以太坊是什么 - 以太坊开发入门指南

2017-11-20 | 2018-07-31 | 以太坊

很多同学已经跃跃欲试投入到区块链开发队伍当中来，可是又感觉无从下手，本文将基于以太坊平台，以通俗的方式介绍以太坊开发中涉及的各晦涩的概念，轻松带大家入门。

写在前面

阅读本文前，你应该大概了解区块链是什么，如果你还不了解，欢迎订阅专栏：区块链技术
指引你从头开始学区块链技术。

以太坊是什么

以太坊（Ethereum）是一个建立在区块链技术之上， 去中心化应用平台。它允许任何人在平台中建立和使用通过区块链技术运行的去中心化应用。

> 对这句话不理解的同学，姑且可以理解为以太坊是区块链里的Android，它是一个开发平台，让我们就可以像基于Android Framework一样基于区块链技术写应用。

在没有以太坊之前，写区块链应用是这样的：拷贝一份比特币代码，然后去改底层代码如加密算法，共识机制，网络协议等等（很多山寨币就是这样，改改就出来一个新币）。
以太坊平台对底层区块链技术进行了封装，让区块链应用开发者可以直接基于以太坊平台进行开发，开发者只要专注于应用本身的开发，从而大大降低了难度。

> 目前围绕以太坊已经形成了一个较为完善的开发生态圈：有社区的支持，有很多开发框架、工具可以选择。

智能合约

什么是智能合约

以太坊上的程序称之为智能合约， 它是代码和数据(状态)的集合。

智能合约可以理解为在区块链上可以自动执行的（由消息驱动的）、以代码形式编写的合同（特殊的交易）。

> 智能合约英文是Smart Contract，和人工智能（ AI:Artificial Intelligence ）的智能没有关系，最早尼克萨博在95年就提出智能合约的概念，它的概念很简单，就是将法律条文写成可执行代码。当时并没有区块链，不过智能合约与区块链最配，我们知道合同都是要一式两份、三或四份，不能控制在某一方手中，这也就是去中心化。

在比特币脚本中，我们讲到过比特币的交易是可以编程的，但是比特币脚本有很多的限制，能够编写的程序也有限，
而以太坊则更加完备（在计算机科学术语中，称它为是“图灵完备的”），让我们就像使用任何高级语言一样来编写几乎可以做任何事情的程序（智能合约）。

智能合约非常适合对信任、安全和持久性要求较高的应用场景，比如：数字货币、数字资产、投票、保险、金融应用、预测市场、产权所有权管理、物联网、点对点交易等等。
目前除数字货币之外，真正落地的应用还不多（就像移动平台刚开始出来一样），相信1到3年内，各种杀手级会慢慢出现。

编程语言：Solidity

智能合约的官方推荐的编程语言是Solidity，文件扩展名以.sol结尾。
Solidity语言和JavaScript很相似，用它来开发合约并编译成以太坊虚拟机字节代码。

> 还有Viper，Serpent，LLL及Bamboo，建议大家还是使用Solidity。
> 更新：Serpent官方已经不再推荐，建议Serpent的用户转换到Viper，他们都是类Python语言。

Browser-Solidity是一个浏览器的Solidity IDE, 大家可以点进去看看，以后我们更多文章介绍Solidity这个语言。

运行环境：EVM

EVM（Ethereum Virtual Machine）以太坊虚拟机是以太坊中智能合约的运行环境。

> Solidity之于EVM，就像之于跟JVM的关系一样，这样大家就容易理解了。
> 以太坊虚拟机是一个隔离的环境，外部无法接触到在EVM内部运行的代码。

而EVM运行在以太坊节点上，当我们把合约部署到以太坊网络上之后，合约就可以在以太坊网络中运行了。

合约的编译

以太坊虚拟机上运行的是合约的字节码形式，需要我们在部署之前先对合约进行编译，可以选择Browser-Solidity Web IDE或solc编译器。

合约的部署

在以太坊上开发应用时，常常要使用到以太坊客户端（钱包）。平时我们在开发中，一般不接触到客户端或钱包的概念，它是什么呢?

以太坊客户端（钱包）

以太坊客户端，其实我们可以把它理解为一个开发者工具，它提供账户管理、挖矿、转账、智能合约的部署和执行等等功能。

> EVM是由以太坊客户端提供的

Geth是典型的开发以太坊时使用的客户端，基于Go语言开发。 Geth提供了一个交互式命令控制台，通过命令控制台中包含了以太坊的各种功能（API）。Geth的使用我们之后会有文章介绍，这里大家先有个概念。

> Geth控制台和Chrome浏览器开发者工具里的面的控制台是类似的，不过Geth控制台是跑在终端里。
> 相对于Geth，Mist则是图形化操作界面的以太坊客户端。

如何部署

智能合约的部署是指把合约字节码发布到区块链上，并使用一个特定的地址来标示这个合约，这个地址称为合约账户。

> 以太坊中有两类账户：
>
> - 外部账户
>   该类账户被私钥控制（由人控制），没有关联任何代码。

> - 合约账户
>   该类账户被它们的合约代码控制且有代码与之关联。
>
> > 和比特币使用UTXO的设计不一样，以太坊使用更为简单的账户概念。
> > 两类账户对于EVM来说是一样的。
>
> 外部账户与合约账户的区别和关系是这样的：一个外部账户可以通过创建和用自己的私钥来对交易进行签名，来发送消息给另一个外部账户或合约账户。
> 在两个外部账户之间传送消息是价值转移的过程。但从外部账户到合约账户的消息会激活合约账户的代码，允许它执行各种动作（比如转移代币，写入内部存储，挖出一个新代币，执行一些运算，创建一个新的合约等等）。
> 只有当外部账户发出指令时，合同账户才会执行相应的操作。

合约部署就是将编译好的合约字节码通过外部账号发送交易的形式部署到以太坊区块链上（由实际矿工出块之后，才真正部署成功）。

运行

合约部署之后，当需要调用这个智能合约的方法时只需要向这个合约账户发送消息（交易）即可，通过消息触发后智能合约的代码就会在EVM中执行了。

Gas

和云计算相似，占用区块链的资源（不管是简单的转账交易，还是合约的部署和执行）同样需要付出相应的费用（天下没有免费的午餐对不对!）。
以太坊上用Gas机制来计费，Gas也可以认为是一个工作量单位，智能合约越复杂（计算步骤的数量和类型，占用的内存等），用来完成运行就需要越多Gas。
任何特定的合约所需的运行合约的Gas数量是固定的，由合约的复杂度决定。
而Gas价格由运行合约的人在提交运行合约请求的时候规定，以确定他愿意为这次交易愿意付出的费用：Gas价格（用以太币计价） * Gas数量。

Gas的目的是限制执行交易所需的工作量，同时为执行支付费用。当EVM执行交易时，Gas将按照特定规则被逐渐消耗，无论执行到什么位置，一旦Gas被耗尽，将会触发异常。当前调用帧所做的所有状态修改都将被回滚， 如果执行结束还有Gas剩余，这些Gas将被返还给发送账户。

> 如果没有这个限制，就会有人写出无法停止（如：死循环）的合约来阻塞网络。

因此实际上（把前面的内容串起来），我们需要一个有以太币余额的外部账户，来发起一个交易（普通交易或部署、运行一个合约），运行时，矿工收取相应的工作量费用。

以太坊网络

有些着急的同学要问了，没有以太币，要怎么进行智能合约的开发？可以选择以下方式：

选择以太坊官网测试网络Testnet

测试网络中，我们可以很容易获得免费的以太币，缺点是需要发很长时间初始化节点。

使用私有链

创建自己的以太币私有测试网络，通常也称为私有链，我们可以用它来作为一个测试环境来开发、调试和测试智能合约。

通过上面提到的Geth很容易就可以创建一个属于自己的测试网络，以太币想挖多少挖多少，也免去了同步正式网络的整个区块链数据。

使用开发者网络(模式)

相比私有链，开发者网络(模式)下，会自动分配一个有大量余额的开发者账户给我们使用。

使用模拟环境

另一个创建测试网络的方法是使用testrpc，testrpc是在本地使用内存模拟的一个以太坊环境，对于开发调试来说，更方便快捷。而且testrpc可以在启动时帮我们创建10个存有资金的测试账户。
进行合约开发时，可以在testrpc中测试通过后，再部署到Geth节点中去。

> 更新：testrpc 现在已经并入到Truffle 开发框架中，现在名字是Ganache CLI。

Dapp：去中心化的应用程序

以太坊社区把基于智能合约的应用称为去中心化的应用程序(Decentralized App)。如果我们把区块链理解为一个不可篡改的数据库，智能合约理解为和数据库打交道的程序，那就很容易理解Dapp了，一个Dapp不单单有智能合约，比如还需要有一个友好的用户界面和其他的东西。

Truffle

Truffle是Dapp开发框架，他可以帮我们处理掉大量无关紧要的小事情，让我们可以迅速开始写代码-编译-部署-测试-打包DApp这个流程。

总结

我们现在来总结一下，以太坊是平台，它让我们方便的使用区块链技术开发去中心化的应用，在这个应用中，使用Solidity来编写和区块链交互的智能合约，合约编写好后之后，我们需要用以太坊客户端用一个有余额的账户去部署及运行合约（使用Truffle框架可以更好的帮助我们做这些事情了）。为了开发方便，我们可以用Geth或testrpc来搭建一个测试网络。

注：本文中为了方便大家理解，对一些概念做了类比，有些严格来不是准确，不过我也认为对于初学者，也没有必要把每一个概念掌握的很细致和准确，学习是一个逐步深入的过程，很多时候我们会发现，过一段后，我们会对同一个东西有不一样的理解。