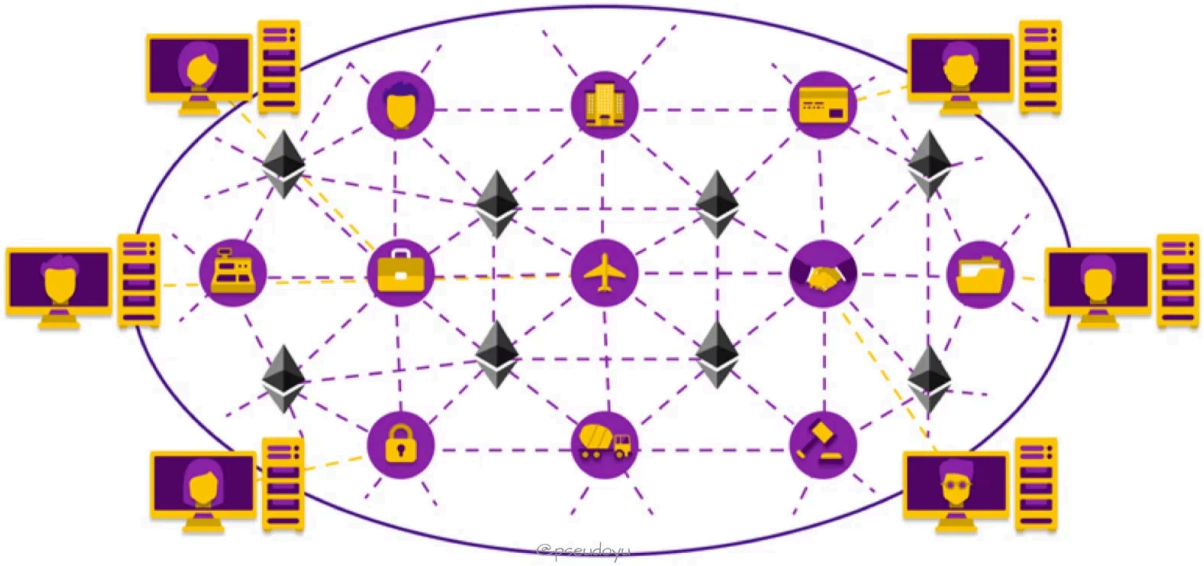
Ethereum 核心技术解读

前言

比特币作为一种去中心化的数字货币，是极其成功的，但受限于比特币脚本（非图灵完备，只能处理一些简单的逻辑），并不能处理很复杂的业务。而 Ethereum 引入了智能合约，使去中心化的概念能够应用于更丰富的应用场景，因此也被称为区块链 2.0。本文将对以太坊核心技术进行解读，如有错漏，欢迎交流指正。

Ethereum 系统

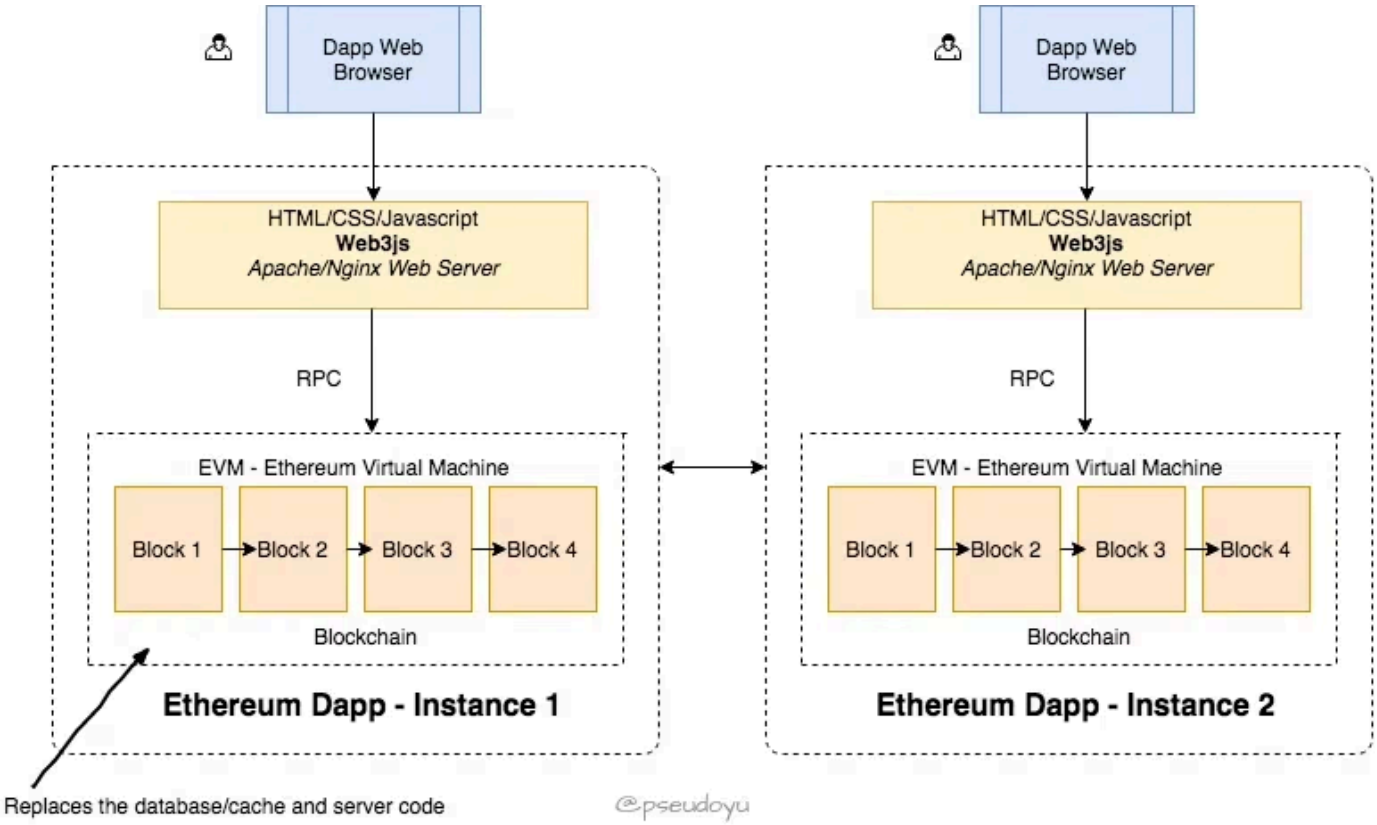
2014 年 1 月，俄罗斯开发者 Vitalik Buterin 发布了以太坊白皮书并成立团队，旨在创造一个集成更通用的脚本语言的区块链平台。其中一位成员 Dr. Gavin Wood 发布了一份黄皮书，涉及 Ethereum Virtual Machin(EVM) 以太坊虚拟的相关技术，这就是 Ethereum 的诞生。



简单来说， Ethereum 是一个开源的去中心化系统，使用区块链来存储系统状态变化，因此也被称为“世界计算机”；它支持开发者在区块链上部署运行不可变的程序，称为智能合约，因此可以支持广泛的应用场景；它使用数字货币 Ether 来衡量系统资源消耗，激励更多人参与 Ethereum 系统建设。

去中心化应用 DApp

狭义来说，DApp 其实就是一个集成了用户界面、支持智能合约、运行于以太坊区块链上的应用。



如上图所示， Ethereum 应用实例部署在区块链网络上（智能合约运行于区块链虚拟机中），而 Web 程序只需要通过 Web3.js 对区块链网络进行 RPC 远程调用，这样用户就可以通过浏览器（DApp 浏览器或 MetaMask 等插件工具）访问去中心化服务应用了。

账本

Ethereum 区块链是一个去中心化的账本（数据库），网络中的所有交易都会存储在区块链中，所有节点都要本地保存一份数据，并且确保每一笔交易的可信度；所有的交易都是公开且不可篡改的，网络中的所有节点都可以查看和验证。

账户

当我们需要登录一个网站或系统（比如邮箱）时，往往需要一个帐号和一个密码，密码通过加密算法以暗文的形式存储在中心化的数据库中。然而，以太坊是一个去中心化的系统，那是怎么生成账户的呢?

和比特币系统原理类似

1. 首先生成一个仅有自己知道的私钥，假设为 sk ，采用 ECDSA(Elliptic Curve Digital Signature Algorithm) 椭圆曲线算法生成对应的公钥 pk
2. 采用 keccak256 算法对公钥 pk 求哈希值
3. 截取后 160 位作为以太坊的地址

用户的私钥和地址一起组成了以太坊的账户，可以存储余额、发起交易等（比特币的余额是通过计算所有的 UTXO 得到的，而不是像以太坊一样存储在账户中）。

其实 Ethereum 账户分为两种类型，上述方式生成的叫 Externally Owned Accounts(EOA) ，外部账户，也就是常规用户拥有的账户，主要是用来发送/接收 Ether 代币或者向智能合约发送交易（即调用智能合约）。

而另一种则是 Contract Accounts ，合约账户，不同于外部账户，这种账户是没有对应的私钥的，而是在部署合约的时候生成的，存储智能合约代码。值得注意的是，合约账户必须要被外部账户或者其他合约调用才能够发送或接收 Ether ，而不能自己主动执行交易。

钱包

存储和管理 Ethereum 账户的软件/插件称为钱包，提供了诸如交易签名、余额管理等功能。钱包生成主要有两种方式，非确定性随机生成或根据随机种子生成。

Gas

Ethereum 网络上的操作也需要“手续费”，称为 Gas ，在区块链上部署智能合约以及转账都需要消耗一定单位的 Gas ，这也是鼓励矿工参与 Ethereum 网络建设的激励机制，从而使整个网络更加安全、可靠。

每个交易都可以设置相应的 Gas 量和 Gas 的价格，设置较高的 Gas 费则往往矿工会更快处理你的交易，但为了预防交易多次执行消耗大量 Gas 费，可以通过 Gas Limit 来设置限制。 Gas 相关信息可以通过 Ethereum Gas Tracker 工具进行查询。

```
If START_GAS * GAS_PRICE > caller.balance, halt
Deduct START_GAS * GAS_PRICE from caller.balance
Set GAS = START_GAS
Run code, deducting from GAS
For negative values, add to GAS_REFUND
After termination, add GAS_REFUND to caller.balance
```

智能合约

上文提到， Ethereum 区块链不仅仅存储交易信息，还会存储与执行智能合约代码。

智能合约控制应用和交易逻辑， Ethereum 系统中的智能合约采用专属 Solidity 语言，语法类似于 JavaScript ，除此之外，还有 Vyper 、 Bamboo 等编程语言。智能合约代码会被编译为字节码并部署至区块链中，一旦上链则不可以再编辑。 EVM 作为一个智能合约执行环境，能够保障执行结果的确定性。

智能合约示例：众筹

让我们想象一个更复杂的场景，假设我要众筹 10000 元开发一个新产品，通过现有众筹平台需要支付不菲的手续费，而且很难解决信任问题，于是，可以通过一个众筹的 DApp 来解决这个问题。

先为众筹设置一些规则

1. 每个想参与众筹的人可以捐款 10-10000 元的金额
2. 如果目标金额达成了，金额会通过智能合约发送给我（即众筹发起人）
3. 如果目标在一定时间内（如 1 个月）没有达成，众筹的资金会原路返回至众筹用户
4. 也可以设置一些规则，比如一周后，如果目标金额没有达成，用户可以申请退款

因为这些众筹条款是通过智能合约实现并部署在公开的区块链上的，即使是发起者也不能篡改条款，且任何人都可以查看，解决了信任问题。

完整代码可以点击这里查看：Demo

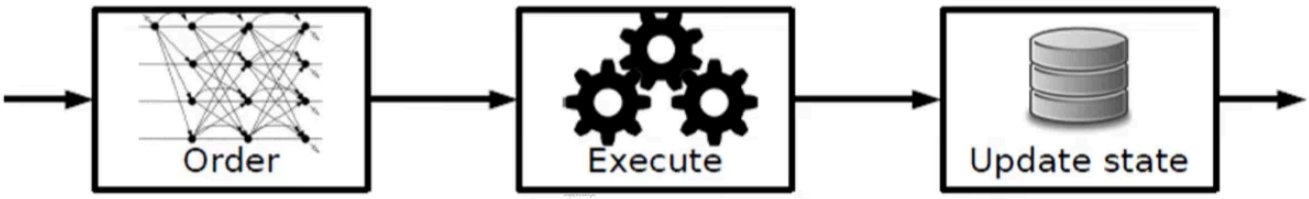
交易

在 Ethereum 中，一个典型的交易是怎么样的呢?

1. 开发者部署智能合约至区块链
2. DApp 实例化合约、传入相应值以执行合约
3. DApp 对交易进行数字签名

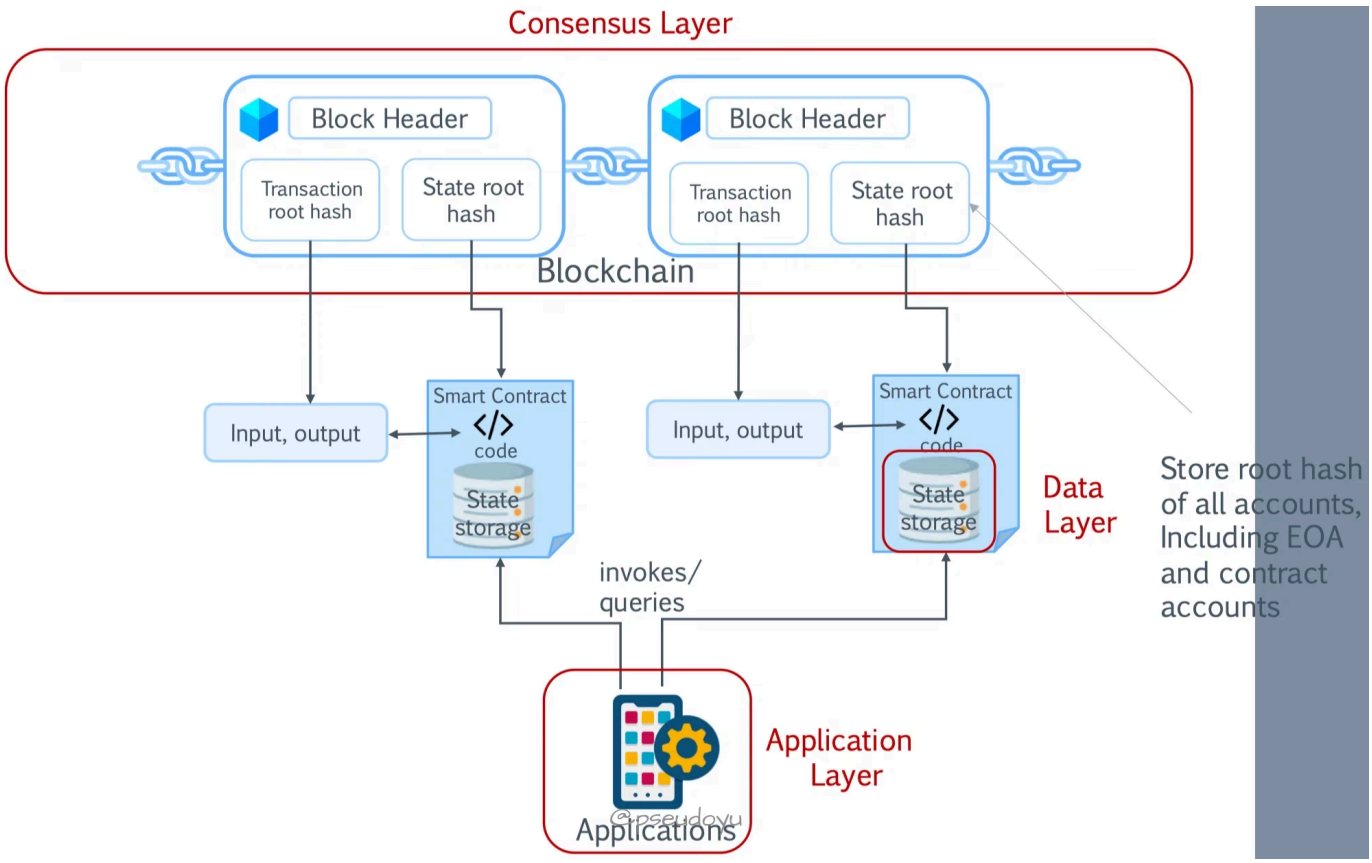
4. 本地对交易进行验证
5. 广播交易至网络中
6. 矿工节点接收交易并进行验证
7. 矿工节点确认可信区块后广播至网络中
8. 本地节点与网络进行同步，接收新区块

架构



Ethereum 采用的是一种 Order - Execute - Validate - Update State 的系统架构。在这种架构下，当产生一笔新的交易，矿工会进行 PoW 工作量证明机制的运算；验证完成后，将区块通过 gossip 协议广播至网络中；网络中的其他节点接收到新区块后，也会对区块进行验证；最终，提交至区块链，更新状态。

具体来看， Ethereum 系统有共识层、数据层、应用层等核心组件，其交互逻辑如下：



如上图所示， Ethereum 数据由 Transaction Root 和 State Root 组成。 Transaction Root 是所有交易组成的树，包含 From 、 To 、 Data 、 Value 、 Gas Limit 和 Gas Price ； 而 State Root 则是所有账户组成的树，包含 Address 、 Code 、 Storage 、 Balance 和 Nonce 。

总结

以上就是对 Ethereum 核心技术的一些解读，智能合约的引入给区块链的应用带来了更多可能性，但仍有很多安全性、隐私性和效率问题需要考虑。针对复杂的企业级应用场景，联盟链是更好的选择，后续将会对 Hyperledger Fabric 进行详尽的分析，敬请期待！

参考资料

1. COMP7408 Distributed Ledger and Blockchain Technology, *Professor S.M. Yiu, HKU*
2. Udacity Blockchain Developer Nanodegree, *Udacity*
3. 区块链技术与应用，*肖臻，北京大学*
4. 区块链技术进阶与实战，*蔡亮 李启雷 梁秀波，浙江大学 | 趣链科技*
5. Ethereum Architecture, *zastrin*
6. Learn Solidity: Complete Example: Crowd Funding Smart Contract, *TOSHBLOCKS*