知乎

深入浅出以太坊：以太坊数据如何存储？

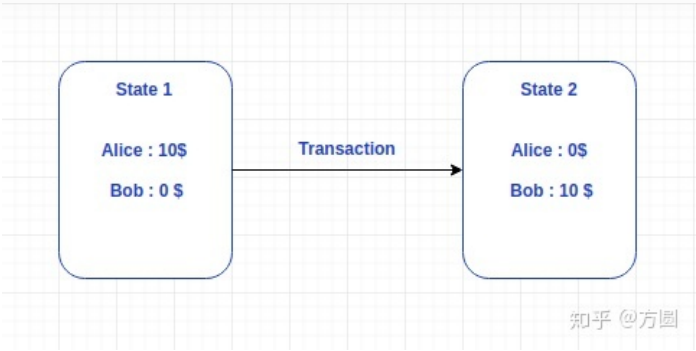
方圆
老码农一枚

本文我们将学习以太坊中的状态和交易是如何存储的以及它与比特币的不同。

本文我们将聚焦以太坊的数据存储层。 我们将介绍区块链“状态”的概念。这里将涵盖Patricia Trie数据结构背后的理论，同时将使用Google的leveldb数据库演示以太坊trie的具体实现。

存储层中

首先，我



由上图可知，我们可以通过执行交易来更改状态。

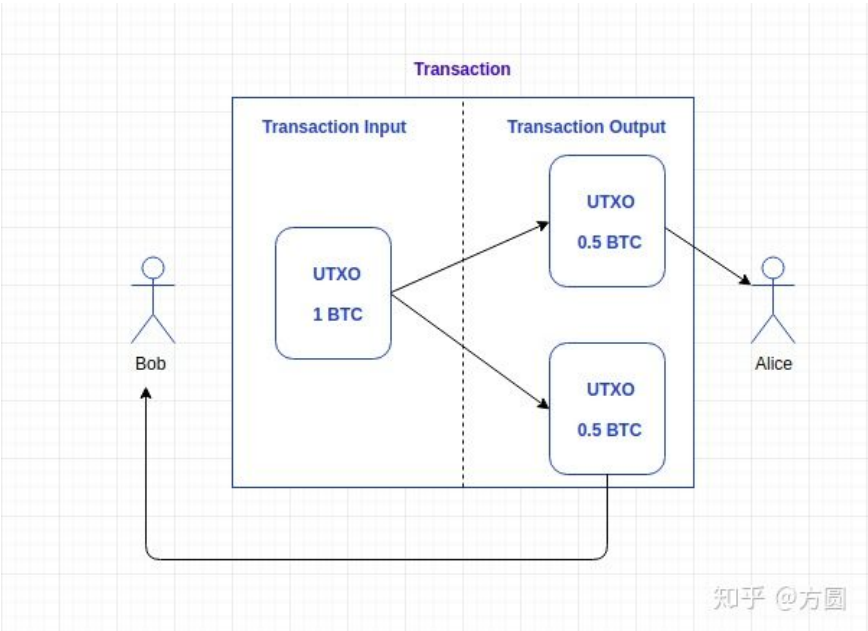
在上面例子中，我们必须跟踪每个账户的余额等细节以及他们在区块链上的交易细节。 下面我们将讲解比特币和以太坊两个不同平台是如何处理这个问题的。

比特币

比特币的“状态”是由其所有未使用交易输出（UTXO）集合表示的。 比特币通过交易来实现价值转移。 更具体地说，比特币用户花费UTXO是通过创建交易并将这些UTXO添加为交易输入来实现的。

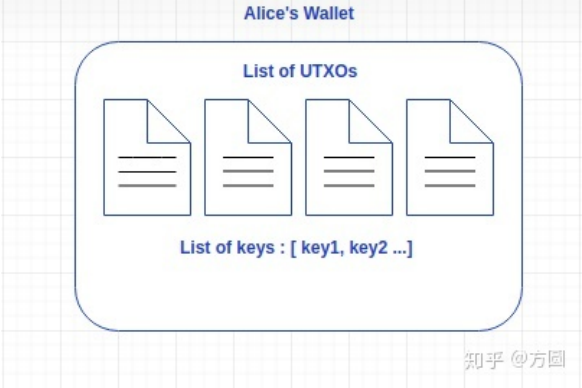
这种UTXO模型使比特币与以太坊不同。 让我们看一些例子来理解其中差异。

首先，比特币UTXO不能被部分花费。 如果一个用户想花0.5比特币，然而他UTXO集合中有1比特币，他必须在发给别人0.5比特币的同时发送给自己0.5比特币来找零。 如果他不发送给自己，那么他将付0.5比特币给矿工。



UTXO 交易

其次，比特币本身不会存储账户余额。 用户只需用私钥签名这些UTXO即可花费比特币。 数字钱包使比特



可视化钱包如何在比特币中工作

比特币的UTXO系统运行良好，部分原因是数字钱包能够帮助完成与交易相关的大多数工作。 包括但不仅限于：

a）处理UTXO

b）存储密钥

c）设定交易费用

d）提供找零地址

e）聚合UTXO（显示可用，待定和总余额）

在UTXO模型中交易类似于纸币。 每个帐户通过计算钱包（对应钱包地址）中所有纸币(UTXO)的和来表示其拥有多少钱。 当我们想要花钱时，需要使用掉包含足够费用的一个或多个UTXO并可能收到找零（新的UTXO）。 每个UTXO只能花费一次，因为一旦UTXO被花费就会从UTXO池中被删除。

总而言之，我们知道：

●比特币区块链不存储账户余额

●比特币钱包持有秘钥

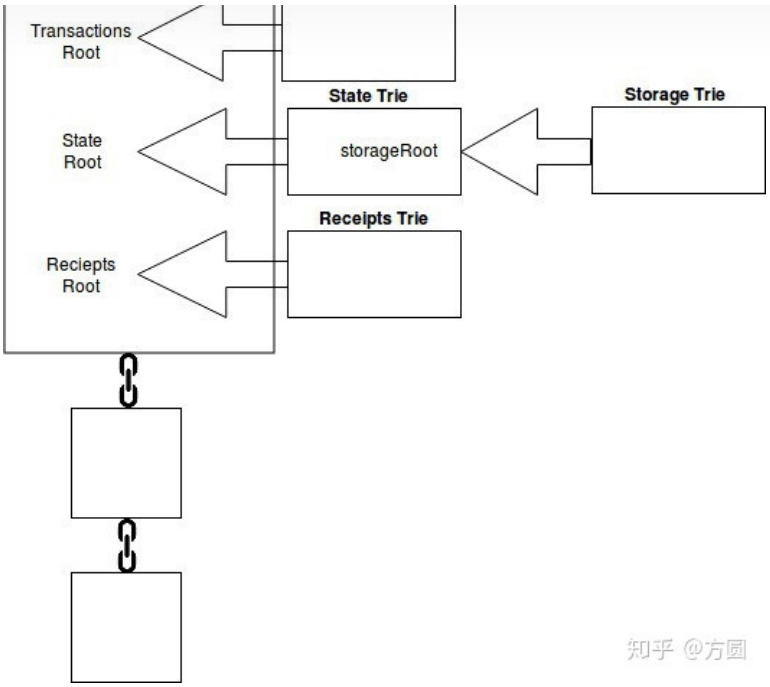
●如果包含在交易中，则会花费整个UTXO（在某些情况下，找零以全新UTXO形式存在）

以太坊

与比特币相反，以太坊通过全局状态管理账户余额等。 以太坊的状态不是一个抽象的概念而是以太坊基础协议的一部分。 正如黄皮书所提到的，以太坊是一种基于交易的“状态”机器，一种可以构建基于所有交易状态机的技术。

与所有其他区块链一样，以太坊区块链起始于它的创世区块。 从0区块的创世状态开始，诸如交易，合约和挖矿等活动将不断改变以太坊区块链的状态。 在以太坊中，例如帐户余额（存储在state trie），每当该帐户有交易发生时，帐户余额的状态就会被改变。

重要的是，帐户余额等数据不会直接存储在以太坊区块链的块中。 只有transaction trie, state trie和receipts trie三者的根节点哈希才直接存储在区块链中。 这在下图中说明。



从上图中，您还会注意到storage trie(保存所有智能合约数据)的根节点哈希实际上指向state trie，而state trie又指向区块链。 我们将尽快聚焦并详细介绍这些内容。

以太坊中有两种截然不同的数据; 永久数据和临时数据。 永久数据的例子是交易，交易被完全确认后将被记录在transaction trie，也将绝不会改变。 临时数据的例子是特定以太坊帐户地址上的余额。 帐户地址的余额存储在state trie中并且每当该特定帐户的交易改变时就会被更改。 在以太坊中，永久数据（如被挖矿的交易）和临时数据（如帐户余额）被单独存储。 以太坊使用trie数据结构来管理数据。

以太坊的记录保存就像银行一样。类似于使用ATM /借记卡， 银行跟踪每张借记卡的金额。当我们需要花钱时，银行会检查其记录以确保我们在确认交易前是否有足够的余额。

UTXO与账户方法的对比

UTXO模型的好处：

●可扩展性 - 由于可以同时处理多个UTXO，因此可以实现并行交易并鼓励可扩展性创新。

●隐私性 - 即使比特币不是一个完全匿名的系统，假设用户每笔交易都使用新地址，那么也能够提供更高级别的隐私。 如果需要更好的隐私性可以考虑更复杂的方案，例如环形签名。

账户/余额模型的好处：

●简单性 - 以太坊选择了更直观的模型，以使开发人员在开发那些需要状态信息或涉及多方的智能合约时更简单。 例如智能合约，它能够直接跟踪状态并能够执行不同的任务。 UTXO的无状态模型会强制交易包含状态信息，这使得合约的设计复杂化。

●效率 - 除简单性外，帐户/余额模型更有效，因为交易时只需要验证发送帐户是否有足够的余额。

账户/余额模型的一个缺点是无法避免双重支付攻击，但通过递增的随机数可以消除这种类型的攻击。 在以太坊中，每个帐户都有一个公共可见的随机数，每次进行交易时，随机数增加一。 这可以防止同一个交易被多次提交。 （注意，这个随机数与以太坊的工作随机数证明不同，后者是一个随机值。）

像计算机
可以从比

深入以太坊trie结构

让我们更深入地了解state trie, storage trie和transaction trie。

独一无二的state trie

在以太坊中只有一个全局state trie。这个全局state trie不断被更新。state trie包含以太坊网络中存在的每个帐户的秘钥键值对。“秘钥”是一个160位标识符（以太坊帐户的地址）。全局state trie中的“值”是通过对以太坊帐户的详细信息进行编码（使用递归长度前缀编码（RLP）方法）来创建的。

账户详细信息如下：

- 随机数

- 余额

- 存储根哈希

- 代码哈希

state trie的根节点（给定时间点的整个state trie的哈希）被用作state trie的安全且唯一的标识符; state trie的根节点由所有内部state trie数据hash产生。

State Trie（Merkle Patricia Trie的leveldb实现）和以太坊块之间的关系

State Trie - 状态trie的根节点的Keccak-256位散列，存储为给定块中的“stateRoot”值。
stateRoot：'0x8c77785e3e9171715dd34117b047dffe44575c32ede59bde39fbf5dc074f2976'

storage trie - 存储合约数据的地方

storage trie是所有合约数据存储的地方。 每个以太坊帐户都有自己的存储空间。 storage trie的根节点的256位hash作为storageRoot值存储在全局state trie中（我们刚刚讨论过）。

每个区块都有一个Transaction trie

每个以太坊块都有自己独立的transaction trie。 一个块包含许多交易。 块中交易的顺序当然由打包块的矿工决定。 transaction trie中特定交易的路径是通过交易在块中的索引（RLP编码）实现的。 已经被确认的区块不会被更新，它包含的交易的位置也不会被改变。 这意味着一旦在块的transaction trie中定位到该交易，就可以一直返回相同的路径来提供检索结果。

以太坊中关于trie的具体例子

以太坊主要的客户端使用两种不同的数据库解决方案来存储他们的trie。 以太坊的Rust客户端Parity使用rocksdb， 而以太坊的Go，C ++和Python客户端都使用leveldb。

Rocksdb超出了这篇文章的范围。 我们先探讨其他3个以太坊客户端是如何使用leveldb。

以太坊和leveldb

LevelDB是开源的键值对数据库，除了数据库主要功能外，还提供对数据的前向和后向迭代，从键到值的有序映射，自定义比较函数和自动压缩功能。使用开源Google压缩/解压缩库“Snappy”可以自动压缩数据。相比追求最大压缩，Snappy的目标是更快的压缩速度。Leveldb提供一种重要的存储和检索机制用于管理以太坊网络的状态。因此，leveldb被以太坊客户端（节点）广泛使用，例如go-ethereum，cpp-ethereum和pyethereum。

虽然trie数据结构的实现可以在磁盘上完成（使用诸如leveldb之类的数据库），但需要注意的是遍历trie和简单查询之间存在的差异。

要了解更多信息，我们必须使用适当的Patricia trie库访问leveldb中的数据。为此，我们需要安装以太坊。

下面是一个易于学习的以太坊教程，用于设置您自己的以太坊专有网络。

如何：创建自己的以太坊区块链。

一旦您建立了以太坊专用网络，就能够执行交易来研究以太坊的“状态”是如何响应网络活动，如交易，合约和挖矿。 如果您无法安装以太坊专用网络，我们提供以太坊专有网络代码示例和屏幕截图。

分析以太

正如我们之前提到的，在以太坊区块链中有许多Merkle Patricia Tries（每个区块都将引用）：

- state trie
- storage trie
- transaction trie
- receipts trie

要在特定块中引用特定的Merkle Patricia Trie，我们需要获取其根哈希作为参考。 以下命令允许我们在创世区块中获取state trie，transaction和receipt trie的根哈希值。

```
web3.eth.getBlock(0).stateRoot
web3.eth.getBlock(0).transactionsRoot
web3.eth.getBlock(0).receiptsRoot
```

注意：如果您想要最新块的根哈希（而不是创世块），请使用以下命令。

```
web3.eth.getBlock(web3.eth.blockNumber).stateRoot
```

安装npm，node，level和ethereumjs

我们将使用nodejs，level和ethereumjs（在Javascript中实现Ethereum的VM）的组合来测试leveldb数据库。 以下命令将进一步准备我们的环境。

```
cd ~
sudo apt-get update
sudo apt-get upgrade
curl -sL https://deb.nodesource.com/setup_9.x | sudo -E bash - sudo apt-get ins
sudo apt-get install nodejs
npm -v
nodejs -v
npm install levelup leveldown rlp merkle-patricia-tree --save
git clone https://github.com/ethereumjs/ethereumjs-vm.git
cd ethereumjs-vm
npm install ethereumjs-account ethereumjs-util --save
```

从这一点开始，运行以下代码将打印以太坊帐户密钥列表（存储在以太坊专用网络的状态根目录中）。 通过代码连接到以太坊的leveldb数据库，进入以太坊的world state（使用区块链中块的stateRoot值），然后访问以太坊专用网络上所有帐户的密钥。

```
//Just importing the requirements
var Trie = require('merkle-patricia-tree/secure');
var levelup = require('levelup');
var leveldown = require('leveldown');
var RLP = require('rlp');
var assert = require('assert');

//Conne
```

```
var db
//Addi
var root = '0x8c77785e3e9171715dd34117b047dffe44575c32ede59bde39fbf5dc074f2976'


//Creating a trie object of the merkle-patricia-tree library
var trie = new Trie(db, root);


//Creating a nodejs stream object so that we can access the data
var stream = trie.createReadStream()


//Turning on the stream (because the node js stream is set to pause by default)
stream.on('data', function (data){
  //printing out the keys of the "state trie"
  console.log(data.key);
});
```

Output for the above code

有趣的是，以太坊中的账户只有在（与该特定账户相关）交易发生后才会添加到state trie。 例如，仅使用“geth account new”创建的新帐户即使在许多区块被生成之后也不会存在于state trie中。 但是，如果针对该账户的交易（一个消耗gas并且包含在已生成的区块中的交易）成功被记录，那么此刻该账户才会出现在state trie。 这是用来防止恶意攻击者不断创建新帐户使state trie膨胀。

解码数据

您会注意到查询leveldb所返回编码后的数据。 这是因为以太坊在与leveldb交互时使用了自己特定的“Modified Merkle Patricia Trie”实现。 以太坊Wiki提供有关以太坊的Modified Merkle Patricia和递归长度前缀（RLP）编码的设计和实现的信息。 简而言之，以太坊已经扩展了trie数据结构。 例如，Modified Merkle Patricia包含一种方法，该方法可以通过使用“扩展”节点来沿着trie快速下降。

在以太坊中，单个Modified Merkle Patricia trie节点是：

●空字符串（称为NULL）

●包含17个元素的数组（称为分支）

●包含2个元素的数组（称为叶子）

●包含2个元素的数组（称为扩展名）

以下示例使用ethereumjs测试。

下面的代码（当提供特定块的状态根以及以太坊帐户地址时）将以人类可读的形式返回该帐户的余额。

以下代码的输出（地址0xaa6b46fa5606826ae8e18faaa6f519064e6130b的帐户余额）

```
//Mozi
//As pe
//Requires the following packages to run as nodejs file https://gist.github.cor


//Getting the requirements
var Trie = require('merkle-patricia-tree/secure');
var levelup = require('levelup');
var leveldown = require('leveldown');
var utils = require('ethereumjs-util');
var BN = utils.BN;
var Account = require('ethereumjs-account');


//Connecting to the leveldb database
var db = levelup(leveldown('/home/timothymccallum/gethDataDir/geth/chaindata')


//Adding the "stateRoot" value from the block so that we can inspect the state
var root = '0x9369577baeb7c4e971ebe76f5d5daddba44c2aa42193248245cf686d20a73028


//Creating a trie object of the merkle-patricia-tree library
var trie = new Trie(db, root);


var address = '0xccc6b46fa5606826ce8c18fece6f519064e6130b';
trie.get(address, function (err, raw) {
    if (err) return cb(err)
    //Using ethereumjs-account to create an instance of an account
    var account = new Account(raw)
    console.log('Account Address: ' + address);
    //Using ethereumjs-util to decode and present the account balance
    console.log('Balance: ' + (new BN(account.balance)).toString());
})
```

结论

我们已经在上面证明了以太坊有能力管理其“状态”。 这种巧妙的前期设计具有以下许多优点。

移动性

鉴于移动设备和物联网（IoT）设备现在无处不在，电子商务的未来取决于安全，健壮和快速的移动应用。

虽然我们知道移动设备在不断进步，但同时区块链大小也变得更大，因此在日常移动设备上存储整个区块链是不切实际的。

快速且安全

以太坊的state trie的设计及其对Modified Merkle Patricia Trie的使用在这个领域提供了许多机会。在以太坊中的trie上执行的每个功能（增加，更新和删除）都使用确定性加密哈希。 此外，trie根节点的唯一加密散列可以用作trie未被篡改的证据。

例如，对任何级别的trie数据的任何更改（例如增加leveldb数据库中的帐户余额）都将彻底更改根哈希。 此加密功能为轻客户端（不存储整个区块链的设备）提供了快速可靠地查询区块链的机会，即账户“0x ... 4857”是否有足够的资金在区块高度“5044866”去完成此次购买？

“Merkle证明的大小复杂度是存储数据量的对数。 这意味着，如果一个节点从受信任的源接收一个状态根，那么他就只需下载一个几千字节的证明数据就能够完全确定地知道该树任何信息的有效性。”

支出限制

以太坊白皮书中提到的一个有趣的想法是储蓄账户的概念。 使用存储账户，两个用户（可能是丈夫和妻子，或商业伙伴）每人可以每天提取1%的帐户总余额。 这个想法只在白皮书的“进一步的应用程序”章节中提到，但它引起了人们的兴趣。因为理论上它可以作为以太坊基础协议层的一部分实现。 您可能会在本文开头回顾我们关于比特币UTXO的讨论。 UTXO对区块链数据不可见，比特币区块链实际上并不存储用户账户余额。 出于这个原因，比特币的基本协议层不太可能（或者可能无法）实现任何类型的每日支出限制。

消费者信心

随着这个领域的不断努力，我们将看到轻量级客户端的大量发展。 更具体地说，安全，健壮和快速的移动应用程序可以与区块链技术交互。

想要在电子商务领域成功实施区块链必须提高速度，安全性和可用性。 通过巧妙设计来提供一个卓越的可用性，安全性和性能区块链，总将有可能提高消费者信心并增加主流使用率。

原文地址：

本文由左文康/方圆翻译，有希望加入BFTF区块链技术联盟的小伙伴可以私信我。

发布于 2018-09-25 15:15

区块链(Blockchain)

大

不管是以太坊还是V神，市场给予的情绪是乐观的，但是自2018下半年之后，以太坊的市场表现迟迟提振不起来，加密货币世界格局真的会被改变吗？打开这篇文章，说说你看好以太坊吗：

以太坊为什么给人一种“日薄西山”的感觉？
okex.me/academy/zh/why-...

赞