加餐：第一周课程学习补充

2018-01-14 14:08:49

Lesson 1 补充学习笔记

by Ellie助教

1.Solidity 数据类型

http://solidity.readthedocs.io/en/develop/types.html#value-types

2.单位系统 Address 类型， Block类型，msg 类型

http://solidity.readthedocs.io/en/develop/units-and-global-variables.html#

3.Solidity constant 对变量的影响

Constant Variable in doc

4.当你部署合约的时候，你其实干了什么

智能合约可以简单的理解为一段可执行的程序片段，具体的代码经过solidity编写之后，发布到区块链上。而以太坊的智能合约也可以理解为一个特殊的交易（包括可执行代码的），被发送出去后会被矿工打包记录在某一个区块中，当需要调用这个智能合约的方法时只需要向这个智能合约的地址发送一笔交易即可。 因为触发的条件和打钱地址都已经被编写在代码里，存储在区块链上，所以可以最大程度的排除人为因素的干扰。

回到首页

5.(

发评论

reference:https://zhuanlan.zhihu.com/p/24012669

当你激活一个智能合约的时候，你在要求整个网络内的每个矿工个体分别执行里面的运算。这会花费他们的时间和精力，Gas是你为这项服务向矿工们支付的机制。 报酬是小额的以太币，想要运行智能合约的人的需要支付报酬来使合约工作。类似于投放一个硬币到自动唱机里。 付款款项（单位以太币）= Gas数量（单位Gas） x Gas价格（单位以太币 / Gas）

5.1 Gas数量

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。类比自动唱机，歌曲的时间越长，音量越大，让它工作你需要支付的则越多。

5.2 Gas 价格

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价格由想运行合约的人规定，在他们提交运行合约请求的时候（有点类似于比特币的交易费）。每个矿工会根据Gas的价格的高低来决定他们是否想作为区块的一部分去运行此合约。如果你希望矿工运行你的合约，你最好提供高一点的Gas价格。在某种程度是这是一场基于合约运行有多愿意付费驱动下的竞价。

5.3 为什么需要Gas

让智能合约花费Gas/以太币/钱可以防止人们随意激活合约， 解决了垃圾交易以及相关问题，如果运行智能合约免费，此类问题会发生。

回到首页

Lesson 2 补充学习笔记

by蔡至诚助教

1. 异常处理(revert, assert和require)

http://solidity.readthedocs.io/en/develop/control-structures.html#error-handling-assert-require-revert-and-exceptions

回滚机制（reverting）是为了确保transaction的原子性，主要分为require样式和assert样式。

> Internally, Solidity performs a revert operation (instruction 0xfd) for a require-style exception and executes an invalid operation (instruction 0xfe) to throw an assert-style exception. In both cases, this causes the EVM to revert all changes made to the state. The reason for reverting is that there is no safe way to continue execution, because an expected effect did not occur. Because we want to retain the atomicity of transactions, the safest thing to do is to revert all changes and make the whole transaction (or at least call) without effect. Note that assert-style exceptions consume all gas available to the call, while require-style exceptions will not

回到首页

> consume any gas starting f
> rom the Metropolis release.

assert用于检查程序运行条件是否满足（例如数组访问是否越界），require用于检查用户输入或外部调用返回值。

> The assert function should
> only be used to test for int
> ernal errors, and to check i
> nvariants. The require funct
> ion should be used to ensur
> e valid conditions, such as i
> nputs, or contract state var
> iables are met, or to validat
> e return values from calls t
> o external contracts.

异常一般会由子调用向上传递。但是send和底层调用call、delegatecall、callcode除外，它们只会返回false（注意：如果被调用账户不存在，call、delegatecall和callcode返回true）。

补充阅读：

- 对于revert、assert和require的使用：https://medium.com/blockchannel/the-use-of-revert-assert-and-require-in-solidity-and-the-new-revert-opcode-in-the-evm-1a3a7990e06e
- REVERT instruction： https://github.com/ethereum/EIPs/pull/206
- 拜占庭分叉后yellow book变化： https://github.com/ethereum/yellowpaper/issues/229

2. 数据位置(storage、memory、calldata

回到首页

http://solidity.readthedocs.io/en/develop/types.html#data-location

Summary

- Forced data location:
  parameters (not return) of external functions: calldata
  state variables: storage
- Default data location:
  parameters (also return) of functions: memory
  all other local variables: storage

补充阅读：

- Solidity的数据位置特性深入详解 http://me.tryblockchain.org/solidity-data-location.html

storage转换为storage: 引用
memory赋值给storage(状态变量): 拷贝
memory赋值给storage(局部变量): 报错
storage转为memory: 拷贝
memory转为memory: 引用

3. 其他

(1). 函数四种可见性(external、public、internal、private)

回到首页

https://solidity.readthedocs.io/en/develop/contracts.html#visibility-and-getters

补充阅读：

- 区块链编程语言Solidity语言函数可见性深入详解 http://www.jianshu.com/p/c3e3ccb466ec

(2). 判断函数调用者身份时可以采用函数修饰符(modifier)的方式

https://solidity.readthedocs.io/en/latest/contracts.html#function-modifiers

(by 顺达、至诚)

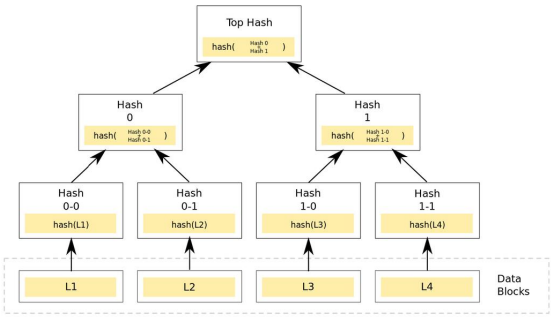
Ethereum的Merkle拓展

by刘虹男助教

Merkle Tree是存储hash值的一棵树。Merkle树的叶子是数据块的hash值，非叶节点是其对应子节点串联字符串的hash。

在最底层，把数据分成小的数据块，有相应地哈希和它对应。往上走，把相邻的两个哈希合并成一个字符串，然后运算这个字符串的哈希，这样每两个哈希就结婚生子，得到了一个”子哈希“。如果最底层的哈希总数是单数，那到最后必然出现一个单身哈希，这种情况就直接对它进行哈希运算，所以也能得到它的子哈希。于是往上推，依然是一样的方式，可以得到数目更少的新一级哈希，最终必然形成一棵倒挂的树，到了树根的这个位置，就剩下一个根哈希了，我们把它叫做 Merkle Root。

回到首页



参考文章

http://www.cnblogs.com/fengzhiwu/p/5524324.html

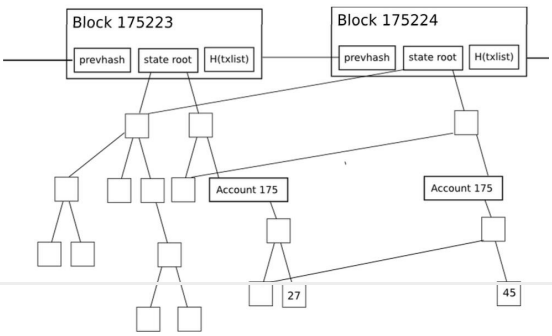
Merkle Patricia Trees 在 Ethereum 的应用

每个以太坊区块头不是包括一个Merkle树，而是为三种对象设计的三棵树，称之为Merkle Patricia tree，包括：

交易tx root
收据receipt root
状态state root

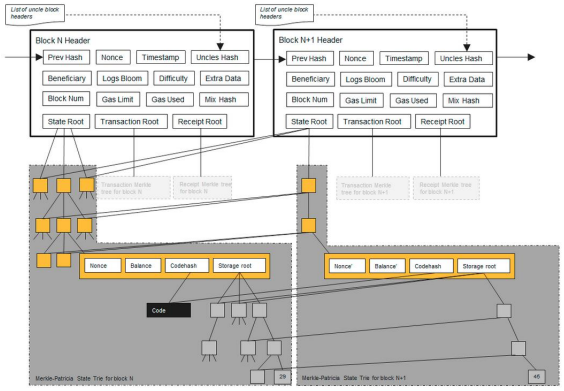
其中，状态树的情况会更复杂些。因为账户状态信息在Ethereum中会经常变更，新的账户会被插入，合约中的storage也会经常变动，因此需要一种数据结构，能够在插入，删除，更新等操作之后，快速计算出新的root值，不需要重新计算整棵树。因此Ethereum中的状态树基本上包含了一个键值映射，其中的键是地址（账户地址&合约地址），而值包括nonce,balance,codeHash以及storageRoot（storageRoot是Merkle树根，存储合约中的storage数据）。

state root (仅包含key) 结构图：



回到首页

state root (包含key – value) 结构图



参考文章

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

https://github.com/ethereum/wiki/wiki/Patricia-Tree

http://gavwood.com/Paper.pdf

合约的storage读取

合约中storage状态存储在上面介绍的storageRoot的Merkle树中，首先通过合约地址为键，在stateRoot状态树中进行搜索，找到对应value中的storageRoot，接着在storageRoot树中对合约storage进行键值映射查找。

在合约中的所有storage状态，是从第一个按序进行索引的，一个索引键占用256个字节。

对于合约中不同类型的storage，键的计算方式可以参考文章

https://medium.com/aigang-network/how-to-read-ethereum-contract-storage-44252c8af925

最后，通过使用eth_getStorageAt方法，传入合约地址，storage的键，区块号（或者latest, earliest,pending三种状态的区块）进行storage的value查询，参考文章

https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getstorageat

回到首页

115 0

0条评论 最新

已加载全部

回到首页