# Merkle Tree及其优化

## Merkle Tree 介绍

Merkle Tree是一种树形的数据结构，每个叶节点均以数据块的哈希作为标签，而除了叶节点以外的节点则以其子节点标签的加密哈希作为标签。Merkle Tree 能够高效、安全地验证大型数据结构的内容，是哈希链的一种推广形式。其又名哈希树，由Ralph Merkle申请专利定义，故亦称为墨克尔树。

Merkle Tree 通常是一个二叉树的形式（也有可能是多叉树），它以特定的方式逐层向上计算，直到树根。Merkle Tree的叶子节点的value是数据集合的单元数据或者其哈希值（例如图1.1中单元数据T1与其哈希值H1）。非叶子节点的value根据其左右孩子的哈希值计算得出。

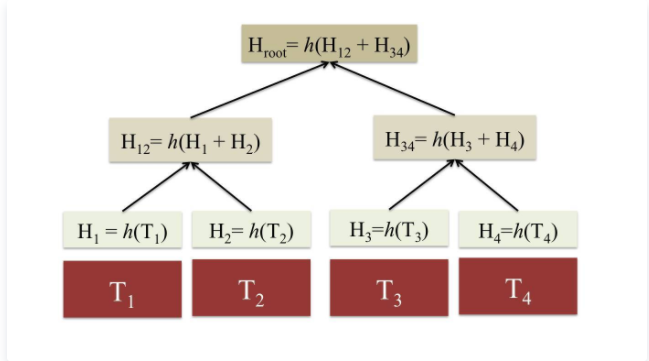


图 1-1 Merkel Tree 结构

1. 创建Merkle Tree

创建Merkle Tree时，首先将单元数据放入相同数量的叶子节点中，然后逐层向上的去构建树中的每一层，易得构建复杂度为，其中n是数据块的大小，Merkle Tree的树高是。

1. 检索数据块

假设A，B两台机器有相同的16个文件，其对应的哈希值分别是。这时我们可以通过Merkle Tree 快速比较两台机器上的文件是否完全相同。从Merkle Tree 的创建过程可以看出，其根节点的value是所有叶子节点value的唯一特征。只要两颗Merkle Tree中叶子节点的value有一处不同，其根节点就不会相同。

如果A和B中第5个文件不同（不同）：

1. 从Root出发，发现其根节点哈希值不同，检索其孩子和。
2. 不同，相同，检索的孩子和。
3. 相同，不同，检索的孩子和。
4. 不同，相同，检索的孩子和。
5. 不同，相同，获取其文件信息进行对比。

以上对比过程的理论复杂度是。所以，Merkle Tree支持较快的检索不同的文件，这在P2P网络中用于确保从其他节点接受的数据块没有损坏且没有被破坏。

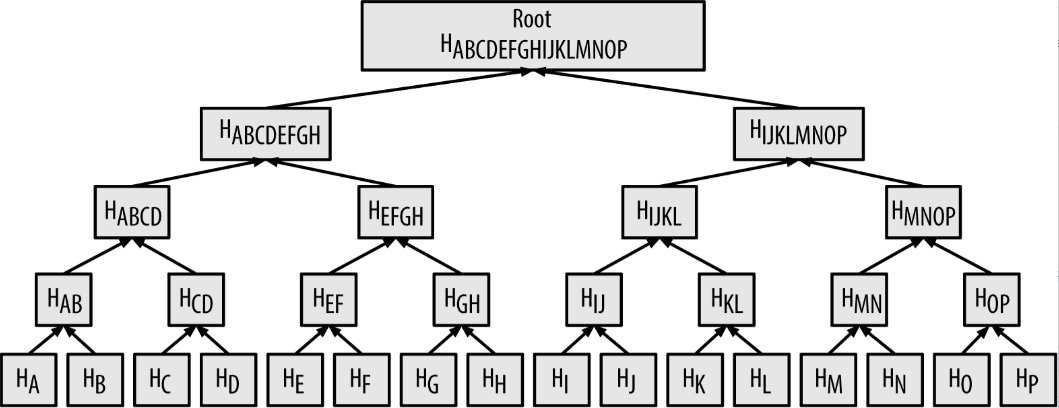


图 1-2 Merkle Tree

基于该特点，在传输大规模数据时，可以将其分散成组来确保数据传输的完整性。如果有一个组在传输时出差错，Merkle Tree可以快速检测出来时哪一个组出了差错，然后及时的重传该小组的数据。

1. 更新、插入与删除

Merkle Tree的插入与删除是一个工程上的问题，不同的应用场景会有不同的插入方法。如果要确保树是平衡的，可以使用平衡二叉树如AVL树、红黑树、Splay等基本模式来实现Merkle Tree树，它们都可以支持不超过时间复杂度的插入、删除与更新。

1. Merkle Proof

Merkle Tree不仅可以快速的比对两份大数据是否相同，还可以在不完整的Merkle Tree 上来进行完整性检验，快速判断某数据小组是否在其应处的位置。

在图1-3中，包含了8个有效元素，在传输过程中，如果先收到了数据64，我们尝试在其他数据尚未获取之前，证明以下两点：

1. 数据64的确存在于这组数据中
2. 数据64存在的位置是第4格

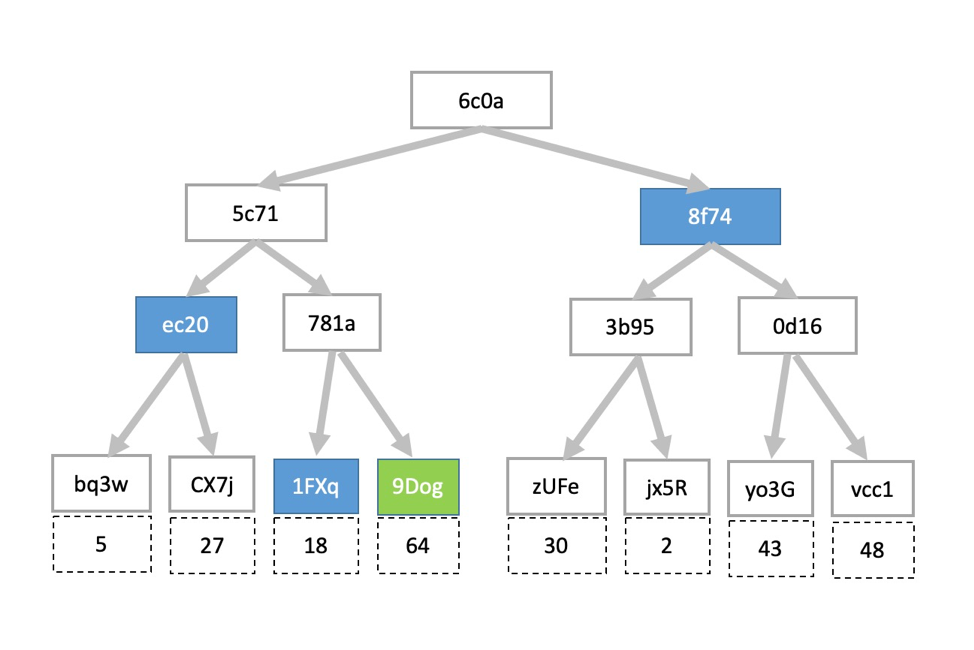


图 1-3 Merkle Proof 证明数据64有效存在于该树中

根据Merkle Proof，我们仅需要知道4个哈希值即可证明：

1. 数字64的相邻数字18的哈希值（1FXq）
2. 相邻分组{5,27}的哈希值（ec20）
3. 相邻分支的哈希值（8f74）
4. 根节点哈希值（6c0a）

证明时，首先计算64的哈希值9Dog，然后串联哈希值1FXq，计算得到联合哈希值781a。

重复上述过程，将781a与ec20串联，计算得出联合哈希值5c71。

重复上述过程，将5c71与8f74串联，计算得出根哈希值6c0a。

若通过Merkle Proof得到的哈希值6c0a与根哈希一致，表示数据64的确存在于这组数据中，并且存在的位置是第4格。

## Merkle Tree的缺陷

对于验证属于list格式的信息而言，二叉Merkle Tree是非常好的数据结构。

## 优化方案

## 优化效果

## 总结