区块链编程

keyou
2019.12.12

区块链开发的维度

1. **Dapp 开发**：在已有区块链平台上开发智能合约并基于该合约开发配套应用，比如 CryptoKitties,xx溯源平台等；
2. BaaS 平台开发：围绕如何部署、运维、存储、安全等搭建云平台，方便其他人能够快速部署私有区块链，比如华为的BCS，360的BaaS平台；
3. 公链开发：区块链程序本身的开发，比如Ethereum,EOS,hyperledger/fabric,小蚁的NEO，360的磐石链等；

在Ethereum上进行Dapp开发

1. 搭建用于测试的区块链网络（搭建私有链）；
2. 安装编译工具；
3. 编写智能合约；
4. 部署智能合约；
5. 调用智能合约；
6. 封装为Dapp；

1. 部署单节点私有网络

1. 安装geth程序，可以下源码也可以下载已经编好的包；
2. 使用 geth account 创建新**账户**；
3. 使用puppeth生成genesis.json文件；
4. 使用 geth --datadir data2 init data2/141.json 初始化节点；
5. 使用以下命令启动节点；

```
geth --allow-insecure-unlock --datadir ./data --networkid 141 --rpc --ws --unlock
0x6101fd752d01cbecf55359ab8c56083d11d49f86 --mine
```

2. 安装solc（solidity compiler）

最简单的方式：
npm install -g solc

或者安装C++ snap版(Linux):
sudo snap install solc

或者安装C++ deb版(ubuntu)：
sudo add-apt-repository ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install solc

3. 编写智能合约

- 合约采用solidity语言进行编写；
- 它由以太坊开发；
- 专门用于编写智能合约；
- 运行在EVM虚拟机上，使用go开发内置于geth程序中；
- 是一种面向对象语言；
- 合约代码一般不会很长；
- 越长的合约代码部署时需要消耗的**gas**就越多；

```
1  // filename: Test.sol
2  pragma solidity ^0.5.0;
3
4  contract Test{
5      uint public click_times = 10;
6      mapping(address => uint) public times;
7      event print(address addr);
8
9      constructor() public {
10         click_times = 5;
11     }
12
13     function click(uint t) public {
14         if(click_times<=0) return;
15         click_times -= t;
16         times[msg.sender]++;
17         emit print(msg.sender);
18     }
19
20     function getData() public view returns (uint) {
21         return times[msg.sender];
22     }
23 }
```

4. 部署智能合约 —— 编译

用以下命令编译Test.sol合约

```
solc --bin --abi Test.sol
```

它生成了2个重要的内容，BIN 和 ABI；

BIN是编译生成的合约程序；

ABI是调用该合约程序时要使用的接口描述；

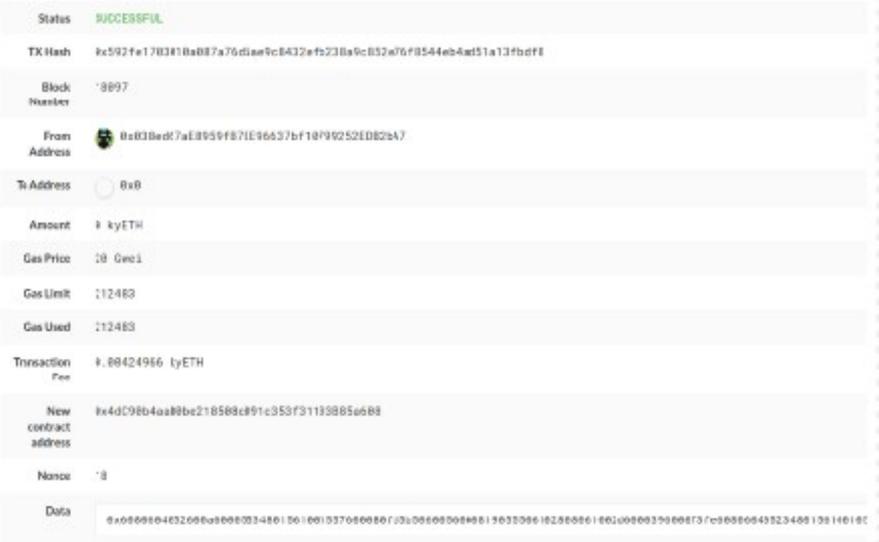
```
keyou@zephyrm /media/keyou/dev/works/ether-test/contracts
$ solc --bin --abi Test.sol

======= Test.sol:Test =======
Binary:
6080604052600a600055348015610015576000 80fd5b506005600081905550610280806100 2d60003960
d5b506004361061004c5760003560e01c80633bc5de3014610051578063ca8731bd1461006f578063ce7
5b600080fd5b61005961011356 5b604051808281526020019150506040518091039 0f35b61007761015a
0910390f35b6100cf6004803603602081101561 00a357600080fd5b81019080803573ffffffffffffff
92919050505061016056 5b604051808281526020019150506040518091039 0f35b610111600480360360
590602001909291905050506101785 65b005b6000600160003373ffffffffffffffffffffffffffffff
ffffffffffffff1681526020019081526020016000205490509056 5b60005481565b6001602052806 0
0805411610186576102485 65b806000808282540392505081905550600160003373ffffffffffffffff
ffffffffffffffffffffffffffff1681526020019081526020016000206000815480929190600101919
0afc3df2ad502d792bb1fca6c780042bc18e333604051808273ffffffffffffffffffffffffffffffff
ffffffffffff1681526020019150506040518091039 0a15b5056fea265627a7a72315820dfd18d05cc
c5c264b355f88782a64736f6c634300050d0032
Contract JSON ABI
[{"inputs":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"},{
":false,"internalType":"address","name":"addr","type":"address"}],"name":"print","ty
s":[{"internalType":"uint256","name":"t","type":"uint256"}],"name":"click","outputs"
:"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"click_times","
name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"functio
e":"getData","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"paya
type":"function"},{"constant":true,"inputs":[{"internalType":"address","name":"","ty
ts":[{"internalType":"uint256","name":"","type":"uint256"}],"payable":false,"stateMu
```

4. 部署智能合约 —— 部署

合约的部署有三种方法：
1. 使用geth提供的console接口；
2. 使用第三方钱包；
3. 编写代码，调用geth提供的web3.js接口；
4. 编写代码，调用geth提供的RPC/HTTP/Websoket接口进行部署；

所有的部署方式会产生相同的效果；
部署后会生成一个**合约地址**；
合约代码存储在**交易**的data字段上传到网络中；

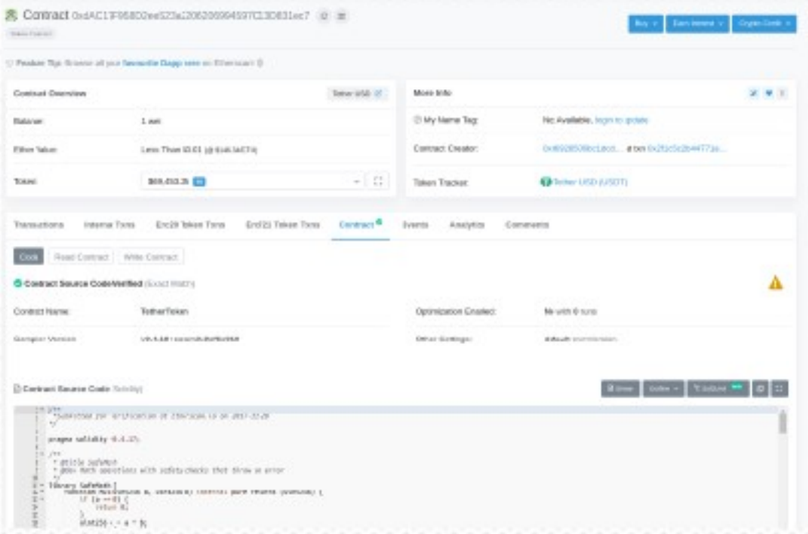


4. 部署智能合约 —— 确认

当你要参与一个智能合约时你可能需要检查该合约是否是你期望的，称为**Contract Verify**。

打开官方的区块链浏览器，输入合约地址，上传合约代码进行在线验证；

主要校验代码是否和合约中的二进制代码一致，也可以在本地进行校验；



5. 调用智能合约

可以通过以下方法调用合约：
1. 使用geth提供的console接口；
2. 使用第三方钱包；
3. 编写代码，调用geth提供的web3.js接口；
4. 编写代码，调用geth提供的RPC/HTTP/Websoket接口进行部署；

演示

6. 封装Dapp

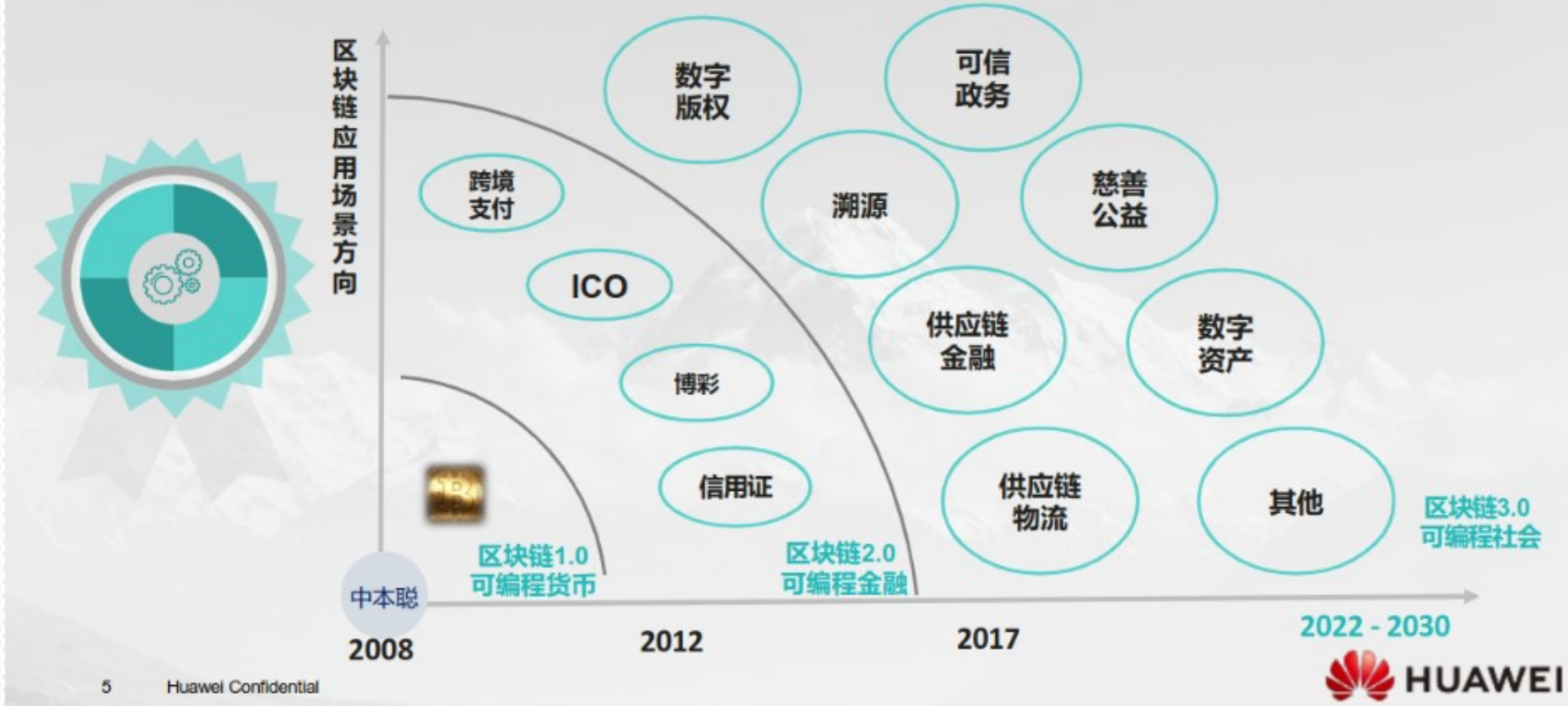
Dapp是一个应用，它和合约进行通信，**将状态/数据存储在区块链中**，除此之外和普通应用并无区别；

和区块链通信的方法有以下几种：
1. 使用web3.js接口；
2. 使用RPC/HTTP/Websoket接口；

演示

区块链的现状

区块链场景正从金融向众多行业拓展



HUAWEI


区块链第一代
区块链第二代
区块链第三代
2008 ~
2014 ~
2019 ?
支付货币
隐私加密保护
CASH
XMR
2014
ZCASH
2016
智能合约
BTC
2008
ETH
2014
EOS
2017
严格意义上的有待发掘
LTC
2011
DAG
IOTA
IOTA
2015

当前流行的Dapp的市场行情

DApp Name	DApp Platform	Category	Users in the last 24 hours
PRA Candybox	EOS	Token Distribution	7600
OCADAPP			
ENBank			
Gakex			
EOS Knights			

DApp Name	DApp Platform	Category	Transaction Volume last 24 hours
TRONbet	Tron	Gambling	$6.9 million
TronWoW			
Poker EOS			
EOS Jacks			
FCK			

DApp Name	DApp Platform	Category	Number of transactions in the last 24 hours
TronWow	Tron	Gambling	364.5k
POKE 25	EOS	Gambling	309.5k
BetHash	EOS	Gambling	303.3k
TRONbet	Tron	Gambling	235.3k
Dice	EOS	Gambling	161.2k

区块链专利

排名	企业简称	国别/地区	2019年公开的全球区块链发明专利申请数量/件	2018年公开的全球区块链发明专利申请数量/件
1	阿里巴巴	中国	1005	155
2	中国平安	中国	464	48
3	Nchain	安提瓜和巴布达	314	155
4	Bizmodeline	韩国	219	0
5	微众银行	中国	217	19
6	元征科技	中国	185	62
7	复杂美	中国	172	16
8	IBM	美国	169	96
9	瑞策科技	中国	141	0
10	腾讯	中国	137	38
11	网心科技	中国	131	22
12	百度	中国	129	21
13	众安科技	中国	123	27
14	Walmart	美国	115	57
15	中国联通	中国	111	40

编号	专利名称
1	基于黑名单灰名单的打币拦截方案
2	企业闲散算力管理平台
3	区块链节点的画像数据获取方案
4	基于地址树的可疑节点真实身份溯源方案
5	区块链节点的身份信息绑定方案
6	区块链攻击事件的感知方法、装置及计算设备
7	基于算法类型的区块链攻击事件感知方法及装置
8	基于时间戳的区块链攻击事件感知方法及装置
9	基于交易笔数的区块链攻击事件感知方法及装置
10	基于区块链的空间搜索方法、系统及计算设备
11	基于区块链的威胁情报处理方法、系统及计算设备

IPRDAILY中文网与INCOPAT创新指数研究中心联合发布

人家口中的区块链

基础协议	语言	合约	共识	关账时间	技术支持方	特点	性能	开源许可
Bitcoin	C++	是	PoW	10min	Core/Classic	公链+图灵不完备	7	MIT
Ethereum	Go/C++	是	PoW/PoS	12S左右	EthCore	公链/联盟+图灵完备,GAS	25	GPL
Fabric	Go	是	PBFT		Linux基金会	开发框架	300~2000	Apache
BitShares	C++	否	DPoS			高性能、分布式交易	100K	MIT
Ripple	C++		RPCA	1S-8S	Ripple	分布式交易	1000	MIT
Stellar	C++	是	SCP/FBA	5S左右	Jed Mccaleb/Stellar基金会	分布式交易/图灵不完备	1000	Apache
Tendermint	Go	是	PBFT	可调		开发框架	10K	Apache
……								

1. 所选平台适应的链类型（公有链、联盟链、专有链）与应用使用范围的匹配程度；
2. 所选平台使用的开发平台与应用技术开发平台的匹配程度；
3. 所选平台开发支撑环境（包括SDK、测试网络、开发文档、运维保障等等）情况；

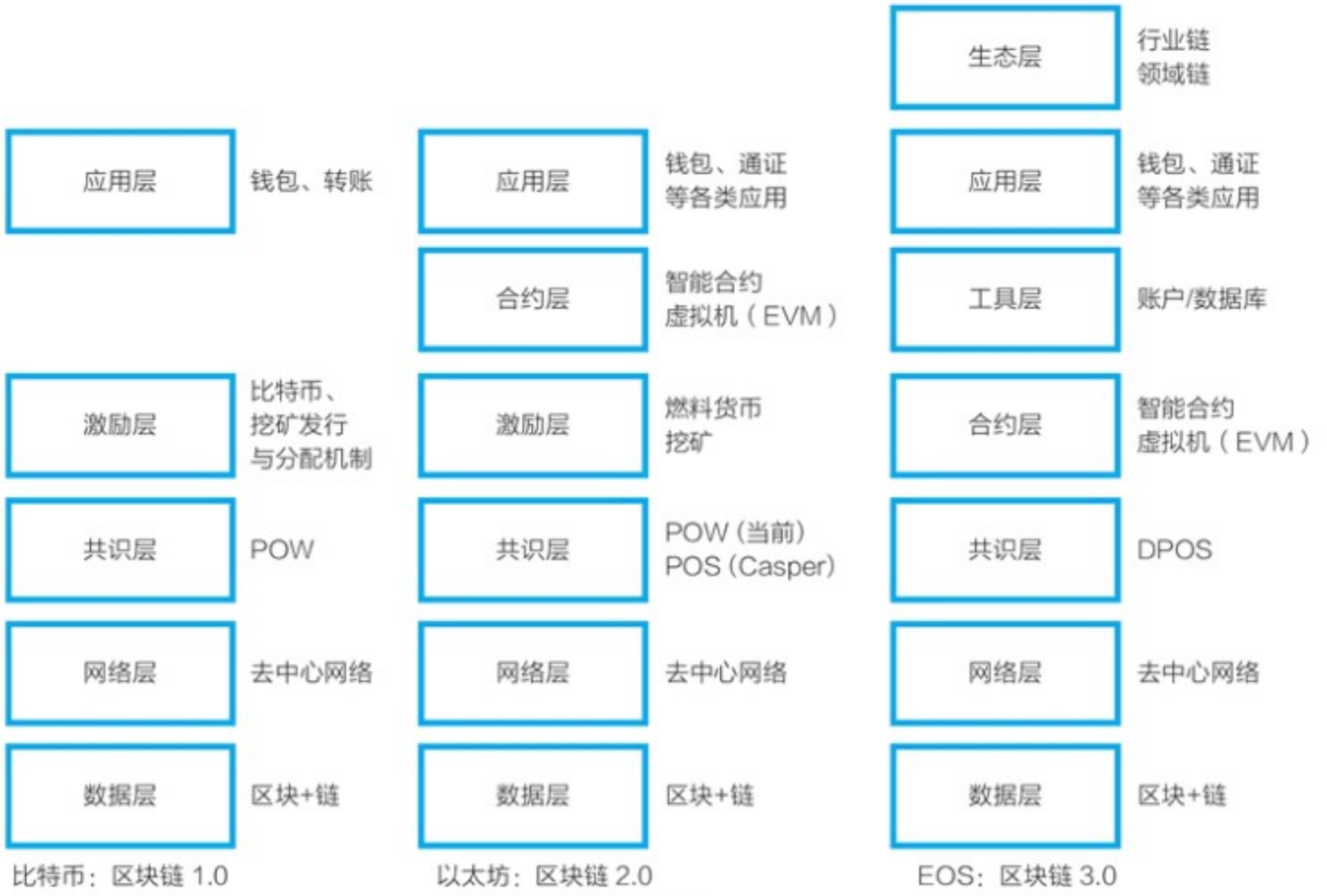

生态层
行业链
领域链
应用层
钱包、转账
应用层
钱包、通证
等各类应用
应用层
钱包、通证
等各类应用
合约层
智能合约
虚拟机（EVM）
工具层
账户/数据库
激励层
比特币、
挖矿发行
与分配机制
激励层
燃料货币
挖矿
合约层
智能合约
虚拟机（EVM）
共识层
POW
共识层
POW（当前）
POS（Casper）
共识层
DPOS
网络层
去中心网络
网络层
去中心网络
网络层
去中心网络
数据层
区块+链
数据层
区块+链
数据层
区块+链
比特币：区块链 1.0
以太坊：区块链 2.0
EOS：区块链 3.0

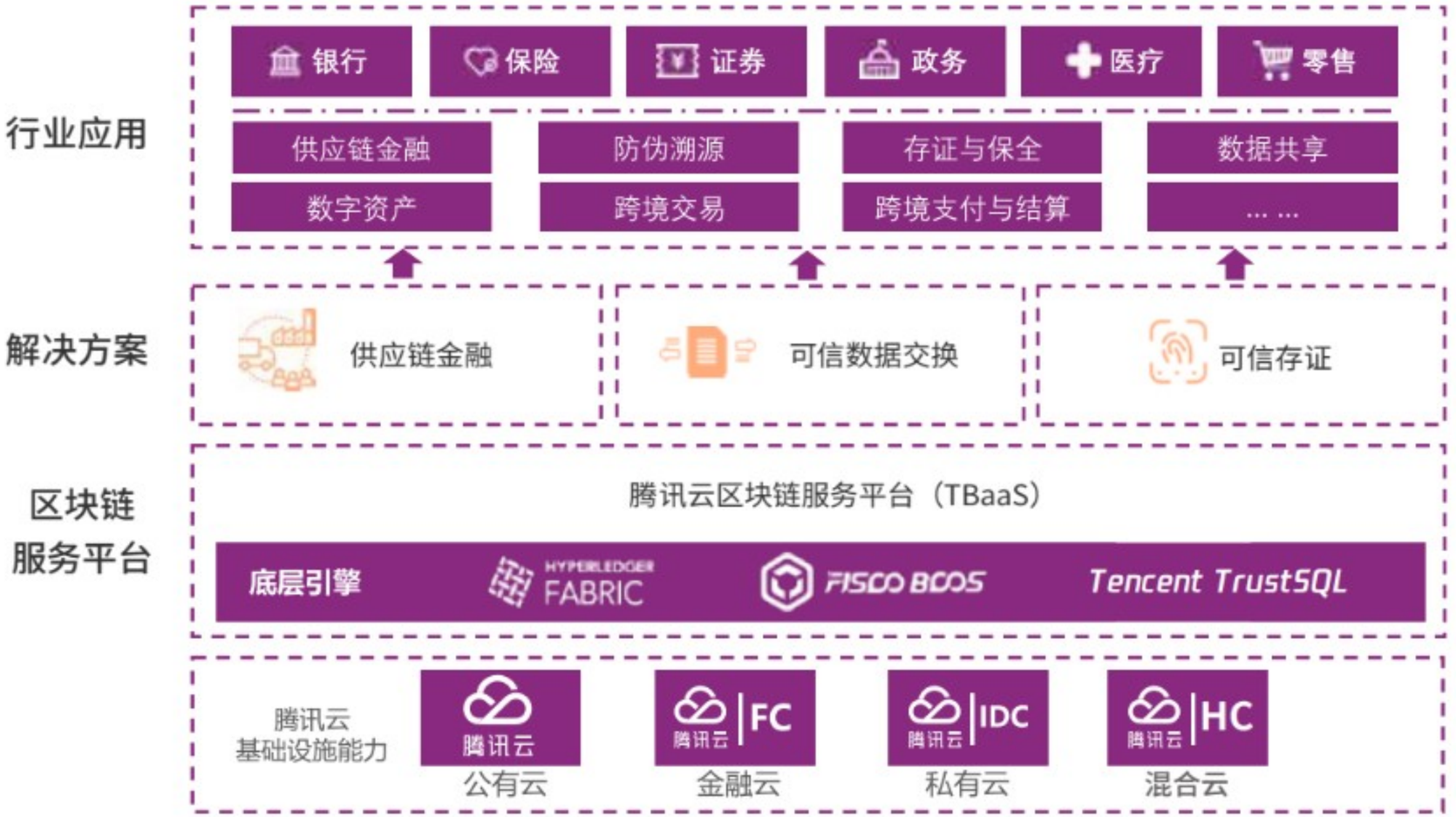
图3：EOS体系架构

腾讯区块链

图 3-1 TrustSQL 架构图

Bitcoin + Ethereum

腾讯云区块链



Fabric

图 3-6 腾讯云区块链总体能力

华为区块链

Fabric

360区块链


应用服务层
数字资产
数字存证
分布式账本
共享经济
区块链服务平台层
区块链BaaS平台
管理门户层
用户管理
账户管理
权限管理
密钥管理
风控审计
运营监控
配置部署
全息监控
合约管理
报表统计
基础服务
网关服务
接口适配
网络通讯
数据存储
智能合约层
合约开发
合约测试
沙箱发布
链上执行
区块链层
360磐石链
Ethereum
Fabric
容器云
镜像管理
容器部署
服务编排
代码发布
弹性伸缩
灰度升级
错误恢复
负载均衡
基础设施服务层
公有云
私有云
混合云

区块链的实现原理(超简版)

词汇表

ICOs
钱包 账户 交易
区块链浏览器 **挖矿** **区块** 确认数
区块链 公联 私链 联盟链
共识算法 **PoW** PoS PoA PBFT **拜占庭**容错 零知识证明
智能合约 通证**Token** 分布式应用Dapp
EIP/ERC **ERC20** ERC721
gossip bootnode 分片sharding **默克尔树** Merkle Patricia Trie **UTXO** ……

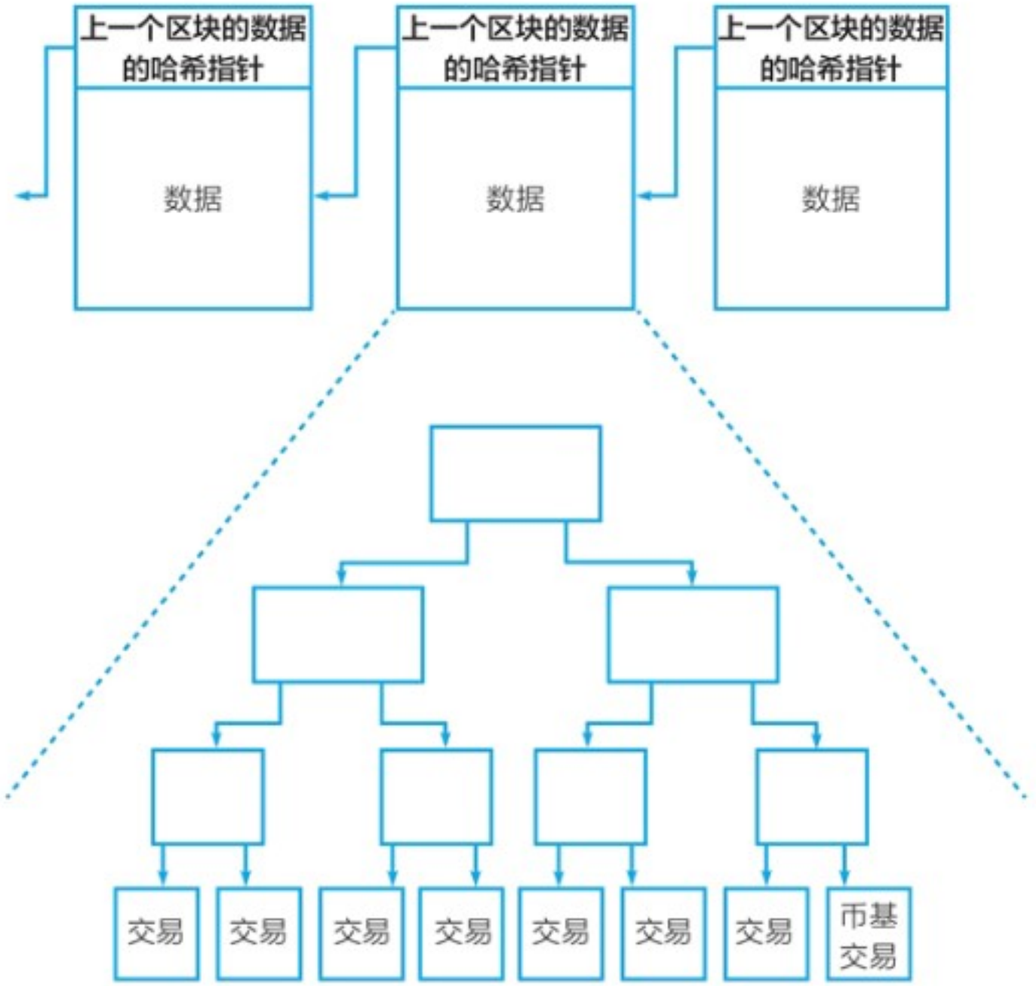
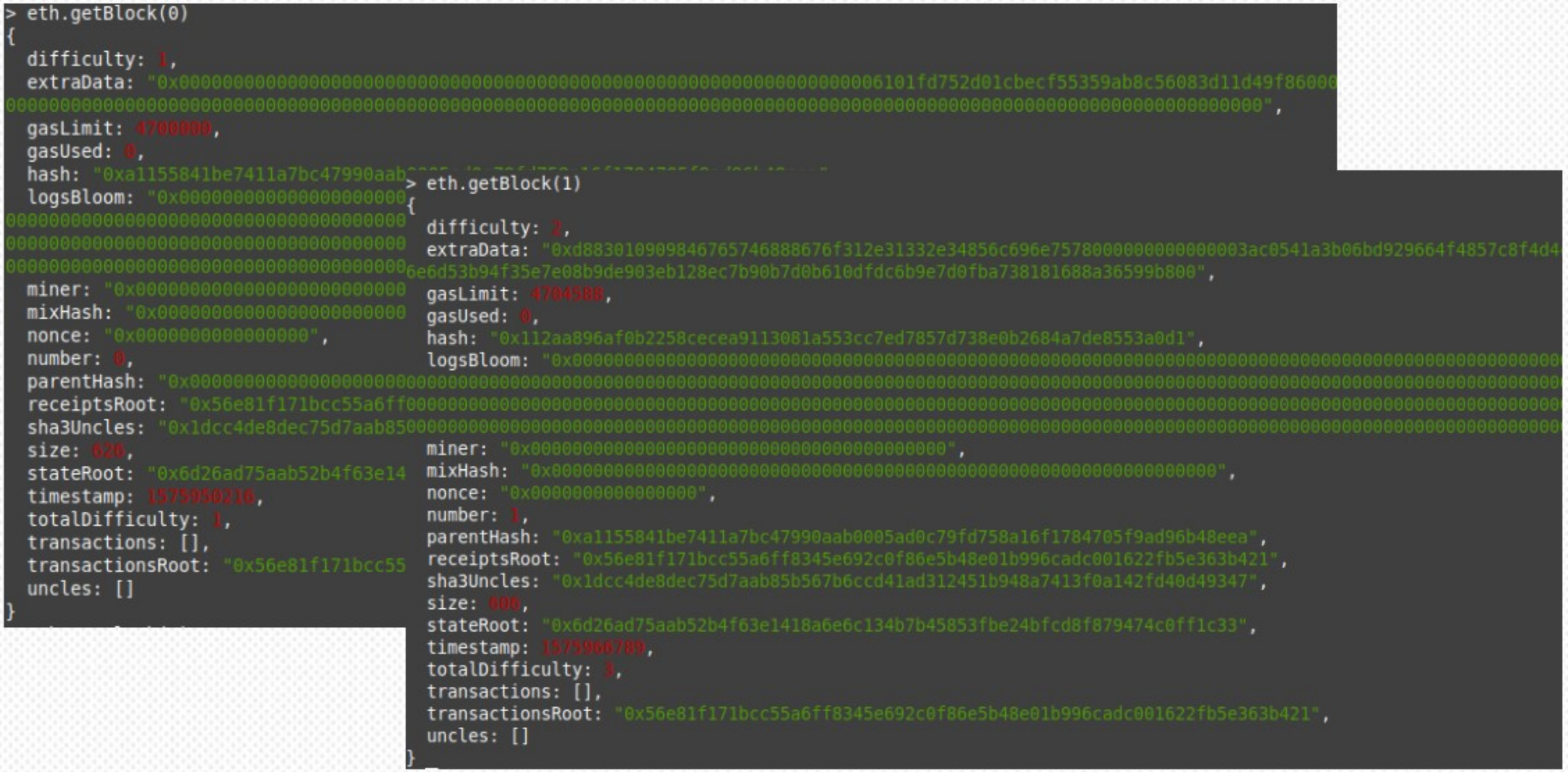


图1：比特币区块链的"区块链"与梅克尔树



```
> eth.getBlock(0)
{
  difficulty: 1,
  extraData: "0x0000000000000000000000000000000000000000000000000000000000000000000000006101fd752d01cbecf55359ab8c56083d11d49f86000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  gasLimit: 4700000,
  gasUsed: 0,
  hash: "0xa1155841be7411a7bc47990aab
  logsBloom: "0x00000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  miner: "0x0000000000000000000000000000000000000000",
  mixHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  nonce: "0x0000000000000000",
  number: 0,
  parentHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 620,
  stateRoot: "0x6d26ad75aab52b4f63e14
  timestamp: 1575950216,
  totalDifficulty: 1,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55
  uncles: []
}
```

```
> eth.getBlock(1)
{
  difficulty: 2,
  extraData: "0xd883010909846765746888676f312e31332e34856c696e757800000000000003ac0541a3b06bd929664f4857c8f4d4
6e6d53b94f35e7e08b9de903eb128ec7b90b7d0b610dfdc6b9e7d0fba738181688a36599b800",
  gasLimit: 4704588,
  gasUsed: 0,
  hash: "0x112aa896af0b2258cecea9113081a553cc7ed7857d738e0b2684a7de8553a0d1",
  logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
  miner: "0x0000000000000000000000000000000000000000",
  mixHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  nonce: "0x0000000000000000",
  number: 1,
  parentHash: "0xa1155841be7411a7bc47990aab0005ad0c79fd758a16f1784705f9ad96b48eea",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 600,
  stateRoot: "0x6d26ad75aab52b4f63e1418a6e6c134b7b45853fbe24bfcd8f879474c0ff1c33",
  timestamp: 1575960789,
  totalDifficulty: 3,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

简答（以下回答针对比特币）

1. 挖矿是在做什么?
生成一个新的区块，交易被记录到区块中，获得代币奖励，也就是新币被凭空发行出来；
新的区块要保存一个hash值，该hash值需要满足一定的条件（小于某一个数），只有通过大量的计算才可能得到。调整区块中的nonce值来使该条件成立。小于的那个数是根据difficulty值计算来的。
2. 如何保证区块的生成速度是10分钟？（如何保证新币的发行速度是10分钟？）
统计生成2016个区块所需的平均时间，如果该时间比10分钟短，就使用更严格的条件，否则使用更宽松的条件。
3. 为什么要控制生成区块的速度?
控制新币的发行速度；
4. 如何防止区块链分叉?
分叉区块后面最先到达6个区块的叉获胜。

总结

我们想做什么？

1. **Dapp 开发**：在已有区块链平台上开发智能合约并基于该合约开发配套应用，比如 CryptoKitties,xx溯源平台等；
2. BaaS 平台开发：围绕如何部署、运维、存储、安全等搭建云平台，方便其他人能够快速部署私有区块链，比如华为的BCS，360的BaaS平台；
3. 公链开发：区块链程序本身的开发，比如Ethereum,EOS,hyperledger/fabric,小蚁的NEO，360的磐石链等；

我们的能力如何?

- Dapp开发的难度不大，传统的应用/系统开发人员+智能合约开发即可完成。
- BaaS平台的开发大多围绕区块链周边进行，这些部分都是传统领域，所以难度也不大;
- 公链开发涉及较多的分布式问题，需要有分布式系统开发经验，其中如何快速的达成共识，如何设计可编程部分是较难的部分，但是这方面有很多现有的算法可供参考，所以难度是可控的。

- 所以，这三个层面我们都能做。
- 我们之所以做IoT平台是因为我们有很多的设备需要管控，我们有强需求，而对于区块链，我们也需要找到这些强需求。

我们能做什么？

需要有想象力。

Q&A?

Thanks!