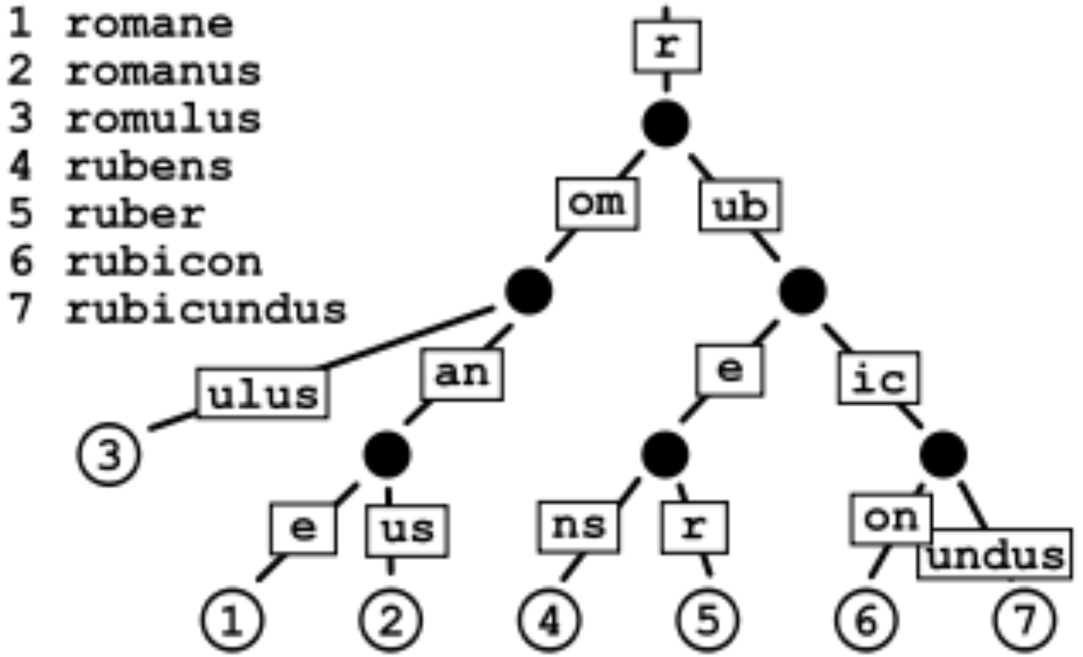
- 快速验证，无需完整数据
- 轻量节点：移动端、快速初始化……

https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification

Trie以及Radix Tree



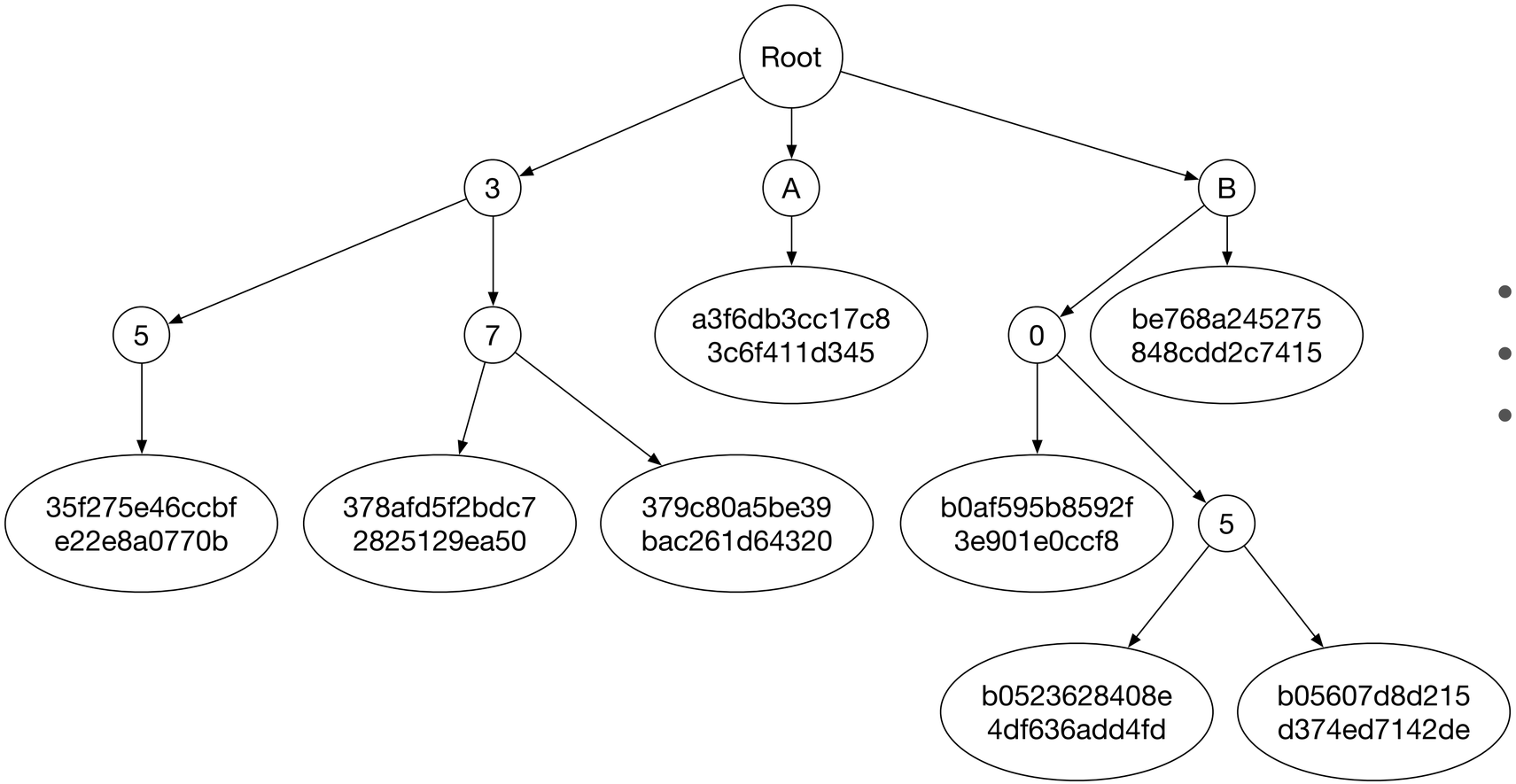
https://en.wikipedia.org/wiki/Trie



https://en.wikipedia.org/wiki/Radix_tree

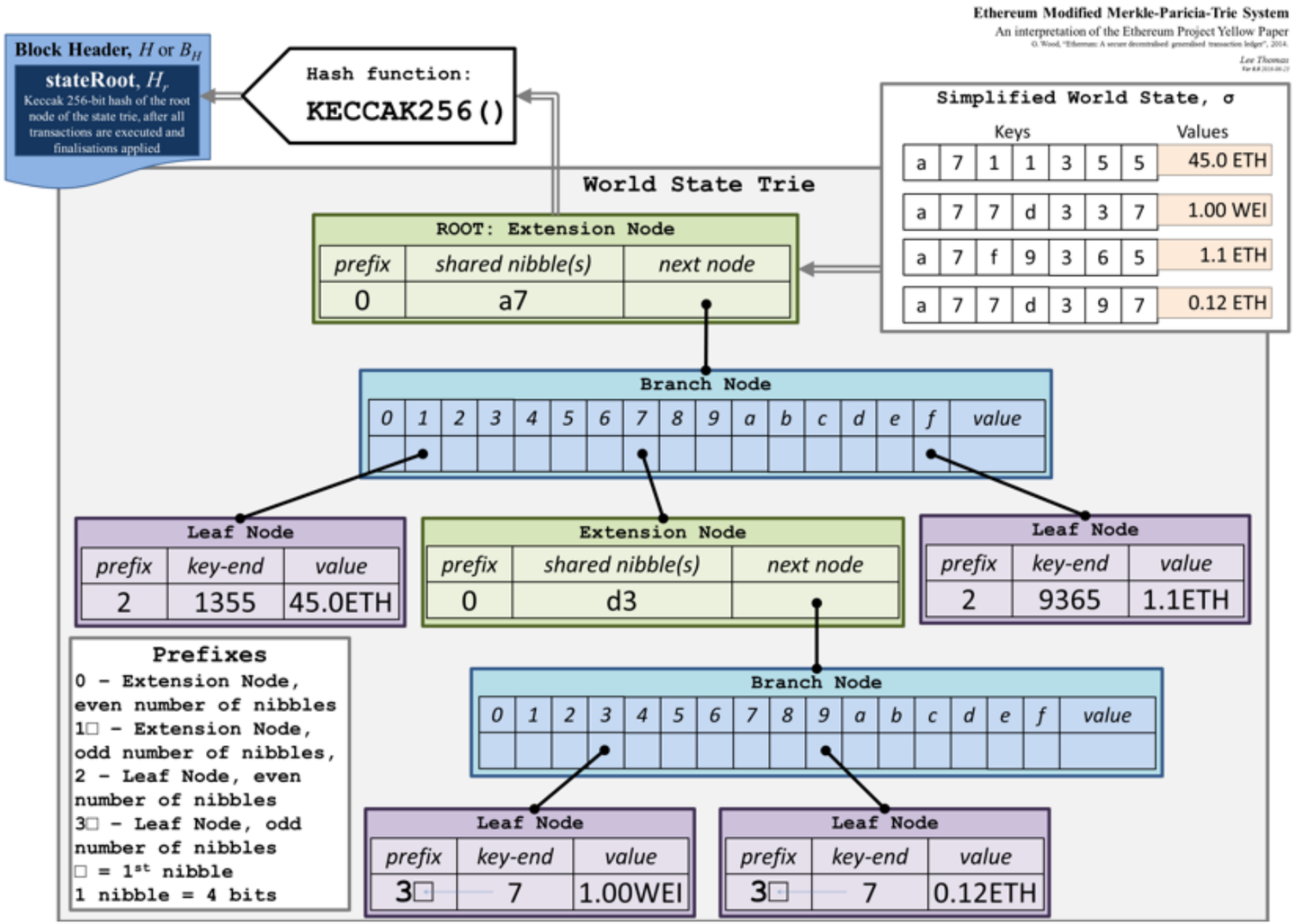
- 提高查询效率，减轻存储压力
- Merkle Tree也对应有相应的改进及应用

Ripple应用: SHAMap



- Trie+Merkle Tree
- 优势：快速定位数据
- 用于存储账户余额、挂单信息等

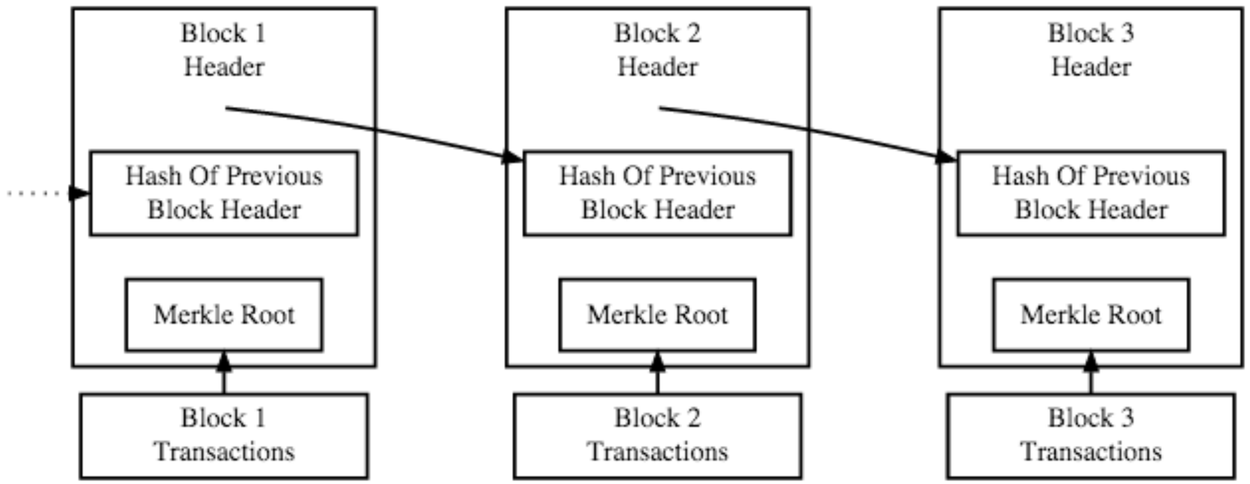
01 Ethereum应用: Merkle Patricia Tree



- Radix Tree+Merkle Tree
- 存储效率更高

https://ethereum.stackexchange.com/questions/6415/eli5-how-does-a-merkle-patricia-trie-tree-work

01 Block+Chain



Simplified Bitcoin Block Chain

https://bitcoin.org/en/developer-guide#block-chain-overview

- 区块链数据结构基础之二
- 链式快照，快照中存储数据（比特币：交易、时间、随机数……）
- 当前Block包含前一个Block的hash形成链
- 最新Block的Hash即可校验所有历史数据的任一点篡改

PoX：Proof of something

PoW	PoS	DPoS	BFTs	CFT
•参与者竞争计算，先算出者获得记账权及奖励 •俗称挖矿 •公平 •但效率低 •能耗巨大	•根据权益比例概率性获得记账权及奖励 •节能 •但依赖权益设计	•权益持有者通过投票选举领导者代理记账权及奖励 •高效 •但有中心化倾向	•联盟链常用，预选定节点记账，算法容忍一定的恶意节点 •高效 •但预选定节点规模有限 •攻击容忍度低	•分布式系统常用，通常为选举模式，算法容忍一定的故障节点 •高效 •但恶意节点容忍度低

- 不同的方法，共同的目的，为每个Block提供信任支撑

02.

智能合约及数据模型

智能合约的发展历史及主流的两类数据模型

1 区块链技术基石

不同区块链平台都依赖的几个基础技术

3 经典项目概览

经典项目彼此技术上的独特性

02/1 智能合约演进历史

智链CHAINNOVA
www.chainnova.com

智能合约概念提出

Nick Szabo
'A smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises.'

Bitcoin/P2PKH

Forth-like, stack based
简单脚本
非图灵完备
不支持循环体

Bitcoin/P2SH

BIP 16
支持更复杂的脚本
如Multisig、Escrow

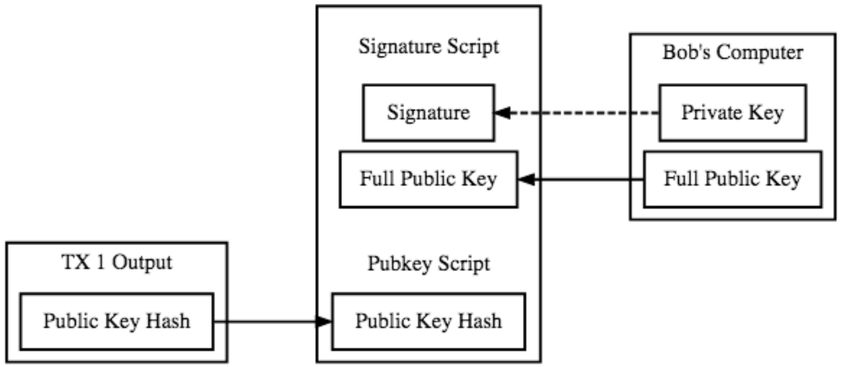
Ripple/NXT

内置合约模板，根据需求选择使用，不支持定制

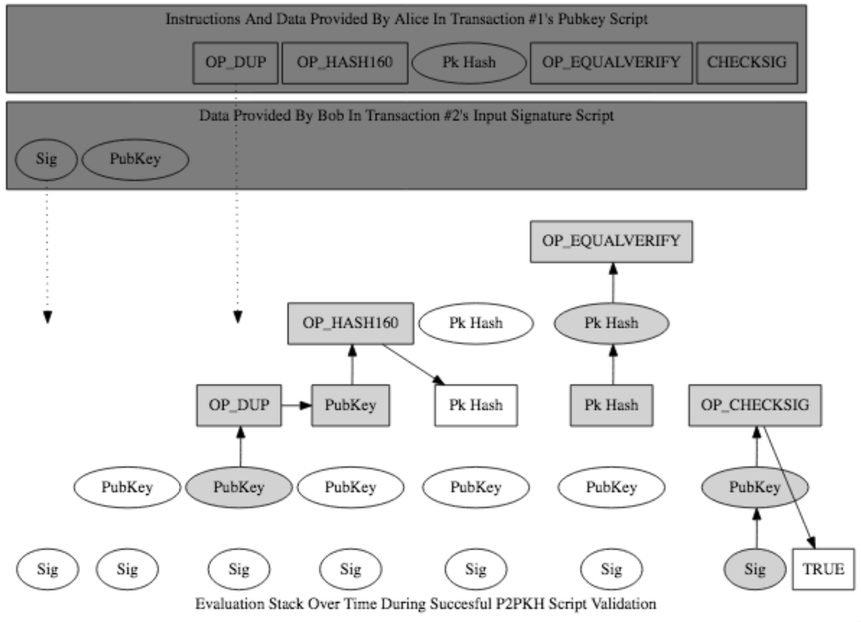
Ethereum

图灵完备的智能合约，运行在EVM虚拟机上，Solidity语言

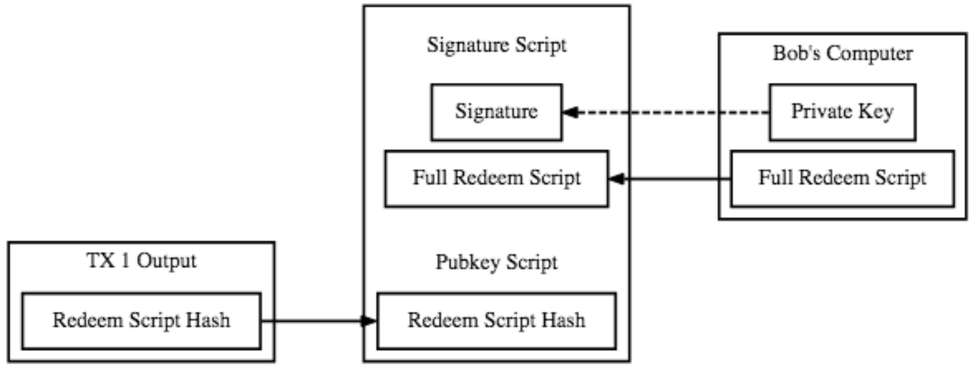
02/2 Bitcoin里的合约: P2PKH, P2SH



Spending A P2PKH Output

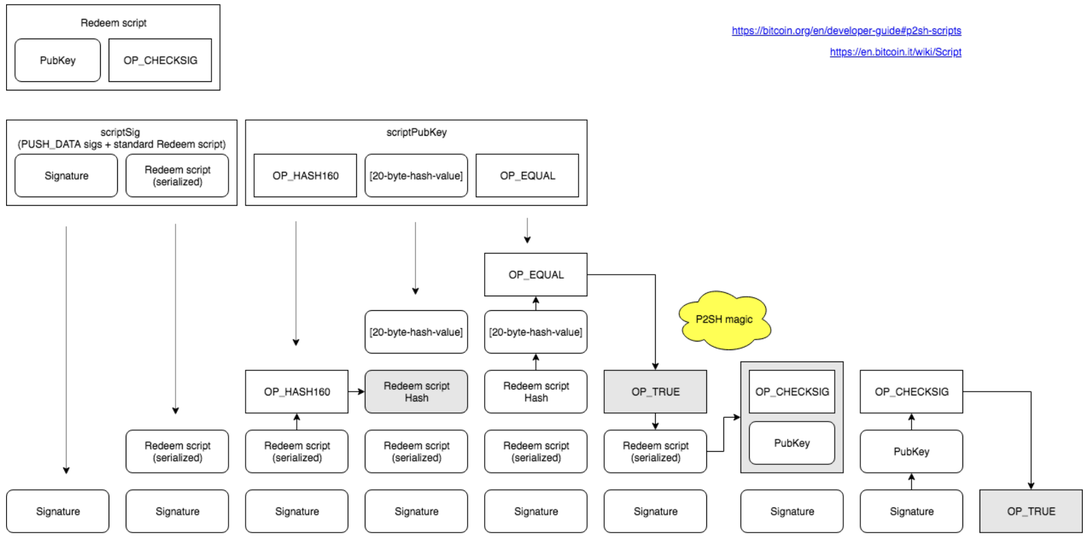


Evaluation Stack Over Time During Succesful P2PKH Script Validation



Spending A P2SH Output

P2SH script execution



Evaluation Stack Over Time During Succesful P2SH Script Validation

02/3 Ripple里的合约: TX

Field	JSON Type	Internal Type	Description
Account	String	Account	*(Required)* The unique address of the account that initiated the transaction.
TransactionType	String	UInt16	*(Required)* The type of transaction. Valid types include: `Payment`, `OfferCreate`, `OfferCancel`, `TrustSet`, `AccountSet`, `SetRegularKey`, `SignerListSet`, `EscrowCreate`, `EscrowFinish`, `EscrowCancel`, `PaymentChannelCreate`, `PaymentChannelFund`, and `PaymentChannelClaim`.
Fee	String	Amount	*(Required; auto-fillable)* Integer amount of XRP, in drops, to be destroyed as a cost for distributing this transaction to the network. Some transaction types have different minimum requirements. See Transaction Cost for details.
Sequence	Unsigned Integer	UInt32	*(Required; auto-fillable)* The sequence number, relative to the initiating account, of this transaction. A transaction is only valid if the `Sequence` number is exactly 1 greater

Example Payment JSON

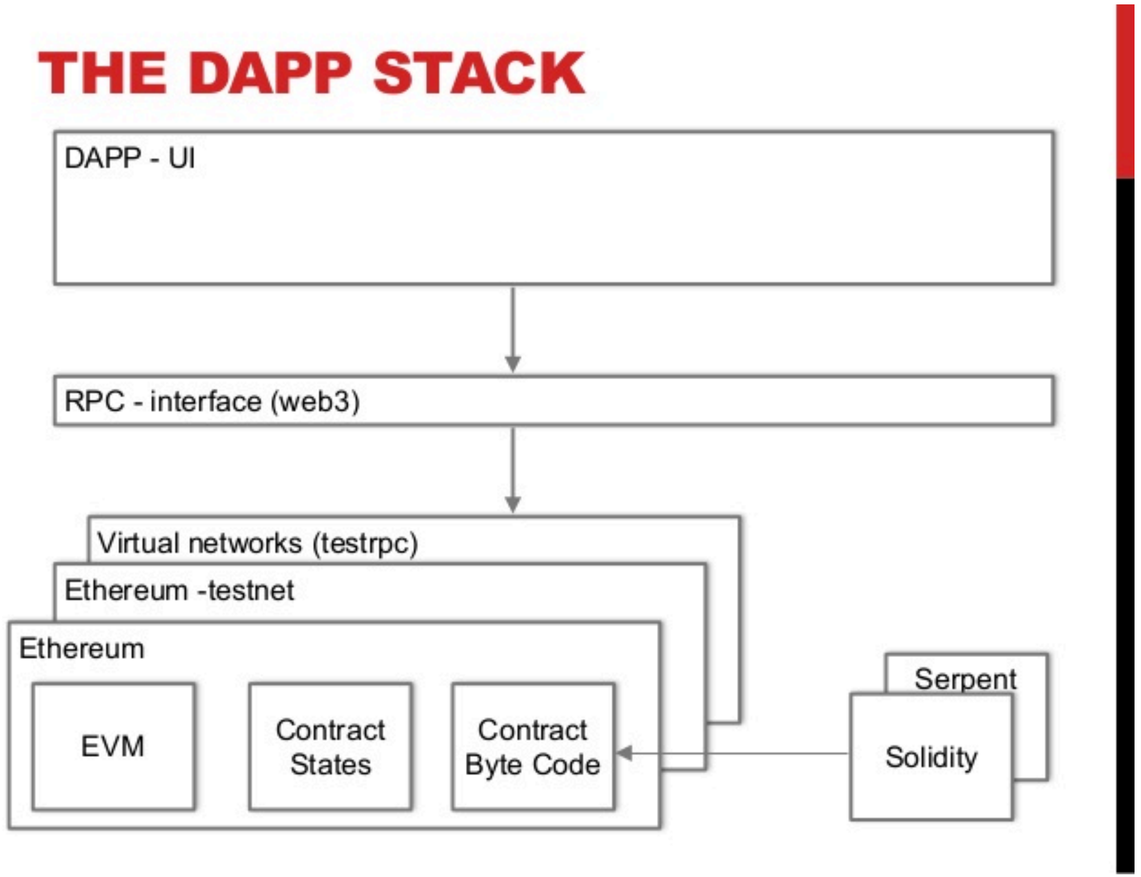
```
{
  "TransactionType" : "Payment",
  "Account" : "rf1BiGeXwwQoi8Z2ueFYTEXSwuJYfV2Jpn",
  "Destination" : "ra5nK24KXen9AHvsdFTKHSANinZseWnPcX",
  "Amount" : {
     "currency" : "USD",
     "value" : "1",
     "issuer" : "rf1BiGeXwwQoi8Z2ueFYTEXSwuJYfV2Jpn"
  },
```

Expand

- 有限的交易类型
- 预置的参数字段
  - 优点：简单

https://developers.ripple.com/transaction-common-fields.html

02/4 Ethereum里的合约: EVM, Solidity




```
pragma solidity ^0.4.22;

/// @title Voting with delegation.
contract Ballot {
    // This declares a new complex type which will
    // be used for variables later.
    // It will represent a single voter.
    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;   // index of the voted proposal
    }

    // This is a type for a single proposal.
    struct Proposal {
        bytes32 name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    // This declares a state variable that
    // stores a `Voter` struct for each possible address.
    mapping(address => Voter) public voters;

    // A dynamically-sized array of `Proposal` structs.
    Proposal[] public proposals;

    /// Create a new ballot to choose one of `proposalNames`.
    constructor(bytes32[] proposalNames) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;

        // For each of the provided proposal names,
        // create a new proposal object and add it
        // to the end of the array.
        for (uint i = 0; i < proposalNames.length; i++) {
            // `Proposal({...})` creates a temporary
            // Proposal object and `proposals.push(...)`
            // appends it to the end of `proposals`.
            proposals.push(Proposal({
                name: proposalNames[i],
                voteCount: 0
```

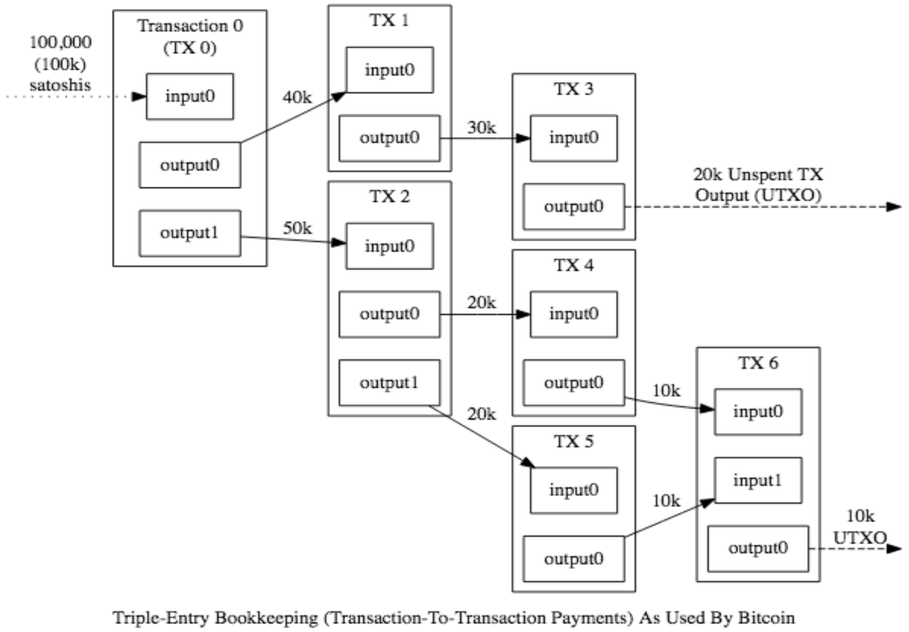
https://www.slideshare.net/MartinKppelmann/build-dapps-13-dev-tools

http://solidity.readthedocs.io/en/v0.4.24/solidity-by-example.html

02/5 两大主流数据模型

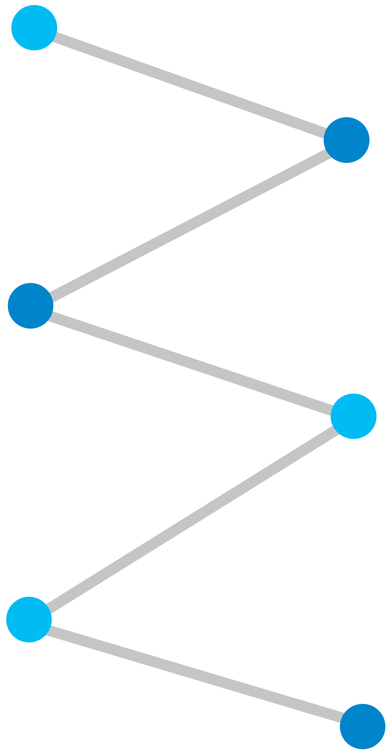
UTXO

- 以Bitcoin为代表
- 每一笔交易的输出作为下一笔交易的输入



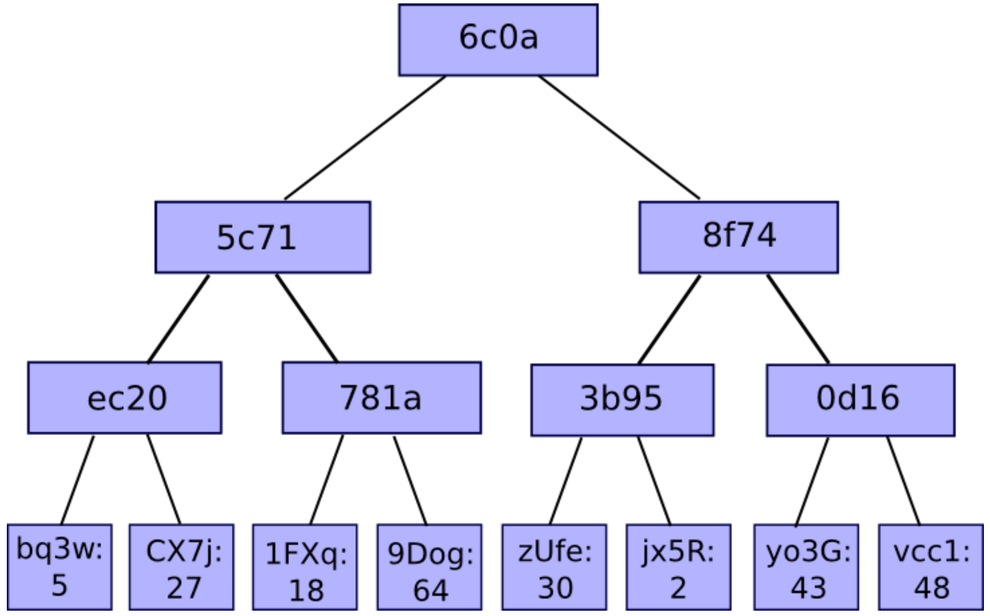
Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

https://bitcoin.org/en/developer-guide#transaction-data



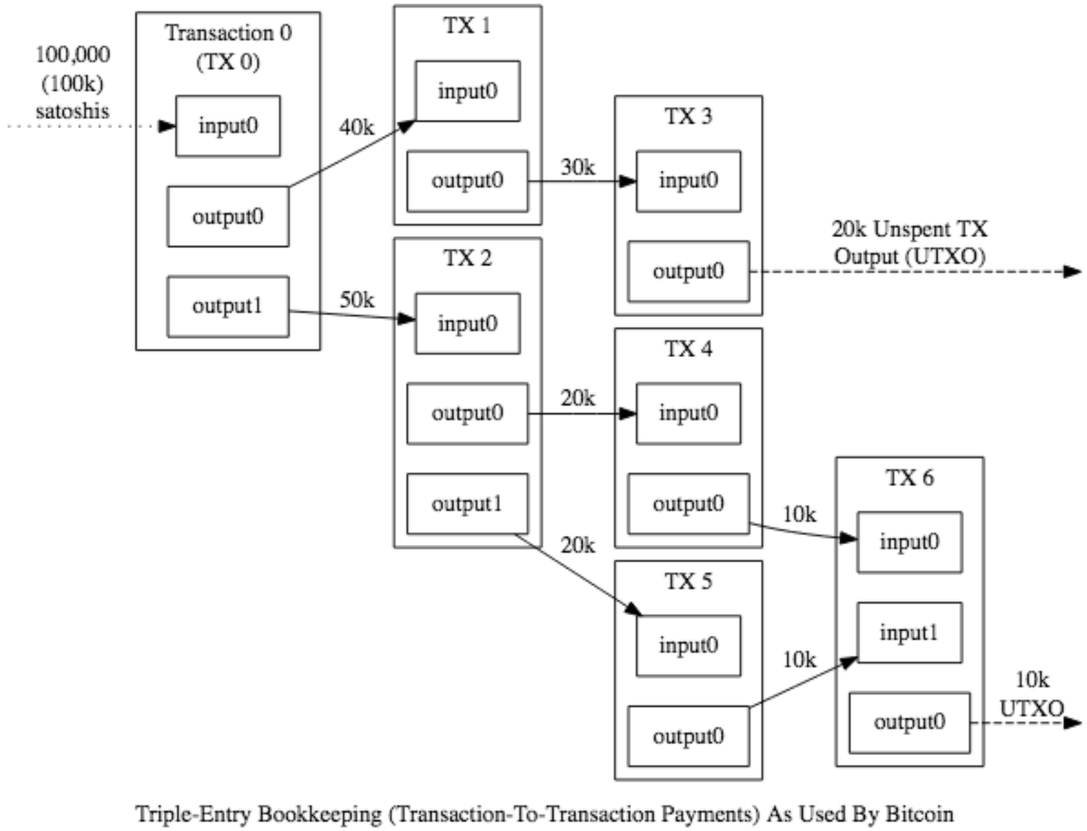
World State

- 以Ripple、Ethereum、HyperLedger Fabric为代表
- 一个全局的账户余额表



https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/

02/6 数据模型：UTXO

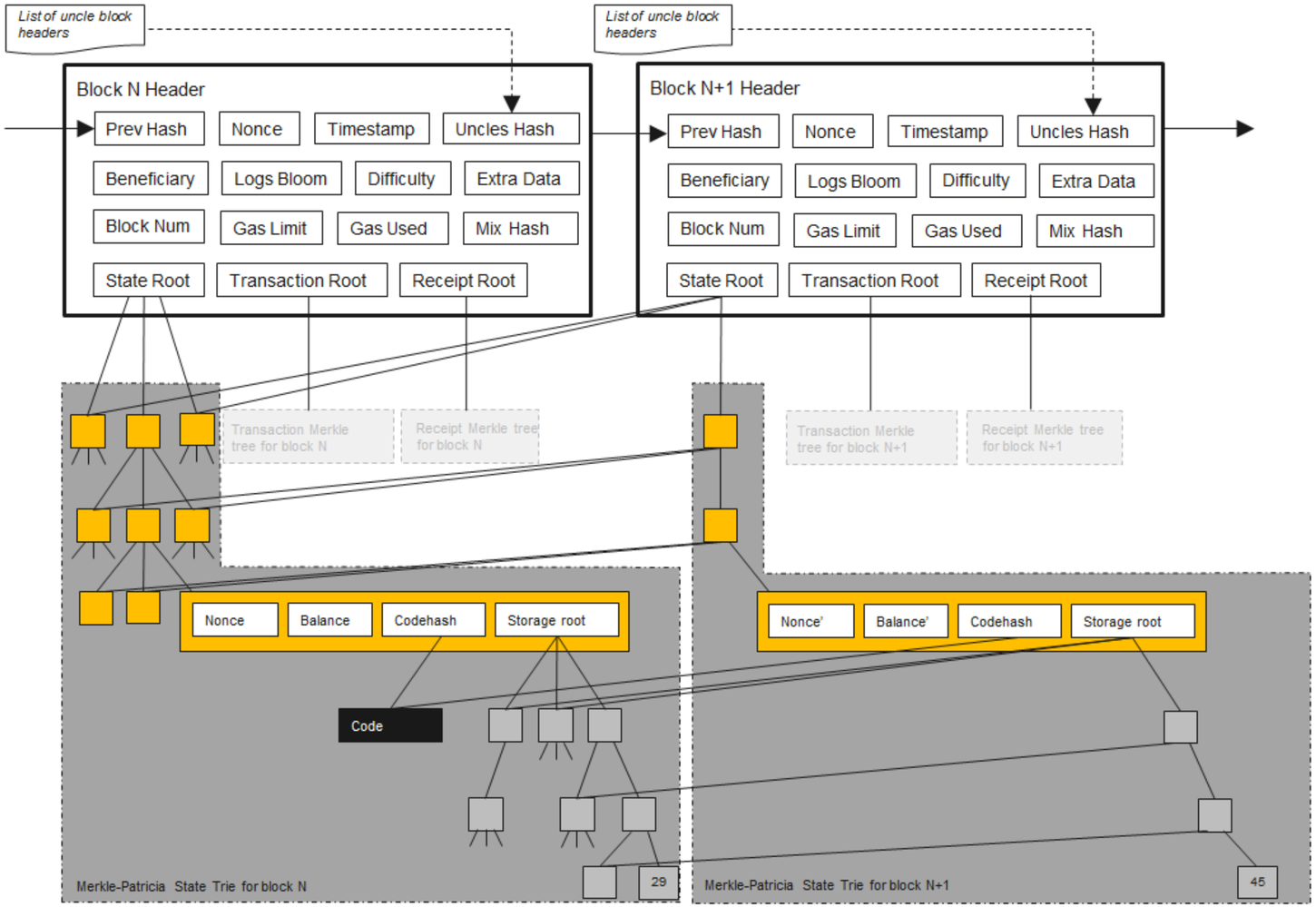


Triple-Entry Bookkeeping (Transaction-To-Transaction Payments) As Used By Bitcoin

https://bitcoin.org/en/developer-guide#transaction-data

- UTXO即未花费交易输出
- 以Bitcoin为代表
- 安全性强
- 扩展性高
- 保护隐私
- 存储压力较大
- 复杂度提升
- 查询性能有损失

02 数据模型：World State



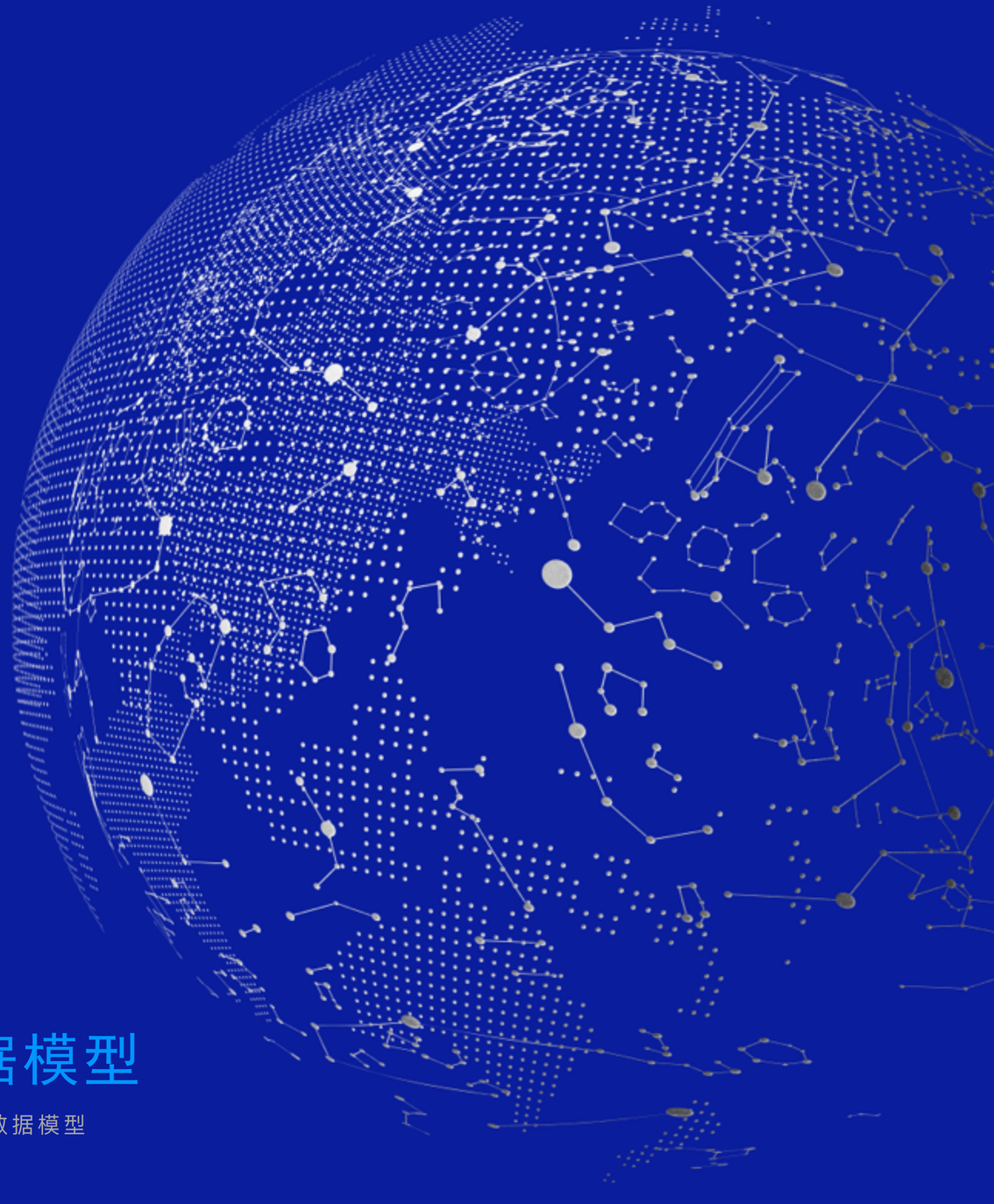
- 保存当前所有账户状态
- Copy on write模型
- 以Ripple、Ethereum、HyperLedger Fabric为代表
- 结构简单
- 效率高
- 历史追溯效果较弱
- 易受双花攻击

https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture/757#757

03.

经典项目概览

经典项目彼此技术上的独特性



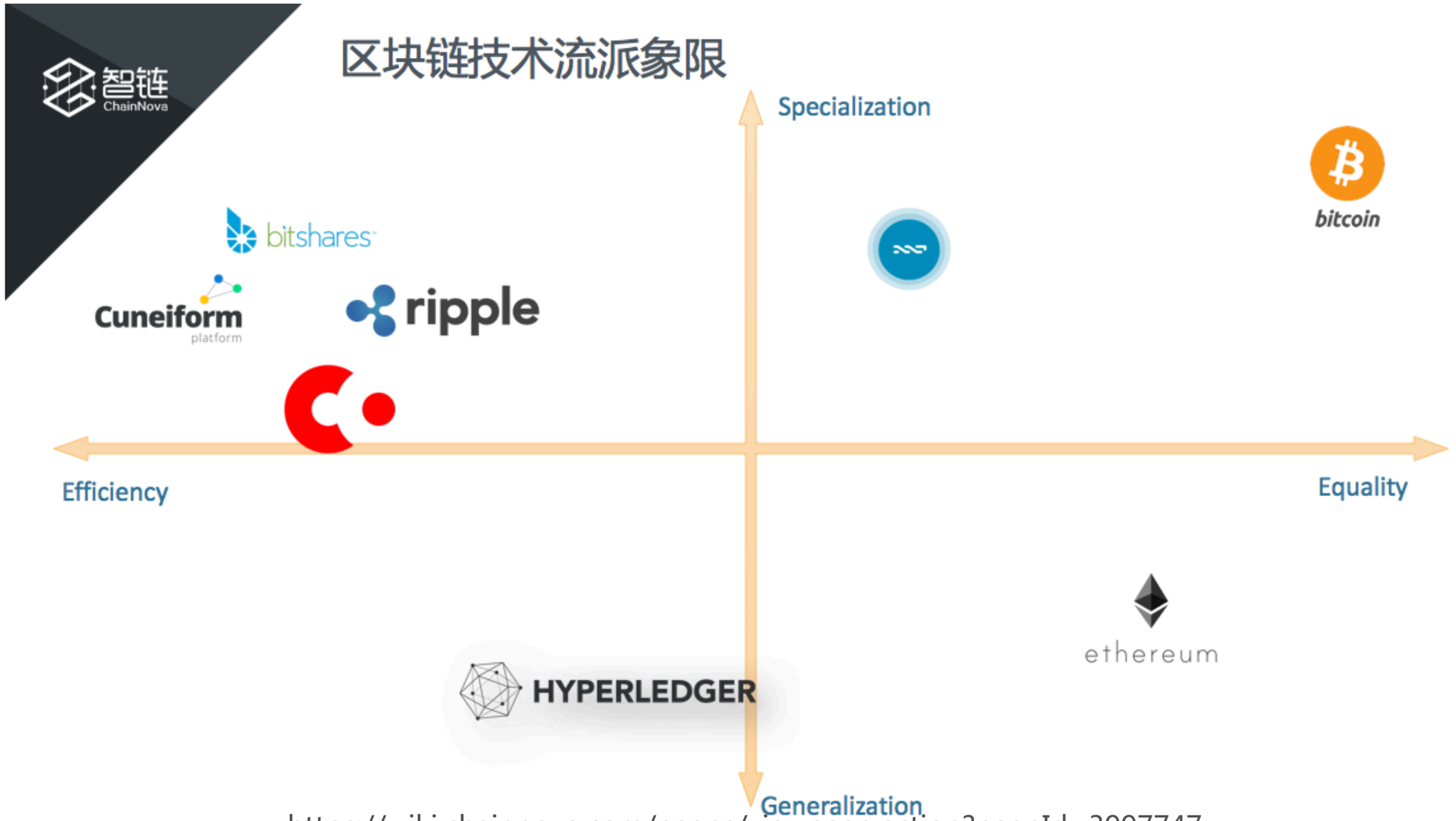
1 区块链技术基石

不同区块链平台都依赖的几个基础技术

2 智能合约及数据模型

智能合约的发展历史及主流的两类数据模型

项目象限图



https://wiki.chainnova.com/pages/viewpage.action?pageId=3997747

03/2 主流项目概览

名称	共识	开发语言	智能合约	存储	延迟/TPS	适用场景	类型	扩展性	成熟度	通用性	性能
Fabric	Kafka/ PBFT	Go	Docker/ Go	LevelDB/ CouchDB	3sec/200	数据共享、溯源、防篡改、数字资产交易、积分	联盟链	高	中	高	中
Corda	非常规，多种共识	Kotlin	JVM/ Kotlin	H2	非全局账本，无法量化	汇兑，供应链金融	联盟链	中	高	中	高
Ripple	Quorum	C++	No	RocksDB	10sec/1K	汇兑，数字资产交易	公有链	中	中	中	中
Ethereum	POW	Go	EVM/ Solidity	LevelDB	15sec/ 10+	去中心化应用	公有链	高	中	高	低
Bitshares	DPOS	C++	No	LevelDB	15sec/1M	数字资产交易	公有链	中	低	中	高
Bitcoin	POW	C++	No	LevelDB	10min/7	去中心化点对点支付	公有链	低	高	低	低

https://wiki.chainnova.com/pages/viewpage.action?pageId=3997747

主办方 Geekbang 极客邦科技 InfoQ

THANKS

智链 CHAINNOVA
www.chainnova.com

BCCon
Global
Blockchain
Eco-Technology
Conference