# Research report on MPT

刘莹

202100460164

## 一、概念

默克尔树（Merkle Patricia Tree）在以太坊中是一种通用的，用来存储键值对的数据结构，可以简称为“MPT”，是字典树 Redix tree 的变种，也是以太坊的核心算法之一。

MPT 对于树中节点的插入、查找、删除操作，这种结构可以提供对数级别的复杂度 O(log(N))，所以它是一种相对高效的存储结构。

## 二、如何根据键值对构造默克尔树

### （一）、节点类型

#### 1、 Branch

（1）由 17 个元素组成的元组，格式为：(v0,……,v15,vt)。

（2）其中，v0～v15 的元素是以其索引值（0x0~0xf）为路径的子节点数据的 keccak256 哈希值，如果没有子节点数据则元素为空。

（3）vt 为根节点到当前节点的父节点所经过的路径对应的 value 值，也就是根节点到父节点所经过的路径组成了一个键 key，这个 key 对应的 value 存在 vt 里面，如果这个 key 没有对应的 value，那么 vt 为空。

#### 2、 Leaf

（1）两个元素组成的元组，格式为：（encodePath,value）；

（2）encodedPath 为当前节点路径的十六进制前缀编码；

（3）value 是从根节点到当前节点路径组成的键对应的值。

**3、 Extension**

（1）两个元素组成的元组，格式为：（encodePath,key）；

（2）encodedPath 为当前节点路径的十六进制前缀编码；

（3）key 为当前节点子节点数据的 keccak256 哈希值。

### （二）十六进制前缀编码

branch 和 extension 元组的第一个元素 encodePath 就是当前节点路径的十六进制前缀编码（Hex-Pretix Encoding，HP 编码）。使用 HP 编码能够区分节点是扩展结点还是叶子节点。

而 HP 编码，和当前节点类型还有当前路径半字节长度的奇偶有关。共有四种前缀：

| 路径所对应的节点类型 | 路径长度 | 二进制数值 | 十六进制数值 | 最终前缀（HP前缀） |
| --- | --- | --- | --- | --- |
| Extension | 偶数个半字节 | 0000 | 0x0 | 0x00 |
| Extension | 奇数个半字节 | 0001 | 0x1 | 0x1 |
| Leaf | 偶数个半字节 | 0010 | 0x2 | 0x20 |
| Leaf | 奇数个半字节 | 0011 | 0x3 | 0x2 |

所以 extension 节点有两种前缀：0x00、0x1；leaf 有两种前缀：0x20、0x3。

可以看到最终前缀在偶数个半字节 0x0、0x2 后补了一个 0，变成了 0x00，0x20，目的是为了凑成整字节，避免出现半字节导致长度不便

于合并。

HP 前缀需要放在原始路径前面去组成 HP 编码，实例：

| 原始数据 | 节点类型 | HP前缀 | HP编码 |
| --- | --- | --- | --- |
| (0x012345,key)（注：012345长度为偶数） | Extension | 0x00 (0000 0000) | (**0x00**012345,key) |
| (0x12345,key)（注：12345长度为奇数） | Extension | 0x1 (0001) | (**0x1**2345,key) |
| (0x0f1cb8,value)（注：0f1cb8长度为偶数） | Leaf | 0x20 (0010 0000) | (**0x20**0f1cb8,value) |
| (0xf1cb8,value)（注：f1cb8长度为奇数） | Leaf | 0x3 (0011) | (**0x3**f1cb8,value) |

三、构造一颗默克尔树

上面的概念不容易理解，现在我们以下面的例子，一步步来进行树的构造，帮助我们更好的理解：

我们假设有一组（4 个）键值对数据需要用树来存储：

```
<64 6f> : 'verb'
<64 6f 67> : 'puppy'
<64 6f 67 65> : 'coin'
<68 6f 72 73 65> : 'stallion'
```

为方便解释说明以及阅读，我们把键值对数据的“键”表示为十六进制字符串，“值”则保留为原始字符串。在实际使用时，它们都需要经过特定的编码变换。

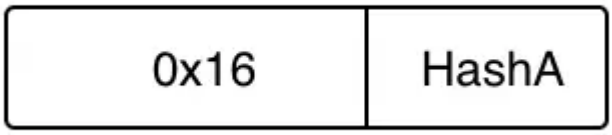
1、每棵树都有根节点，默克尔树的根节点会保存当前路径和子节点哈希，所以很明显，根节点会是一个 extension 节点。

上面节点类型介绍了 extension 格式为：（encodePath,key），encodePath 是十六进制的 HP 编码。分析给出的 4 个键我们可以得出都是以 6 开头，后面分为 4、8 两条路。所以根节点存储的共同路

径值为 0x6。

由于 0x6 只有一位，所以路径长度是奇数，节点又是 extension 类型，所以 HP 前缀是 0x1，组合出来的 HP 编码：0x16。

所以当前默克尔树如下图：



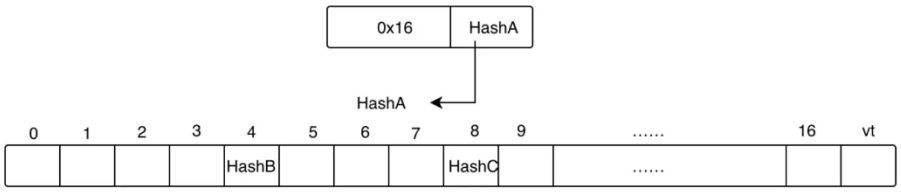

HashA 代表着子节点的哈希值。

2、根节点已经找到，但在根节点后出现了两条路，这个时候需要使用 branch 来处理这种多条路径的情况。

上文说到，branch 由 17 个元素组成的元组，格式为：(v0,……,v15,vt)。其中，v0 ~ v15 是以其索引值（0x0~0xf）为路径的子节点数据的 keccak256 哈希值，如果没有子节点数据则为空。

这里 4 和 8 就是索引值，4、8 对应元素是其字节点的哈希值。

所以当前默克尔树如下图：

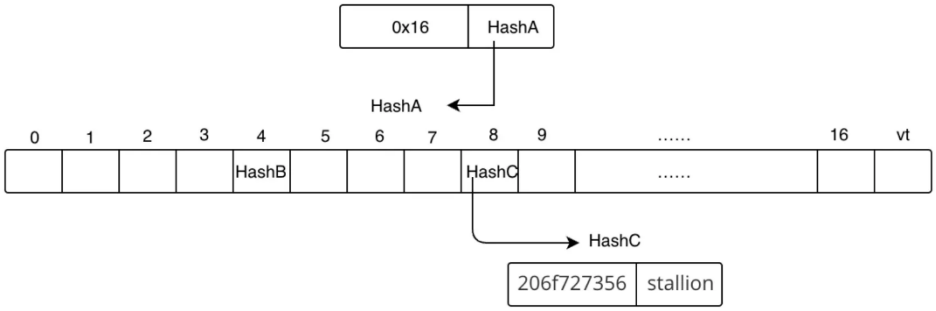



3、我们可以观察到，在 0x68 后只有唯一路径了，即 0x6f727356，而 value 为“stallion”，所以不再分叉的情况下，就不是 branch 或者 extension 了，而应该是一个叶节点。

上文提到，leaf 节点是两个元素组成的元组，格式为：（encodePath,value），encodedPath 为当前节点路径的十六进制前缀编码，value 是从根节点到当前节点路径组成的键，所对应的值。

当前节点的路径是 0x6f727356，长度是偶数，节点类型是 leaf，所以可以得出 HP 前缀是 0x20，HP 编码是 0x206f727356。所以可得该 leaf 节点：（0x206f727356,"stallion"）。

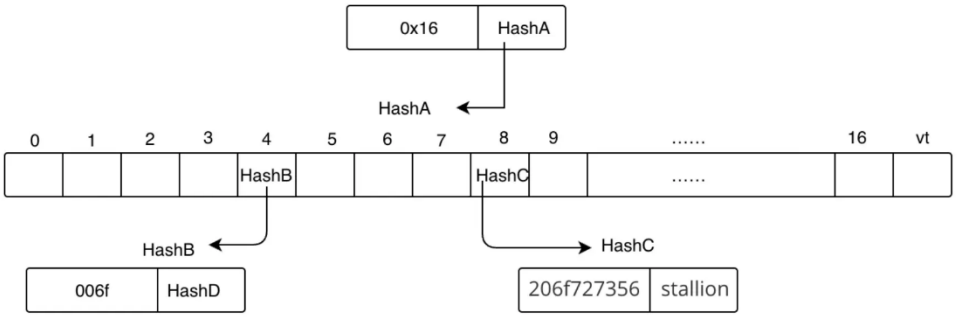
所以当前默克尔树如下图：




4、说完了 8，我们再说说 4 这部分，路径 4 后面有共同路径 6f，6f 后才产生 null 和 6 两条分叉。

共同路径 6f 是一个 extension 节点，extension 节点格式不再介绍，开始计算 HP 编码，6f 长度是偶数，又是 extension 类型，所以 HP 前缀为 0x00，HP 编码为 0x006f。

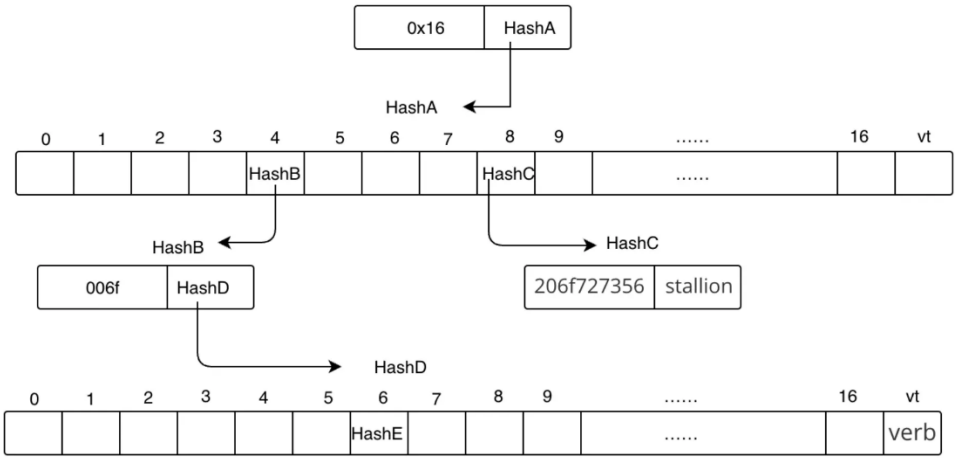
所以当前默克尔树如下图：




5、6f 后分出了 null 和 6，是多条路径，所以 HashD 的节点是一个 branch 节点，6 是索引值，索引为 6 的元素存储着子节点 hash；而 null 是没有的，上文提到：vt 为根节点到当前节点的父节点所经过的路径组成的键对应的 value。

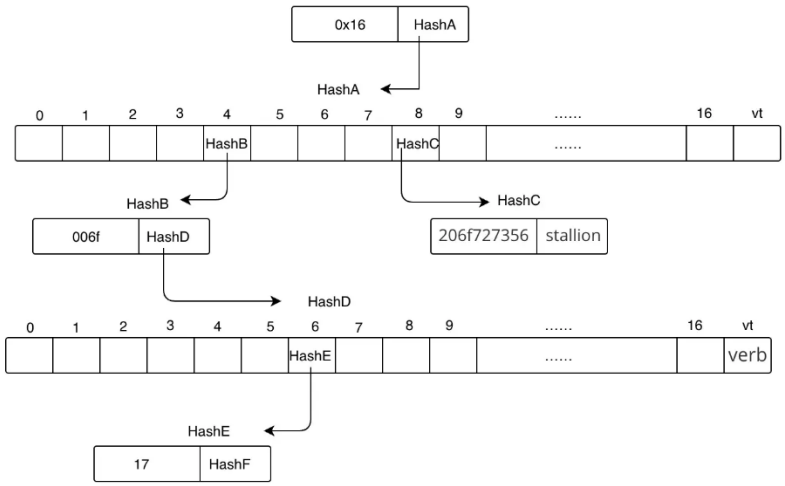
则代表当前 HashD 节点该存储从根节点到父节点 0x646f 组成的键对应的值：'verb'。那么该由 HashD 的 vt 保存'verb'。

所以当前默克尔树如下图：

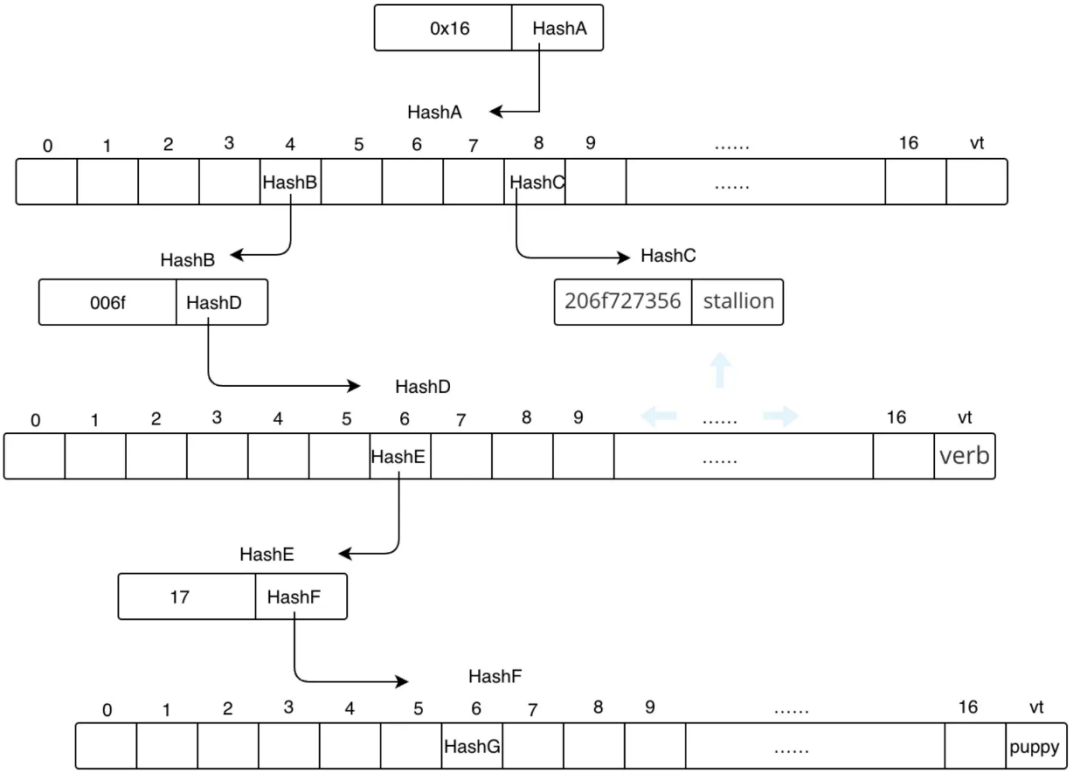


6、接下来是共同路径 7，一个 extension 节点，开始计算 HP 编码，7 长度是奇数，又是 extension 类型，所以 HP 前缀为 0x1，HP 编码为 0x17。

所以当前默克尔树如下图：

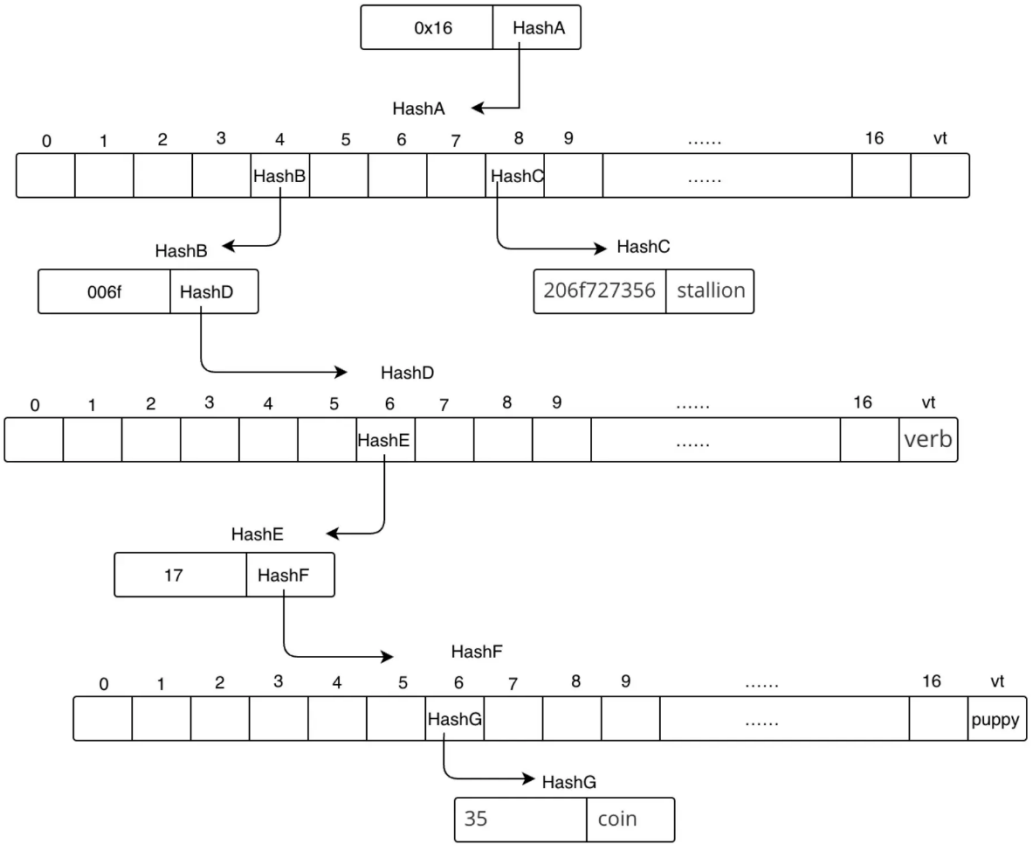



7、7 后分出了 null 和 6，是多条路径，与第五步相同，HashF 是一个 branch 节点，索引为 6 的元素存储子节点哈希，vt 存储'puppy'的值。所以当前默克尔树如下图：




8、好了，现在只剩下一条路径了，表示这最后一个是一个 leaf 叶子节点，路径为 5，路径长度为奇数，索引 HP 前缀为 0x3，HP 编码为 0x35。

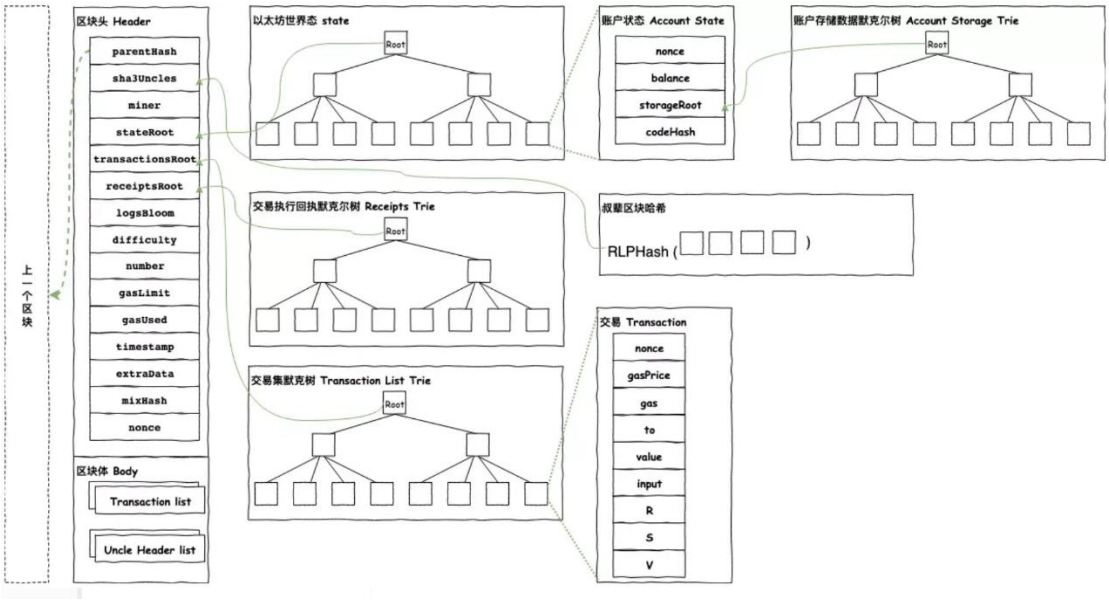
所以当前也是最终的默克尔树如下图：




## 三、总结

从构造过程中我们可以看出，MPT 中节点之间，是通过哈希值来确定的。由于哈希值的特性，只要数据有了微小改动，就会导致根节点改变，所以我们可以用树的根节点来代表整个树中数据的状态，这样就不用保存整个树的数据。

在以太坊中，默克尔树有着大量的应用，比如保存和验证系统中的所有账户状态、所有合约的存储状态、区块中的所有交易和所有收据数据的状态等。




参考：https://learnblockchain.cn/article/5321