默克尔帕特里夏树（Merkle Patricia Trie，简称 MPT）是一种用于存储和检索数据的数据结构，常被应用于区块链技术中。MPT 是以太坊（Ethereum）区块链的核心数据结构之一，用于存储交易数据、合约代码和账户状态等。

MPT 的构建原理主要涉及默克尔树（Merkle Tree）和前缀树（Trie）。默克尔树是一种将大量数据进行哈希计算并形成树状结构的方法，它可以有效验证数据的完整性。而前缀树是一种特殊的树状数据结构，用于高效地存储和检索字符串。下面首先对默克尔树与前缀树做出介绍，之后在此基础上融合前两种结构引入默克尔帕特里夏树（MPT）。

1、默克尔树（Merkle Tree）

Merkle 树[1]是由计算机科学家 Ralph Merkle 在很多年前提出的，并以他本人的名字来命名。默克尔树是一种基于哈希的数据结构，它是哈希列表的一种推广。

1） 特点

- Merkle tree 是一种树，大多数是二叉树，也可以多叉树，无论是几叉树，它都具有树结构的所有特点；
- Merkle tree 叶子节点的 value 是数据项的内容，或者是数据项的哈希值；
- 非叶子节点的 value 根据其孩子节点的信息，然后按照 Hash 算法计算而得出的；

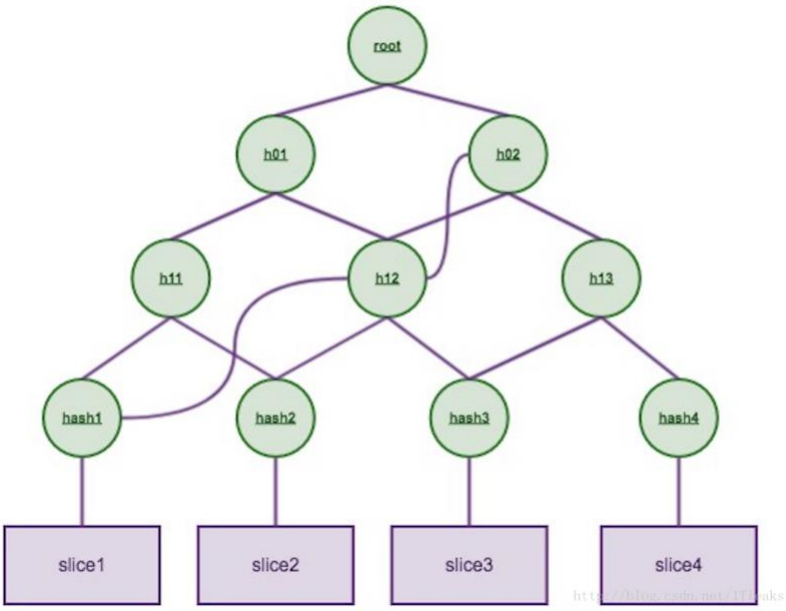


图 1：Merkle Tree

将相邻两个节点的哈希值合并成一个字符串，然后计算这个字符串的哈希，得到的就是这两个节点的父节点的哈希值。如果该层的树节点个数是单数，那么对于最后剩下的树节点，这种情况就直接对它进行哈希运算，其父节点的哈希就是其哈希值的哈希值。（对于单数个叶子节点，采用复制最后一个叶子节点凑齐偶数个叶子节点的方式）。循环重复上述计算过程，最后计算得到最后一个节点的哈希值，将该节点的哈希值作为整棵树的哈希，即默克尔根。

若两棵树的根哈希一致，则这两棵树的结构、节点的内容必然相同。

2） 优势

- 快速重哈希

Merkle tree 的特点之一就是当树节点内容发生变化时，能够在前一次哈希计算的基础上，仅仅将被修改的树节点进行哈希重计算，便能得到一个新的根哈希用来代表整棵树的状态。

■ 轻节点扩展

采用 Merkle tree，可以在公链环境下扩展一种“轻节点”。轻节点的特点是对于每个区块，仅仅需要存储约 80 个字节大小的区块头数据，而不存储交易列表，回执列表等数据。然而通过轻节点，可以实现在非信任的公链环境中验证某一笔交易是否被收录在区块链账本的功能。这使得像比特币，以太坊这样的区块链能够运行在个人 PC，智能手机等拥有小存储容量的终端上。

对于轻节点来说，验证一条交易只需要验证包含该交易的路径即可，并不需要把所有交易的 Hash 全部重新算一遍。

2、前缀树（Trie）

Trie[2]，又称前缀树或字典树，是一种有序树状的数据结构，其中的键通常是字符串，常用语存储 Key-value 数据结构。

Trie 与二叉查找树不同，键不是直接保存在节点中，而是由节点在树中的位置决定。一个节点的所有子孙都有相同的前缀，节点对应的 key 是根节点到该节点路径上的所有节点 key 值前后拼接而成，节点的 value 值就是该 key 对应的值。根节点对应空字符串 key。

常见的用来存英文单词的 trie 每个节点是一个长度为 27 的指针数组，index0-25 代表 a-z 字符，26 为标志域。如图 2:

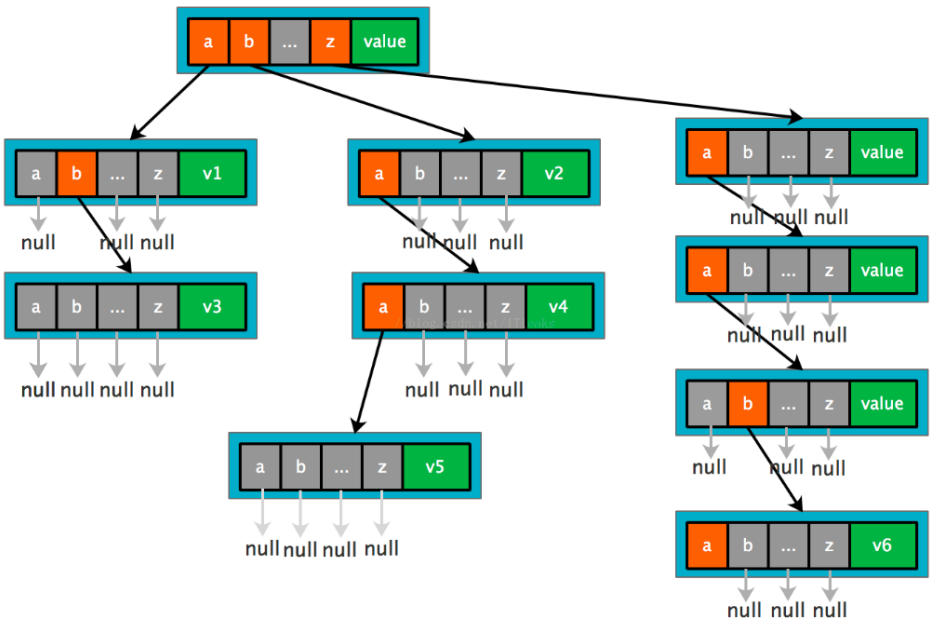


图 2：Trie

1） 优势

相比于哈希表，使用前缀树来进行查询拥有共同前缀 key 的数据时十分高效，例如在字典中查找前缀为 pre 的单词，对于哈希表来说，需要遍历整个表，时间效率为 O(n)，然而对于前缀树来说，只需要在树中找到前缀为 pre 的节点，且遍历以这个节点为根节点的子树即可。但是对于最差的情况（前缀为空串），时间效率为 O(n)，仍然需要遍历

整棵树，此时效率与哈希表相同。

相比于哈希表，在前缀树不会存在哈希冲突的问题。

2) **劣势**

直接查找效率低下 前缀树的查找效率是O(m)，m为所查找节点的key长度，而哈希表的查找效率为O(1)。且一次查找会有m次IO开销，相比于直接查找，无论是速率、还是对磁盘的压力都比较大。

可能会造成空间浪费 当存在一个节点，其key值内容很长(如一串很长的字符串)，当树中没有与他相同前缀的分支时，为了存储该节点，需要创建许多非叶子节点来构建根节点到该节点间的路径，造成了存储空间的浪费。

3、默克尔帕特里夏树（Merkle Patricia Tree)

在MPT中，数据被存储在一个树状结构中，该结构由节点和边组成。每个节点可以有多个子节点，而边代表节点之间的关系。MPT的每个节点都包含一个256位的数据块(Blob)，其中存储了一个键值对。键是一个256位的哈希值，值可以是任何数据。MPT按照键的前缀进行组织，因此它也被称为键前缀树。

1) **三种节点类型**

- 分支结点（branch node)：包含16个分支，以及1个value

分支节点，因为MPT树中的key被编码成一种特殊的16进制的表示，再加上最后的value，所以分支节点是一个长度为17的list，前16个元素对应着key中的16个可能的十六进制字符，如果有一个[key,value]对在这个分支节点终止，最后一个元素代表一个值，即分支节点既可以搜索路径的终止也可以是路径的中间节点。

- 扩展结点（extension node)：只有1个子结点

扩展节点，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。

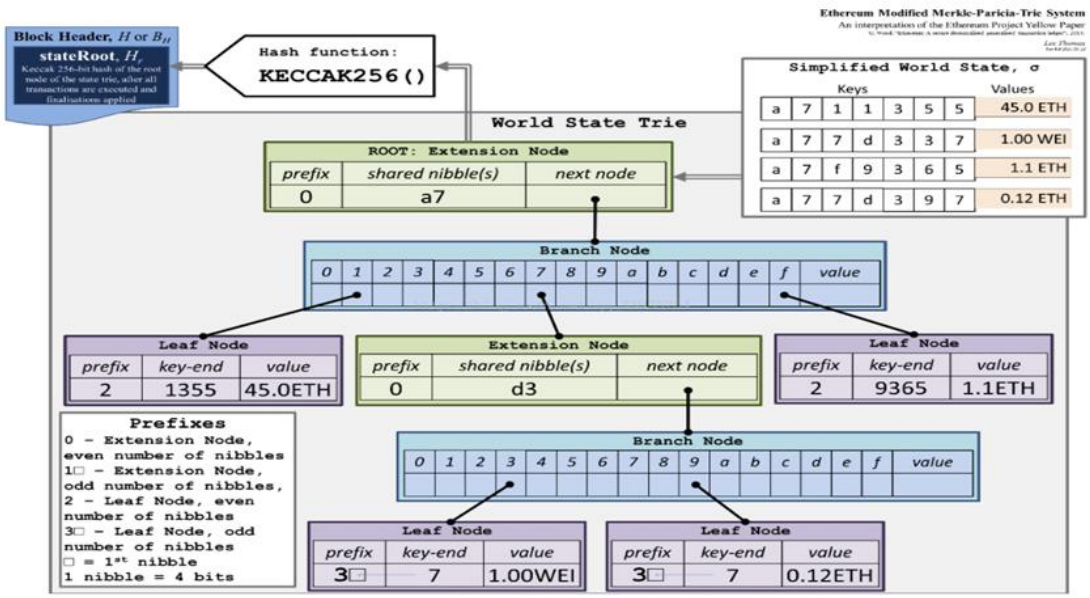
- 叶子结点（leaf node)：没有子结点，包含一个value

叶子节点，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。

2) **MPT中的Merkle**

即指向下一级节点的指针是使用节点的确定性加密hash，而不是传统意义上下一级节点地址的指针如果给定的trie的根哈希是公开的，则任何人都可以通过给出给定path上的所有节点，来证明在给定path上存在一个给定值，对于攻击者,不可能提供一个不存在的（key，value）对的证明，因为根哈希最终基于它下面的所有哈希，所以任何修改都会改变根哈希。

3) **官方表示形式**



4、参考资料

【1】https://zhuanlan.zhihu.com/p/46702178

【2】https://blog.csdn.net/ITleaks/article/details/79992072

【3】https://www.cnblogs.com/whyaza/p/10034128.html#%E9%BB%98%E5%85%8B%E5%B0%94%E6%A0%91

【4】查询资料得到 MPT 代码实现-- MPT/默克尔帕克里夏树/MPT.py at main · WangHao-nlp/MPT · GitHub