**课程报告**

|  |  |
| --- | --- |
| **课程名称：** | **企业软件项目实训** |
| **学生姓名：** | **李克奉** |
| **学生学号：** | **201630664789** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2018-2019第二学期** |

**软件学院**

**2019年6月**

1. 区块链技术原理**；**

区块链”技术最初是由一位化名中本聪的人为比特币（一种数字货币）而设计出的一种特殊的数据库技术，它基于密码学中的椭圆曲线数字签名算法（ECDSA）来实现去中心化的P2P系统设计。但区块链的作用不仅仅局限在比特币上。现在，人们在使用“区块链”这个词时，有的时候是指数据结构，有时是指数据库，有时则是指数据库技术，但无论是哪种含义，都和比特币没有必然的联系。

从数据的角度来看：区块链是一种分布式数据库（或称为分布式共享总账，Distributed Shared Ledger），这里的“分布式”不仅体现为数据的分布式存储，也体现为数据的分布式记录（即由系统参与者来集体维护）。简单的说，区块链能实现全球数据信息的分布式记录（可以由系统参与者集体记录，而非由一个中心化的机构集中记录）与分布式存储（可以存储在所有参与记录数据的节点中，而非集中存储于中心化的机构节点中）。

从效果的角度来看：区块链可以生成一套记录时间先后的、不可篡改的、可信任的数据库，这套数据库是去中心化存储且数据安全能够得到有效保证的。

区块链是一种把区块以链的方式组合在一起的数据结构，它适合存储简单的、有先后关系的、能在系统内验证的数据，用密码学保证了数据的不可篡改和不可伪造。它能够使参与者对全网交易记录的事件顺序和当前状态建立共识。

如今的区块链技术概括起来是指通过去中心化和去信任的方式集体维护一个可靠数据库的技术。其实，区块链技术并不是一种单一的、全新的技术，而是多种现有技术（如加密算法、P2P文件传输等）整合的结果，这些技术与数据库巧妙地组合在一起，形成了一种新的数据记录、传递、存储与呈现的方式。简单的说，区块链技术就是一种大家共同参与记录信息、存储信息的技术。过去，人们将数据记录、存储的工作交给中心化的机构来完成，而区块链技术则让系统中的每一个人都可以参与数据的记录、存储。区块链技术在没有中央控制点的分布式对等网络下，使用分布式集体运作的方法，构建了一个P2P的自组织网络。通过复杂的校验机制，区块链数据库能够保持完整性、连续性和一致性，即使部分参与人作假也无法改变区块链的完整性，更无法篡改区块链中的数据。

区块链技术涉及的关键点包括：去中心化（Decentralized）、去信任（Trustless）、集体维护（Collectively maintain）、可靠数据库（ReliableDatabase）、时间戳（Time stamp）、非对称加密（AsymmetricCryptography）等。可以说，区块链技术的本质是一种互联网协议。

**区块链有四大核心技术**

**区块+链**

区块链将数据库的结构进行创新，把数据分成不同的区块，每个区块通过特定的信息链接到上一区块的后面，前后顺连来呈现一套完整的数据，这也是“区块链”这三个字的来源。每一个区块上记录的交易是上一个区块形成之后、该区块被创建前发生的所有价值交换活动，这个特点保证了数据库的完整性。在绝大多数情况下，一旦新区块完成后被加入到区块链的最后，则此区块的数据记录就再也不能改变或删除。这个特点保证了数据库的严谨性，即无法被篡改。区块链就是区块以链的方式组合在一起，以这种方式形成的数据库我们称之为区块链数据库。区块链是系统内所有节点共享的交易数据库，这些节点基于价值交换协议参与到区块链的网络中来。

由于每一个区块的块头都包含了前一个区块的交易信息压缩值，这就使得从创世块（第一个区块）到当前区块连接在一起形成了一条长链。由于如果不知道前一区块的“交易缩影”值，就没有办法生成当前区块，因此每个区块必定按时间顺序跟随在前一个区块之后。这种所有区块包含前一个区块引用的结构让现存的区块集合形成了一条数据长链。“区块+链”的结构为我们提供了一个数据库的完整历史。从第一个区块开始，到最新产生的区块为止，区块链上存储了系统全部的历史数据。区块链为我们提供了数据库内每一笔数据的查找功能。区块链上的每一条交易数据，都可以通过“区块链”的结构追本溯源，一笔一笔进行验证。区块+链=时间戳，这是区块链数据库的最大创新点。区块链数据库让全网的记录者在每一个区块中都盖上一个时间戳来记账，表示这个信息是这个时间写入的，形成了一个不可篡改、不可伪造的数据库。

#### 分布式结构——开源的、去中心化的协议

#### 中心化的体系中，数据都是集中记录并存储于中央电脑上。区块链让每一个参与数据交易的节点都记录并存储下所有的数据。区块链构建一整套协议机制，让全网每一个节点在参与记录的同时也来验证其他节点记录结果的正确性。只有当全网大部分节点（或甚至所有节点）都同时认为这个记录正确时，或者所有参与记录的节点都比对结果一致通过后，记录的真实性才能得到全网认可，记录数据才允许被写入区块中。区块链构建一个分布式结构的网络系统，让数据库中的所有数据都实时更新并存放于所有参与记录的网络节点中。这样即使部分节点损坏或被黑客攻击，也不会影响整个数据库的数据记录与信息更新。

#### 区块链根据系统确定的开源的、去中心化的协议，构建了一个分布式的结构体系，让价值交换的信息通过分布式传播发送给全网，通过分布式记账确定信息数据内容，盖上时间戳后生成区块数据，再通过分布式传播发送给各个节点，实现分布式存储。

#### 区块链中每一笔新交易的传播都采用分布式的结构，根据P2P网络层协议，消息由单个节点被直接发送给全网其他所有的节点。

#### 区块链技术让数据库中的所有数据均存储于系统所有的电脑节点中，并实时更新。完全去中心化的结构设置使数据能实时记录，并在每一个参与数据存储的网络节点中更新，这就极大的提高了数据库的安全性。

#### 通过分布式记账、分布式传播、分布式存储，系统内的数据存储、交易验证、信息传输过程全部都是去中心化的。

#### 非对称加密算法

在“加密”和“解密”的过程中分别使用两个密码，两个密码具有非对称的特点。解密时的密码（在区块链中被称为“私钥”）是只有信息拥有者才知道的，被加密过的信息只有拥有相应私钥的人才能够解密（信息的安全性）。加密时的密码（在区块链中被称为“公钥”）是公开全网可见的，所有人都可以用自己的公钥来加密一段信息（信息的真实性）

区块链系统内，所有权验证机制的基础是非对称加密算法。在区块链系统的交易中，非对称密钥的基本使用场景有两种：1、公钥对交易信息加密，私钥对交易信息解密。私钥持有人解密后，可以使用收到的价值。2、私钥对信息签名，公钥验证签名。通过公钥签名验证的信息确认为私钥持有人发出。

从信任的角度来看，区块链实际上是数学方法解决信任问题的产物。区块链技术中，所有的规则事先都以算法程序的形式表述出来，人们完全不需要知道交易的对手方是“君子”还是“小人”，更不需要求助中心化的第三方机构来进行交易背书，而只需要信任数学算法就可以建立互信。区块链技术的背后，实质上是算法在为人们创造信用，达成共识。

#### 脚本

脚本可以理解为一种可编程的智能合约。在一个去中心化的环境下，所有的协议都需要提前取得共识，那脚本的引入就显得不可或缺了。有了脚本之后，区块链技术就会使系统有机会去处理一些无法预见到的交易模式，保证了这一技术在未来的应用中不会过时，增加了技术的实用性。

一个脚本本质上是众多指令的列表，这些指令记录在每一次的价值交换活动中，价值交换活动的接收者（价值的持有人）如何获得这些价值，以及花费掉自己曾收到的留存价值需要满足哪些附加条件。通常，发送价值到目标地址的脚本，要求价值的持有人提供以下两个条件，才能使用自己之前收到的价值：一个公钥，以及一个签名（证明价值的持有者拥有与上述公钥相对应的私钥）。

脚本的神奇之处在于，它具有可编程性：（1）它可以灵活改变花费掉留存价值的条件，例如脚本系统可能会同时要求两个私钥、或几个私钥、或无需任何私钥等；（2）它可以灵活的在发送价值时附加一些价值再转移的条件，例如脚本系统可以约定这一笔发送出去的价 值以后只能用于支付中信证券的手续费、或支付给政府等。

1. **联盟链和公有链的异同；**

区块链大致可以分为公有链（Public Blockchain）、私有链（Private Blockchain）以及联盟链（Consortium Blockchain）三大类。

其中去中心化程度最高的是公有链。这种以比特币以及以太坊为代表的公有区块链，不受第三方机构控制，世界上所有的人都可读取链上的数据记录、参与交易以及竞争新区块的记账权等。程序开发者无权干涉用户，各参与者（即节点）可自由加入以及退出网络，并按照意愿进行相关操作。

私有区块链则完全相反，该网络的写入权限由某个组织或者机构全权控制，数据读取权限受组织规定，要么对外开放、要么具有一定程度的访问限制。简单来说，可以将其理解为一个弱中心化或者多中心化的系统。由于参与节点具有严格限制且少；与公有链相比，私有链达成共识的时间相对较短、交易速度更快、效率更高、成本更低。不过这种类型的区块链更适合于特定机构内部使用，比如Linux基金会、R3CEVCorda平台以及Gem Health网络的超级账本项目。

而联盟链则是介于公有链以及私有链之间的区块链，可实现“部分去中心化”。链上各个节点通常有与之相对应的实体机构或者组织；参与者通过授权加入网络并组成利益相关联盟，共同维护区块链运行。从某种程度上来说，联盟链也属于私有链的范畴，只是私有化程度有所不同而已。为此其同样具有成本较低、效率较高的特点，适用于不同实体间的交易、结算等B2B交易。

1. **信任链是如何建立的**

信任链，或称数字证书链，是一连串的[数字证书](https://zh.wikipedia.org/wiki/%E9%9B%BB%E5%AD%90%E8%AD%89%E6%9B%B8" \o "数字证书)，由[根证书](https://zh.wikipedia.org/wiki/%E6%A0%B9%E8%AF%81%E4%B9%A6)为起点，透过[层层信任](https://zh.wikipedia.org/wiki/%E4%BC%A0%E9%80%92%E5%85%B3%E7%B3%BB)，使[终端实体证书](https://zh.wikipedia.org/wiki/%E5%85%AC%E9%96%8B%E9%87%91%E9%91%B0%E8%AA%8D%E8%AD%89#%E7%B5%82%E7%AB%AF%E5%AF%A6%E9%AB%94%E8%AD%89%E6%9B%B8)的持有者可以获得转授的信任，[以证明身份](https://zh.wikipedia.org/wiki/%E8%BA%AB%E5%88%86%E6%A0%87%E8%AF%86%E6%96%B9%E5%BC%8F)。基于[信息安全](https://zh.wikipedia.org/wiki/%E4%BF%A1%E6%81%AF%E5%AE%89%E5%85%A8)的考虑，在进行电子商务或使用政府服务时，交易的另一方用户，以根证书为基础，凭借对签发机构的信任，相信当时持有信任链终端的证书持有者确为其人，并透过[公开密钥加密](https://zh.wikipedia.org/wiki/%E5%85%AC%E5%BC%80%E5%AF%86%E9%92%A5%E5%8A%A0%E5%AF%86)确保通信[保密](https://zh.wikipedia.org/wiki/%E4%BF%9D%E5%AF%86%E6%80%A7)、透过[数字签名](https://zh.wikipedia.org/wiki/%E6%95%B8%E4%BD%8D%E7%B0%BD%E7%AB%A0)确保[内容无误](https://zh.wikipedia.org/wiki/%E6%95%B0%E6%8D%AE%E5%AE%8C%E6%95%B4%E6%80%A7)、以及保证对方[无法抵赖](https://zh.wikipedia.org/w/index.php?title=%E7%84%A1%E6%B3%95%E6%8A%B5%E8%B3%B4&action=edit&redlink=1)。

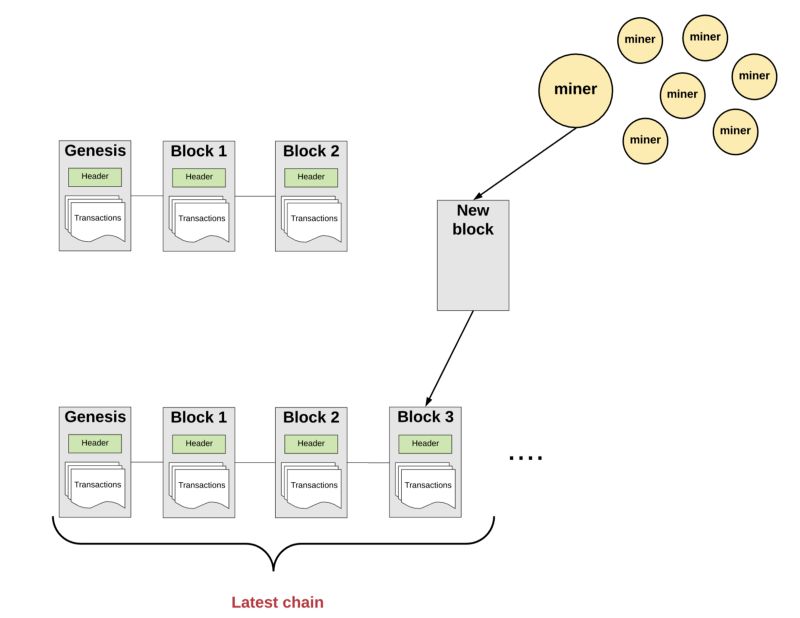
区块链定义了一种协议，允许两个人在互联网上以“点对点”的方式相互交易。当您在区块链上从一个帐户向另一个帐户进行数字传输时，您将信任底层的区块链系统，以使它能够传输并确保发送者的真实性和货币有效性。

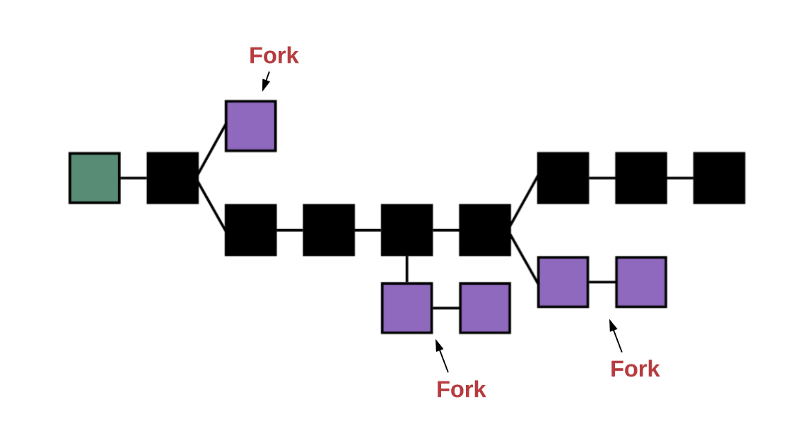
区块链有一个共享的分类账，它告诉我们系统状态的绝对真相。它利用数学、经济学和博弈论来激励系统中的所有各方达成“共识”，或者在这个分类账上达成一致。

以比特币为例。比特币协议有一种称为“工作证明”的共识算法，该算法将系统维系在一起。为了在两个消费者之间达成交易，算法要求一组节点(称为“矿工”)通过解决复杂的算法问题来竞争验证交易。换句话说，比特币“在经济上激励”矿商购买和使用计算能力来解决复杂的问题。这些经济激励措施包括:

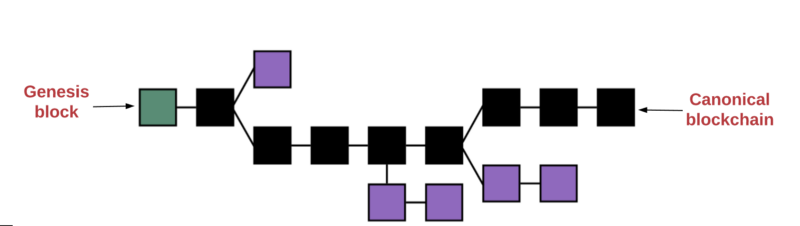
* 矿商赚取交易费，用户支付交易费用，以及矿工们为了成功地解开这个难题而赚取新的比特币。
* 由于这些经济刺激，矿工们不断地观察网络，以便他们能收集一套新的交易，以适应新的“街区”。然后他们用他们的计算资源来解决复杂的算法，以“证明”他们做了一些工作。

第一个解决该算法的矿商将证明和新的块(以及它的所有事务)添加到区块链并将其广播到网络。在这一点上，网络上的每个人都同步最新的区块链，因为这是每个人都相信的“真理”。



由于矿工们都在竞争运行计算，所以有时多个块同时得到解决。这就产生了多个链的“叉”: 

当有这样的分叉时，网络的“规范”链是“最长”的，这是大多数矿工信任并继续工作的。



以这种方式添加到区块链的每一个新的块都给系统增加了更多的安全性，因为想要创建覆盖历史的新块的攻击者需要比网络中其他任何人更快地解决这个问题。这几乎是不可能做到的，因此不可能逆转工程师或更改这些块中的数据。这就是为什么用户信任继续信任系统的原因。

因此，当我们在区块链进行交易时，我们正在锚定我们对那些放弃他们的资源去做一些工作以确保没有双重支出的矿工们的信任。

当然，即使机器的一致性是完美的，我们也不能保证百分之百的几率在维护网络信任的其他重要方面达成共识。例如，当底层网络需要升级、改进或修复时，我们需要一些方法来信任网络及其所有成员可以适当地处理这些变化。

1. **链式存储和MPT存储**

**链式存储**

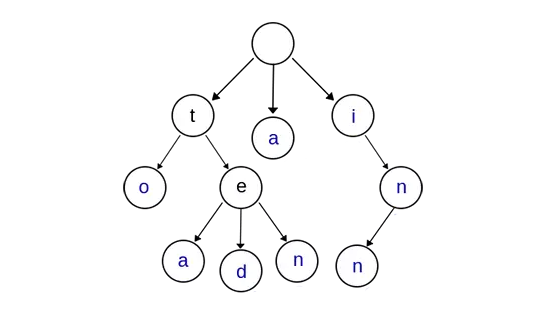
链式存储结构，又叫链接存储结构。在计算机中用一组任意的[存储单元](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%8D%95%E5%85%83/8727749)存储线性表的[数据元素](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0/715313)(这组存储单元可以是连续的,也可以是不连续的)。它不要求逻辑上相邻的元素在物理位置上也相邻.因此它没有[顺序存储结构](https://baike.baidu.com/item/%E9%A1%BA%E5%BA%8F%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84)所具有的弱点,但也同时失去了[顺序表](https://baike.baidu.com/item/%E9%A1%BA%E5%BA%8F%E8%A1%A8)可[随机存取](https://baike.baidu.com/item/%E9%9A%8F%E6%9C%BA%E5%AD%98%E5%8F%96)的优点.

一般在计算机的硬盘中，文件都是链式存储的。我们知道，多个[扇区](https://baike.baidu.com/item/%E6%89%87%E5%8C%BA/3642285)组成一个[簇](https://baike.baidu.com/item/%E7%B0%87/13014767)，簇是计算机存储数据的基本单位。而一个文件是存储在多个在空间上也许并不相连的簇中的。这就是链式存储。但是为了能够读取出这个文件，计算机会在该文件第一部分的尾部写上第二部分所在的簇号。第二部分的尾部又写上第三部分，以此类推，最后一部分写上一段代码，表示这是该文件的最后一部分。值得一提的是，高簇号在后。（如代码所示的1234实为簇3412）文件所占簇可认为是随机分配的。

**MPT存储**

MPT (Merkle Patricia Tries) 是[以太坊](https://www.tuoniaox.com/" \t "_blank)存储数据的核心数据结构，它是由 Merkle Tree 和 Patricia Tree 结合的一种树形结构，理解 [MPT](https://www.tuoniaox.com/news/p-343588.html) 有助于我们更好的理解以太坊的数据存储。在了解 MPT 数据结构之前，我们需要先来看看基本的 Tree 结构和 Merkle Tree、Patricia Tree。

Trie 树，又称前缀树或字典树，是一种有序树，用于保存关联数组，其中的键通常是字符串。一个节点的所有子孙都有相同的前缀，也就是这个节点对应的字符串，而根节点对应空字符串。



上图是一棵 Trie 树，表示了字符串集合{“a”, “to”, “tea”, “ted”, “ten”, “i”, “in”, “inn”} ，从上图中我们可以看出Trie树的特点：

1．根节点不包含字符，除根节点外的每一个子节点都包含一个字符。

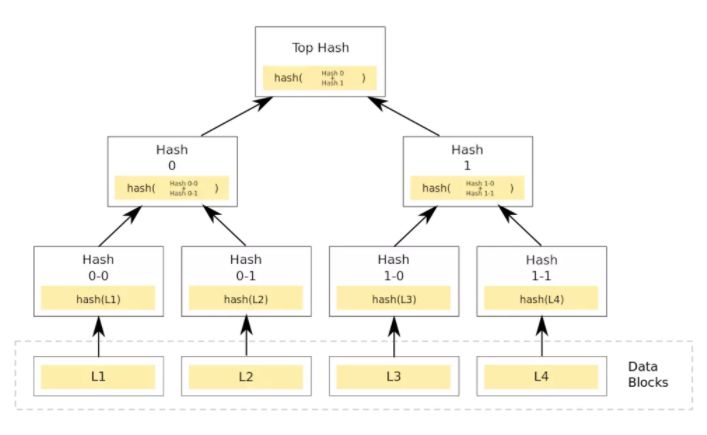
2. 从根节点到某一个节点，路径上经过的字符连接起来，为该节点对应的字符串。

3. 每个节点的所有子节点包含的字符互不相同。

但是从上面的结构也可以看出一个问题：高度不可控。所以就有了 Patricia 树 (压缩前缀树)，后面会介绍到。

Merkle树，也被称为 Hash Tree，中文名称：默克尔树，主要用于数据集较大时的文件校验。其主要特点为：

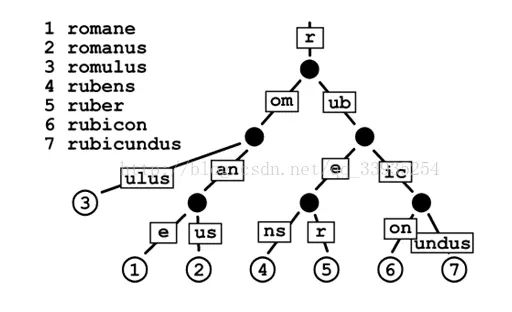
* 叶[节点](https://www.tuoniaox.com/news/p-347359.html" \t "_blank)存储着数据块的 Hash（如：文件块、一段数据集）
* 非叶子节点 (包括中间节点和根节点) 存储着对应子节点 Hash 值串联字符串之后的 Hash 值。



从上图中可以看出：

* 在最底层，和哈希列表一样，我们把数据分成小的数据块，有相应地哈希和它对应；
* 往上走，并不是直接去运算根哈希，而是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再往上推，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，这一代就剩下一个根哈希了，我们把它叫做 Merkle Root。

Patricia 树，或称 Patricia trie，或 crit bit tree，压缩前缀树，是一种更节省空间的 Trie。对于基数树的每个节点，如果该节点是唯一的儿子的话，就和父节点合并。

MPT (Merkle Patricia Tree)

上面我们介绍了Merkle Tree和Patricia Tree，而MPT（Merkle Patricia Tree），顾名思义就是这两者的结合。MTP树种的节点包含空节点、叶子节点、扩展节点和分支节点。

Nibble：它是 key 的基本单元，是一个四元组（四个bit位的组合例如二进制表达的 0010 就是一个四元组）

\*\*空节点\*\*\*\*：简单的表示空，在代码中是一个空串。

叶子节点 (leaf)：只有两个元素，分别为key和value，表示为[key,value]的一个键值对，其中key是key的一种特殊十六进制编码，value是value的RLP编码。

扩展节点 (extension)：也是 [key，value] 的一个键值对，但是这里的value是其他节点的hash值，这个hash可以被用来查询数据库中的节点。也就是说通过hash链接到其他节点。

分支节点 (branch)：分支节点有17个元素，回到 Nibble，四元组是 key 的基本单元，四元组最多有16个值。所以前16个必将落入到在其遍历中的键的十六个可能的半字节值中的每一个。第17个是存储那些在当前结点结束了的节点(例如， 有三个 key,分别是 (abc ,abd, ab) 第17个字段储存了ab节点的值)

这里还有一些知识点需要了解的，为了将 MPT 树存储到数据库中，同时还可以把MPT树从数据库中恢复出来，对于 Extension 和 Leaf 的节点类型做了特殊的定义：如果是一个扩展节点，那么前缀为0，这个0加在 key 前面。如果是一个叶子节点，那么前缀就是1。

同时对 key 的长度就奇偶类型也做了设定，如果是奇数长度则标示1，如果是偶数长度则标示0。

以太坊的每一个区块头，并非只包含一棵 MPT 树，而是包含了三棵 MPT 树，分别对应了四种对象：

State Trie 区块头中的状态树

* key => sha3(以太坊账户地址address)
* value => rlp(账号内容信息account)

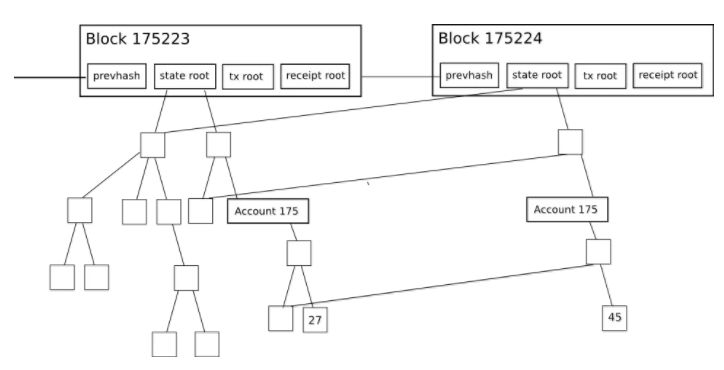
Transactions Trie 区块头中的交易树

* key => rlp(交易的偏移量 transaction index)
* 每个块都有各自的交易树，且不可更改

Receipts Trie 区块头中的收据树

* key = rlp(交易的偏移量 transaction index)
* 每个块都有各自的交易树，且不可更改

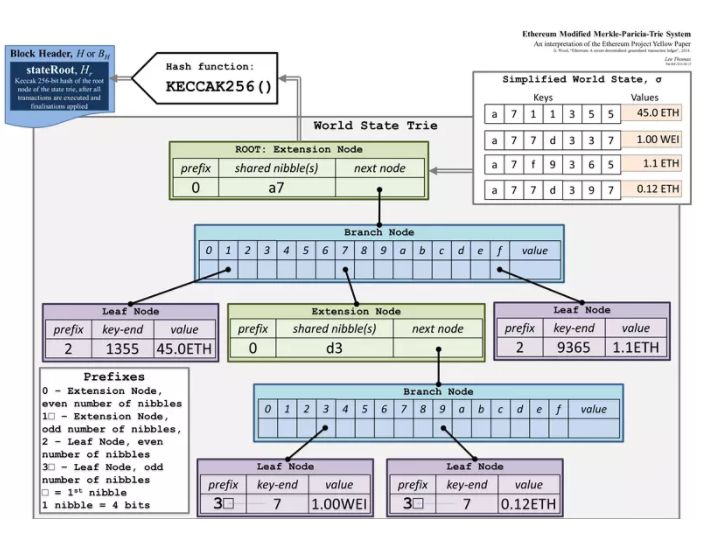
Storage Trie 存储树

* 存储只能合约状态
* 每个账号有自己的Storage Trie
* 

MPT树种还有一个重要的概念：特殊的十六进制前缀(hex-prefix, HP)编码来对key编码，我们先来了解一下编码定义规则：

* RAW 原始编码，对输入不做任何变更
* HEX 十六进制编码：a）RAW编码输入的每个字符分解为高4位和低4位。比如key=>"bob",b的ASCII十六进制编码为0x62，o的ASCII十六进制编码为0x6f，分解成高四位和第四位，16表示终结 0x10，最终编码结果为[6 2 6 15 6 2 16]；b）如果是叶子节点，则在最后加上Hex值0x10表示结束；c）如果是分支节点不附加任何Hex值。

HEX-Prefix 十六进制前缀编码：

* 输入key结尾为0x10，则去掉这个终止符
* key之前补一个四元组这个Byte第0位区分奇偶信息，第1位区分节点类型
* 如果输入key的长度是偶数，则再添加一个四元组0x0在flag四元组后
* 将原来的key内容压缩，将分离的两个byte以高四位低四位进行合并
* 十六进制前缀编码相当于一个逆向的过程，比如输入的是[6 2 6 15 6 2 16]，根据第一个规则去掉终止符16。根据第二个规则key前补一个四元组，从右往左第一位为1表示叶子节点，从右往左第0位如果后面key的长度为偶数设置为0，奇数长度设置为1，那么四元组0010就是2。根据第三个规则，添加一个全0的补在后面，那么就是20.根据第三个规则内容压缩合并，那么结果就是[0x20 0x62 0x6f 0x62]

1. **Gas在智能合约中的作用**

gas是衡量执行某些操作所需的计算量的单位，用来计算为了执行操作而需要支付给网络的费用数额。

智能合约，就是一些代码，运行整个分布式网络中。由于网络中的每一个节点都是一个全节点。这样的好处是容错性强，坏处是效率低，消耗资源与时间（译者注：原来只在一个节点执行一次就行，现在所有节点中每一个，都要执行一模一样的运算）。因为执行计算要花钱，而要执行的运算量与代码直接相关。所以，每个在网络运行的底层操作都需要一定量的Gas。Gas只是一个名字，它代表的是执行所需要花费的成本（注：由于以太坊是图灵完备的，随便一个死循环就将导致网络不可用，所以引入了gas的概念）。整个分布式网络引入了强制限制，来避免停机问题。因此如果你写一个死循环，当gas耗尽后，网络就会拒绝执行接下来的操作，并且回滚你之前的所有操作。  
  
Gas的价格由市场决定，类似于比特币的交易费机制。如果你的Gas价格高，节点则将优先因为利益问题打包你的交易。  
Gas就是用来衡量在一个具体计算中要求的费用单位。Gas price就是你愿意在每个gas上花费Ether的数量，对每个交易，发送者设置gas limit和gas price。gas limit和gas price就代表着发送者愿意为执行交易支付的Wei的最大值。。例如，假设发送者设置gas limit为50,000，gas price为20gwei。这就表示发送者愿意最多支付50,000\*20gwei = 1,000,000,000,000,000 Wei = 0.001 Ether来执行此交易。为执行交易支付的Wei的最大值。

不同存储的消耗（gas消耗）是不一样的，说明如下：

* storage 会永久保存合约状态变量，开销最大；
* memory 仅保存临时变量，函数调用之后释放，开销很小；
* stack 保存很小的局部变量，免费使用，但有数量限制(16个变量)；
* calldata的数据包含消息体的数据，其计算需要增加n\*68的GAS费用；

一经创建，每笔交易都收取一定数量的 gas ，目的是限制执行交易所需要的工作量和为交易支付手续费。EVM 执行交易时，gas 将按特定规则逐渐耗尽。

gas price 是交易发送者设置的一个值，发送者账户需要预付的手续费= gas\_price \* gas 。如果交易执行后还有剩余， gas 会原路返还。

无论执行到什么位置，一旦 gas 被耗尽（比如降为负值），将会触发一个 out-of-gas 异常。当前调用帧（call frame）所做的所有状态修改都将被回滚。

1. **EVM中的数据存储结构；**

每个账户有一块持久化内存区称为 存储 。 存储是将256位字映射到256位字的键值存储区。 在合约中枚举存储是不可能的，且读存储的相对开销很高，修改存储的开销甚至更高。合约只能读写存储区内属于自己的部分。

第二个内存区称为 内存 ，合约会试图为每一次消息调用获取一块被重新擦拭干净的内存实例。 内存是线性的，可按字节级寻址，但读的长度被限制为256位，而写的长度可以是8位或256位。当访问（无论是读还是写）之前从未访问过的内存字（word）时（无论是偏移到该字内的任何位置），内存将按字进行扩展（每个字是256位）。扩容也将消耗一定的gas。 随着内存使用量的增长，其费用也会增高（以平方级别）。

EVM 不是基于寄存器的，而是基于栈的，因此所有的计算都在一个被称为 栈（stack） 的区域执行。 栈最大有1024个元素，每个元素长度是一个字（256位）。对栈的访问只限于其顶端，限制方式为：允许拷贝最顶端的16个元素中的一个到栈顶，或者是交换栈顶元素和下面16个元素中的一个。所有其他操作都只能取最顶的两个（或一个，或更多，取决于具体的操作）元素，运算后，把结果压入栈顶。当然可以把栈上的元素放到存储或内存中。但是无法只访问栈上指定深度的那个元素，除非先从栈顶移除其他元素。

在 Solidity 中，有两个地方可以存储变量 ：存储（storage）以及内存（memory）。Storage变量是指永久存储在区块链中的变量。Memory 变量则是临时的，当外部函数对某合约调用完成时，内存型变量即被移除。

内存(memory)位置还包含2种类型的存储数据位置，一种是calldata，一种是栈（stack）。

Calldata:

这是一块只读的，且不会永久存储的位置，用来存储函数参数。 外部函数的参数（非返回参数）的数据位置被强制指定为 calldata ，效果跟 memory 差不多。

栈（stack）:

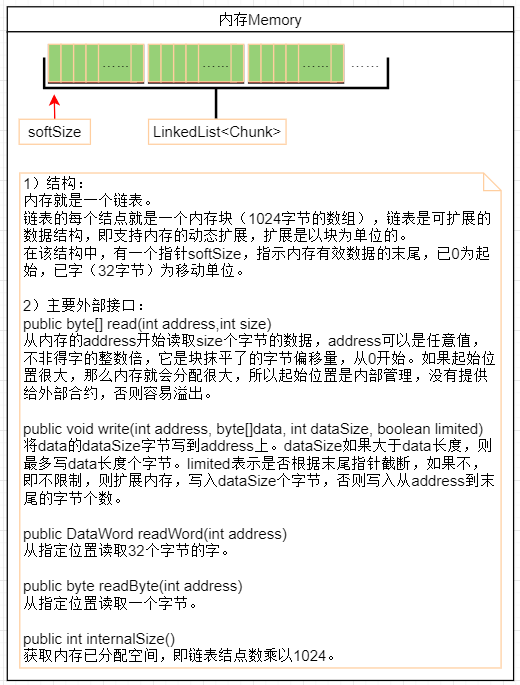
EVM是一个基于栈的语言，栈实际是在内存(memory)的一个数据结构，每个栈元素占为256位，栈最大长度为1024。 值类型的局部变量是存储在栈上。

EVM是一个基于栈的虚拟机。这就意味着对于大多数操作都使用栈，而不是寄存器。基于栈的机器往往比较简单，且易于优化，但其缺点就是比起基于寄存器的机器所需要的opcode更多。

所以EVM有许多特有的操作，大多数都只在栈上使用。

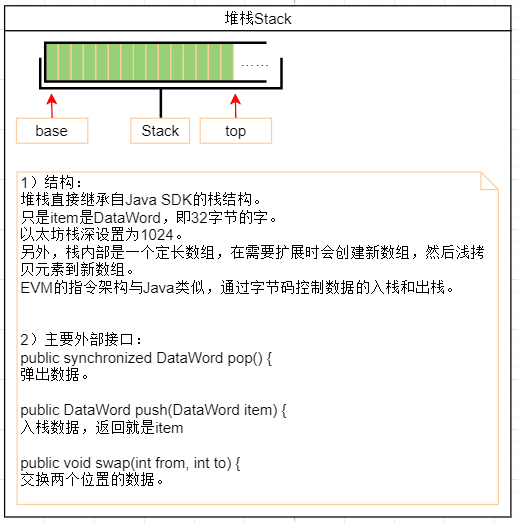
**VM内存**

内存结构像堆栈一样，也提供了数据缓存的功能，但更作用的是提供了合约调用合约等过程中，子合约数据的临时存储。

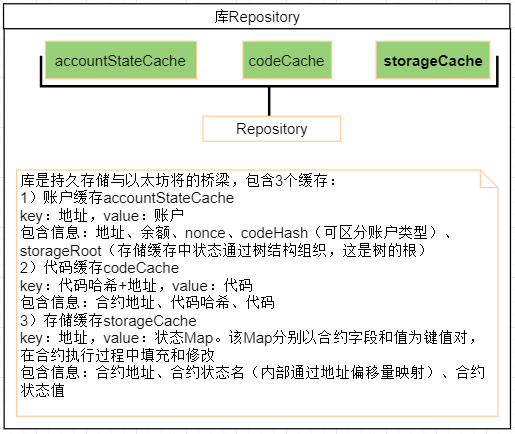


## VM堆栈

EVM的执行模型是基于栈结构的，像JVM一样，这里提供的堆栈就是用来存储字节码执行过程中的中间数据等。

**VM持久化存储**

持久化存储主要是独立于区块存储之外，存储以太坊的账户、合约代码以及合约的状态。



1. **群组架构的好处**

群组架构能克服系统吞吐能力的瓶颈。

有别于传统区块链平台整个网络维护一个账本，所有节点参与到这个账本的共识和存储的做法，群组架构允许网络中存在多个不同的账本，每个账本是一个独立的小组，节点可以选择加入某些小组，参与到该组账本的共识和存储。该架构的特点是：

各群组独立执行共识流程，由群组内参与者决定如何进行共识，一个群组内的共识不受其他群组影响，各群组拥有独立的账本，维护自己的交易事务和数据，使得各群组之间解除耦合独立运作，可以达成更好的隐私隔离；

机构的节点只需部署一次，通过群组设置即可参与到不同的多方协作业务中，或将一个业务按用户、时间等维度分到各群组，群组架构可快速地平行扩展，在扩大了业务规模同时，极大简化了运维复杂度，降低管理成本。

1. **分布式存储有什么优势**

节点可将数据存储在远端分布式系统中，克服了本地化数据存储的诸多限制。有以下优点：

支持多种存储引擎，选用高可用的分布式存储系统，可以支持数据简便快速地扩容；

将计算和数据隔离，节点故障不会导致数据异常；

数据在远端存储，数据可以在更安全的隔离区存储，这在很多场景中非常有意义；

分布式存储不仅支持Key-Value形式，还支持SQL方式，使得业务开发更为简便；

世界状态的存储从原来的MPT存储结构转为分布式存储，避免了世界状态急剧膨胀导致性能下降的问题；

优化了数据存储的结构，更节约存储空间，存取效率更高。

1. **并行计算**

并行计算或称平行计算是相对于串行计算来说的。它是一种一次可执行多个指令的算法，目的是提高计算速度，及通过扩大问题求解规模，解决大型而复杂的计算问题。所谓并行计算可分为时间上的并行和空间上的并行。 时间上的并行就是指[流水线技术](https://baike.baidu.com/item/%E6%B5%81%E6%B0%B4%E7%BA%BF%E6%8A%80%E6%9C%AF/1119843)，而空间上的并行则是指用多个[处理器](https://baike.baidu.com/item/%E5%A4%84%E7%90%86%E5%99%A8/914419)并发的执行计算。

并行计算（Parallel Computing）是指同时使用多种计算资源解决计算问题的过程，是提高计算机系统计算速度和处理能力的一种有效手段。它的基本思想是用多个处理器来协同求解同一问题，即将被求解的问题分解成若干个部分，各部分均由一个独立的处理机来并行计算。并行计算系统既可以是专门设计的、含有多个处理器的超级计算机，也可以是以某种方式互连的若干台的独立计算机构成的集群。通过并行计算集群完成数据的处理，再将处理的结果返回给用户。

过去将交易打包成一个区块，在一个区块中交易顺序串行执行的做法，在FISCO BCOS 2.0中，会通过一套并行交易处理模型实现自定义交易互斥变量。 区块执行过程中，系统将会根据交易互斥变量自动构建交易依赖关系图——DAG，基于DAG并行执行交易，最好情况下性能可提升数倍（取决于CPU核数）。

1. **当前区块链实施的难度**

经过近 3 年的发展，区块链技术已经成为业内公认的，继大数据、云计算、 物联网、人工智能之后，又一颠覆性的新兴技术，也被一些国家列为国家级战略发展技术。

在技术不断积累深入时，国内部分城市已开始鼓励区块链产业入驻，创建出初步的区块链创业扶持政策。与此同时，国外相对成熟的资本市场也正在完善区块链衍生品加密数字货币的相关法规。

技术层面上，区块链技术目前生活化的场景落地以蚂蚁金服、腾讯、百度、京东等巨头企业为主力，区块链创业公司在底层技术上取得较大成果，成为传统企业连接区块链技术的桥梁。

但不可忽视的是，区块链领域的安全问题屡见不鲜，合约机制中的天然弊端和发展过程中产生的漏洞成为不法分子的攻击点。

**参考文献**

Solidity官方文档

<http://cw.hubwiz.com/card/c/solidity-0.5-manual/>

FISCO BCOS技术文档

<https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/index.html>

区块链技术原理博客

<https://blog.csdn.net/zhangcanyan/article/details/51933424>

联盟链公有链异同

<https://36kr.com/p/5129559>

信任链是如何建立的

<https://www.fengli.com/news/23250972.html>

群组架构

<https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/design/architecture/group.html>