计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(9):237−243 [doi: 10.15888/j.cnki.csa.007528]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

区块链中 Merkle 树性能研究①

黄　根[1,3], 邹一波[1,2], 徐　云[1,3]

[1](中国科学技术大学 计算机科学与技术学院, 合肥 230026)
[2](上海海洋大学 信息学院, 上海 201306)
[3](安徽省高性能计算重点实验室, 合肥 230026)
通讯作者: 徐　云, E-mail: xuyun@ustc.edu.cn

摘　要: 区块链技术具有去中心化、安全可靠和不可篡改等特性, 已经得到广大的重视. Merkle 树是区块中核心组成部分, 占据区块存储空间的 96% 以上, 主要用来解决在区块链交易中的简化支付验证问题, 因此选择合适的 Merkle 树结构会极大影响区块链系统性能. 但是, 目前缺乏公共的平台对不同区块链系统下的 Merkle 树性能进行分析和实验验证. 本文提出了一整套相关性能评价与分析指标, 从存储、验证和构建时间等方面, 综合评价比特币、以太坊和超级账本三种主流区块链的 Merkle 树的性能. 本文提出的指标及评价方法不仅为 Merkle 树的进一步研究提供了定量的数据支持, 也为区块链从业者选择 Merkle 树结构提供了理论依据.

关键词: 区块链; 比特币; 以太坊; 超级账本; Merkle 树

引用格式: 黄根,邹一波,徐云.区块链中 Merkle 树性能研究.计算机系统应用,2020,29(9):237–243. http://www.c-s-a.org.cn/1003-3254/7528.html

Performance Analysis and Research of Merkle Trees with Blockchain

HUANG Gen[1,3], ZOU Yi-Bo[1,2], XU Yun[1,3]

[1](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China)
[2](College of Information Technology, Shanghai Ocean University, Shanghai 201306, China)
[3](Key Laboratory of High Performance Computing of Anhui Province, Hefei 230026, China)

Abstract: Blockchain has the characteristics of decentralization, security, reliability, and immutability, and has received widespread attention recently. Merkle tree is the core component of the block, accounting for more than 96% of the block storage. It is mainly used to handle the problem of simplified payment verification in Blockchain transactions. Therefore, choosing the appropriate Merkle tree structure will greatly affect the performance of the Blockchain. However, there is no public platform to analyze and verify the performance of Merkle tree under different Blockchain systems at present. In this study, we propose a set of related performance evaluation and analysis indexes in terms of storage, verification, and build time. The performance of the Merkle tree of the three mainstream Blockchains of Bitcoin, Ethereum, and Hyperledger is evaluated. The index and evaluation method proposed in this study not only provides quantitative data support for further research on Merkle trees, but also provides guidance for Blockchain practitioners in choosing Merkle tree structures.

Key words: Blockchain; Bitcoin; ethereum; hyperledger fabric; Merkle tree

自从 2009 年比特币[1] 问世, 区块链技术逐渐进入人们的视野. 区块链本质就是一个新型的分布式存储系统[2], 是将传统计算机技术中的分布式存储、分布式共识、密码学、点对点传输的各种技术进行创新性结合的新技术, 它也被视为继大型机、个人电脑、互联网和移动/社交网络之后的第五种颠覆性创新技术[3],

① 基金项目: 国家自然科学基金面上项目 (61672480)
Foundation item: General Program of National Natural Science Foundation of China (61672480)
收稿时间: 2019-12-24; 修改时间: 2020-01-20; 采用时间: 2020-02-11; csa 在线出版时间: 2020-09-04

目前已经用于物联网[4]、智能制造[5]、供应链管理[6]、数据资产交易[7] 等多个领域. 对于区块链, 主要实现了分布式存储中的去中心化、去信任化、数据的时间序列化、集体维护等特征, 同时也保障了可编程性、安全性和可靠性[8].

在区块链中的每个区块中, 主要分为区块头和区块体[9]. Merkle 树是区块体的主要构成部分, 构建 Merkle 树也是构建区块体的主要工作. 因此 Merkle 树的构建时间和存储大小都将影响区块链的性能.

在区块链的技术发展过程中, Merkle 树的结构也在不断的发生改变. 目前, 主流的区块链框架主要指比特币、以太坊[10] 和超级账本[11]. 在比特币中, 其采用传统的 Merkle 树结构; 以太坊在比特币的基础上进行了改造, 汲取了 Patricia 树检索高效的优点, 融合 Merkle 树和 Patricia 树而产生的 Merkle Patricia 树[12]; 超级账本为了减少添加数据的代价, 采用了融合 Merkle 树和 Hash 桶的 Bucket 树[11] 结构.

在本文中, 我们首先对区块链中 Merkle 树的结构和关键操作进行分析, 从理论上对 Merkle 树的作用和复杂度进行解析; 然后根据分析结果提出相应的性能评测指标, 最后通过实验进行验证和分析.

1 区块链中的 Merkle 树结构分析

1.1 传统 Merkle 树结构

在计算机科学与密码学中, Merkle 树是一种树形数据结构. 每个叶子节点均以数据块的哈希值作为标签, 而除了叶子节点之外的节点则以其子节点标签的加密哈希值作为标签. Merkle 树可以实现快捷的数据验证, 因此能够高效地验证大型数据结构的内容[13].

如图 1 所示, 该图表明了 Merkle 树的结构. Merkle 树可以被看成是 Hash 表的推广, 即 Hash 表其实是树高为 2 的 Merkle 树. 其形成的过程主要为:

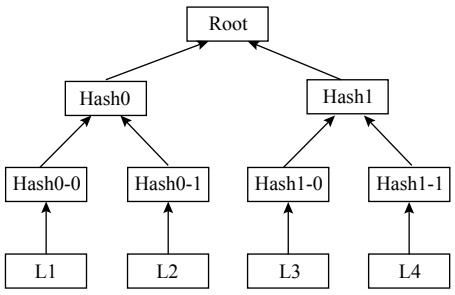

图 1 Merkle 树结构

(1) 在 Merkle 树的最底层, 将数据分成一个个小的数据块, 且各数据块拥有对应的 Hash 值, 作为叶子节点;

(2) 叶子节点到上层树结构时, 将相邻的两个 Hash 值合并成一个字符串, 计算该字符串的 Hash 值;

(3) 每一层都重复过程 (2), 相邻两个 Hash 值便可以一个子 Hash 值. 若最底层的 Hash 总数为单数, 则直接对其做 Hash 运算.

从叶子节点向上依此类推, 最终形成 Merkle 树. 到了根节点就只剩下一个根 Hash 值了, 我们将其称为 Merkle Root.

1.2 区块链中的 Merkle 树结构

在以比特币、以太坊和超级账本为代表的主流区块链系统中, 在 Merkle 上分别有着不同的设计与实现. 下面, 我们将依次介绍这三种不同区块链系统中 Merkle 树的结构.

(1) 比特币的 Merkle 树

在比特币系统中, 区块分为区块头和区块体, 其中区块体则是由 Merkle 树构成. 其中, 整个区块的大小为 2 M, 区块头的大小固定为 80 个字节, 因此, 区块体的大小占了整个区块的 96%.

比特币的 Merkle 结构和传统的 Merkle 树完全相同. 如图 1 所示, 在比特币中, Merkle 树的叶子结点为系统中每一笔交易的 Hash 值. 不同交易利用 Merkle 树的结构进行组合, 最终生成比特币的 Merkle 树, 同时产生由所有交易产生的 Merkle 根.

(2) 以太坊的 Merkle 树

在比特币模型中, 采用 UTXO 模型表示数据, 因此采用传统的 Merkle 树简单快捷[14]; 但是以以太坊为代表的第二代区块链中, 大多采用账户模型[15], 此时传统的 Merkle 树会带来极大的存储空间消耗, 并且不易查询. 因此, 在以太坊中, 对 Merkle 树进行改进, 设计出一种新的数据结构——Merkle Patricia 树进行数据存储.

Merkle Patricia 树简称 MPT, 其本质上是先将前缀树进行改进, 然后再和 Merkle 树进行结合.

在原始的前缀树中, 存在空间浪费、节点不易控制、深度长等缺陷. MPT 在前缀树的基础上进行了改进, 主要做了以下几个方面.

① 为了防止 key 的长度不同造成浪费, 首先将所有的 key 值进行 Hash, 将所有的 key 转换成相同的长度;

② 为了方便对节点的控制, 将所有的节点分为空节点、分支节点、叶子节点和扩展节点 4 种不同的节

点, 分别存储不同的内容;

③ 为了更好的进行比较, 将 key 中的每个参数进行 hex 编码. 同时使用 hex 编码后, 每个值大小为 [0–15], 可以使每个分支节点的容量都减小;

④ 为了缩短树的深度, 将连续两个级以上的扩展节点进行合并.

通过如上 4 步改进, 传统的前缀树结构如图 2 所示. 最后, 将改进后的前缀树和 Merkle 树进行结合, 将存储的 value 值转换成所有子节点的 Hash 值, 即可组成 Merkle Patricia 树.

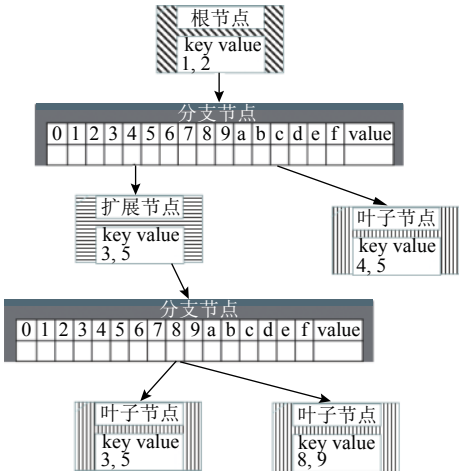

图 2 改进的前缀树

(3) 超级账本的 Merkle 树

超级账本采用账户模型来进行数据存储[16], 但是以太坊的 MPT 树结构复杂、性能低, 超级账本则提出了 Bucket 树[17] 进行数据存储.

Bucket 树其实是揉合了 Merkle 树与 Hash 桶两种数据结构组合而成的, 即 Bucket 树本质上是一棵建立在 Hash 表上的 Merkle 树.

如图 3 所示, B1-B6 为数据的 Hash 桶, 每个桶中储存若干被散列到该桶中的数据项, 所有数据项都按序排列. 每一个桶有可以用一个 Hash 值表示其状态, 该 Hash 值是桶内所有数据项的内容进行 Hash 计算所得. 除底层的 Hash 表之外, 上层是一系列 Merkle 树节点. 一个 Merkle 树节点对应下一层的 n 个 Hash 桶或 Merkle 节点. Merkle 节点的 Hash 值是根据这 n 个孩子节点的 Hash 值计算所得. 通过这种方式不断计算, 与 Merkle 树类似, 最顶端的值即为整棵树的 Hash 值, 即 Merkle 根. 由此可以生成完整的 Bucket 树.

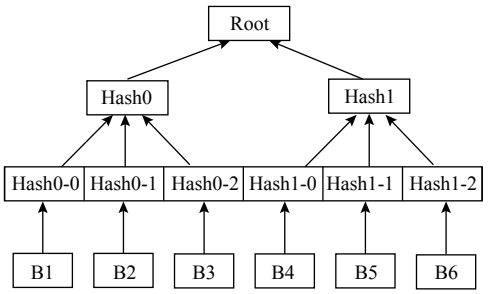

图 3 Bucket 树

2 区块链中 Merkle 关键操作分析

区块链本质上是一个去中心化的数据库, Merkle 树是区块链中核心存储的数据. 和传统的数据库相比, 区块链只能进行增加和查询的操作, 而不能进行删除和修改. 因此, Merkle 树中核心的操作就是数据插入. 同时, Merkle 树的一个重要作用是 SPV 验证 (实现简单支付验证), 即 Merkle 树可以直接下载一个分支并立即验证. 因此, 在本文中, 我们主要分析 Merkle 树中两个核心的操作: 插入数据和 SPV 验证.

2.1 数据插入

数据插入是 Merkle 树中最为核心的操作, 是实现区块链系统中数据插入功能的底层. 因此, 理解不同区块链系统中 Merkle 树的数据插入的原理, 是理解不同区块链系统性能瓶颈的重要途径.

对于比特币中的 Merkle 树, 当需要插入一个新的数据时, 通常在之前的 Merkle 树基础上进行插入. 如图 4 所示, 如果插入数据 L4, 通常情况下只需要在之前的 Mekle 的基础上重新插入一条新的路径, 然后通过和之前的 Merkle 树中的节点进行 Hash 计算, 即可产生新的 Merkle 树.

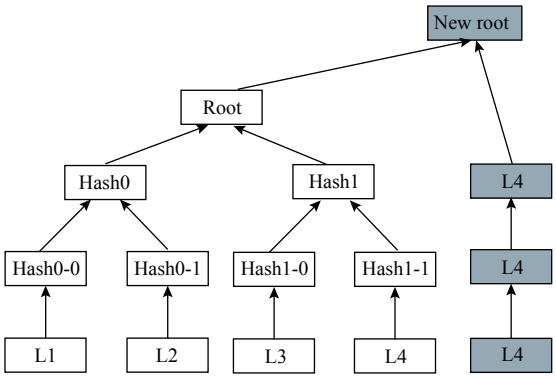

图 4 Merkle 树的插入

对于以太坊中的 MPT, 其插入数据的过程相对比较复杂, 其核心的思想为: 按照 key 的内容, 从根节点

依次从上往下寻找相同长度的路径, 同时随时根据 MPT 的性质改变节点的性质, 插入新的 key 的值, 直到 key 遍历结束. 如图 5 所示, 该虚线路径展现了键值对为<‘1,2,2,4,5’,‘gen’>的数据插入.

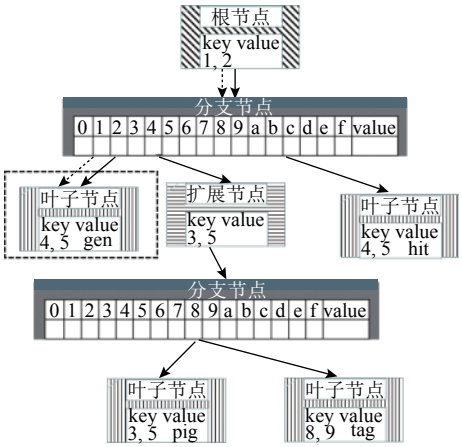

图 5 MPT 的插入

对于超级账本中的 Bucket 树, 当插入新的数据时, 需要对进行计算和合并两个过程.

如图 6 所示, 在 Bucket 树中新插入两条数据项 Entry1,通常通过以下两个步骤:

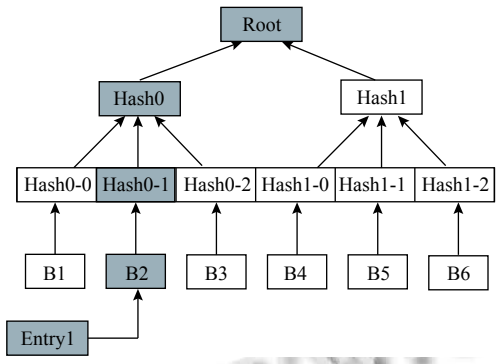

图 6 Bucket 树插入数据

计算: 经过散列值计算, Entry1 应该放到 B2 中.

合并: 在 B2 中需要将新插入的数据与历史数据进行合并, 且按固定的排序算法进行重排序, 最终得到一个新 Hash 桶, 即改变 B2 的值;

当 Hash 桶计算完成之后, 便可进行 Merkle 节点的 Hash 计算. 该过程仅对需要改变的节点进行 Hash 重计算. 对没有变化的孩子节点可直接使用历史的 Hash 值. 即需要重新计算 Hash0-1, Hash0, Root 节点.

2.2 SPV 验证

在比特币中, Merkle 的主要作用是实现 SPV 验证. SPV 验证全称叫做简单支付验证, 即区块链中的 SPV 节点 (轻型节点) 验证某个交易是否已经在交易中, 是区块链系统可以在手机等移动端运行的关键.

在 SPV 验证中, Merkle 主要完整交易的存在性检查. 其过程主要是

(1) SPV 节点从身边的全节点向获取待交易的 Merkle 分支;

(2) 利用 Merkle 分支与本地的交易生成 Merkle 根, 与本地的 Merkle 根进行比较, 验证交易的存在性.

如图 7 所示, SPV 节点验证交易 L3 的存在性, 只需要通过全节点获取 L3 的 Merkle 分支, 即<Hash1-1, Hash0>节点, 然后通过 L3 与 Merkle 分支的计算即可获得 Root, 最后通过比较 Root 和本地存储的 Merkle 根是否相同判断 L3 是否已经在区块中[16].

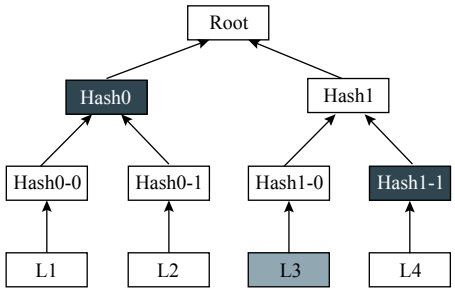

图 7 SPV 验证过程

如上分析, 使用 Merkle 分支进行存在性检查, 对于 n 个交易来说, 传输的区块数为 log(n), 可以大大较少数据传输的数量, 降低网络传输的负担.

对于以太坊来说, 因为其在前缀树的基础上集成了 Merkle 树, 因此在其 SPV 验证和比特币系统相同: 在 SPV 节点中, 也是主要存储了 Merkle 根, 之后从邻居的全节点获取 Merkle 分支, 通过哈希计算查询获取 Root, 最后和本地存在的 Merkle 根进行比较以判断交易的存在性.

对于超级账本的 Bucket 树来说, 由于其叶子结点为有序的哈希桶, 哈希桶中存在大量的数据. 如果需要实现 SPV 验证, 其不仅仅需要传输 Merkle 分支, 同时需要传输 Hash 桶中的其他元素, 会大大增加传输数据的数量, 增加网络负担. 因此, 在超级账本中很少有提及 SPV 相关的操作.

2.3 性能总结

从上述的所有内容中, 我们分析了比特币中的

Merkle 树、以太坊中的 Merkle Patricia 树以及超级账本中的 Bucke 树的结构和主要操作. 假设对于 n 个 <key,value>键值对的数据, MPT 树中每个 key 值 Hash 后的长度为 m, Bucket 树中每个节点有 a 个子节点, 叶子节点个数为 C, 那么从理论上进行分析, Merkle 树, MPT 和 Bucket 树的插入时间复杂度, 空间复杂度和 SPV 传输节点数的理论复杂度如表 1 所示.

表 1 不同种类树理论性能分析

比较项	Merkle树	MPT	Bucket 树
插入时间复杂度	O(log$n$)	O(m)	O($\log_a C$)
空间复杂度	O(n^2)	O(n^2)	O(C^2)
SPV传输节点数	O(log$n$)	O(m)	无

3 实验性能分析指标设计

在之前的工作中, 我们分析了 Merkle 树的结构和两大核心操作. 因此, 在实验中, 我们主要从插入数据性能, 数据存储和 SPV 验证性能 3 个方面进行相关的实验, 以验证之前的理论分析. 为了更好的定量表现出相关性能, 我们设计了相关的指标, 分别体现出插入数据, 数据存储和 SPV 验证的相关性能.

3.1 构建时间

首先考虑插入数据的时间开销. 对于一个相关的数据来说, 插入数据的时间极短, 其插入数据的时间开销可能比程序代码在其他地方运行的时间更短, 因此较难使用程序来统计一个数据的插入时间. 但是, 在区块链系统中, 构建区块的过程中存在非常重要的一步就是将交易池中的众多交易进行打包构建成 Merkle 树, 其本质上就是将交易池的数据一个个插入到 Merkle 树中以构成最终的 Merkle 树并存入区块中. 因此, 在本实验中, 我们采用统计不同规模数据集下 Merkle 树的构建时间来进一步的表现出不同 Merkle 树的插入性能.

3.2 节点数及树的深度

其次, 我们考虑不同 Merkle 树带来的数据存储. 目前区块过大已经是区块链系统中十分关键的问题, 而 Merkle 树作为区块中主要存储的数据, 对比不同 Merkle 树的存储性能对我们之后选择不同的区块链结构是十分有必要的.

对于 Merkle 树来说, 节点数目的大小会直接影响 Merkle 树的存储大小: 更多节点的 Merkle 树节点存储规模显然会更大. 对于不同的 Merkle 树结构来说, 相同的数据在不同的 Merkle 树结构上的节点数目和树的深度是不同的. 因此可以统计不同规模的数据在 Merkle 树上的节点数和深度来定量的表示在不同 Merkle 树的存储性能.

3.3 SPV 验证中 Merkle 分支节点数

如第 2 节所述, 区块链中的 SPV 验证是 Merkle 树中的重要功能之一, 因此检测 SPV 验证的相关性能也是对 Merkle 树性能分析的重要组成部分.

在进行 SPV 验证时, 主要关心两个性能指标. 首先是 Merkle 分支的节点数, 该指标会影响在网络传输中的性能, 太大的 Merkle 分支会造成较大网络传输的负担, 严重影响网络传输的性能. 其次, Merkle 分支树也会影响 SPV 验证的时间性能, 因为更多的 Merkle 分支节点就需要更多的重组 Hash 运算, 影响做 SPV 验证的时间效率. 因此, 在本实验分析中, 我们只需要利用 Merkle 分支的节点数目来分析对 SPV 验证的网络传输和时间的性能.

4 比较实验结果与分析

在本实验中, 我们通过实验设计, 利用 Merkle 树的构建时间、Merkle 树的节点数、树的深度和 SPV 验证的 Merkle 分支节点数来进一步分析和验证不同区块链系统中 Merkle 树的相关性能.

4.1 实验设计

本文中, 我们主要进行分析和比较比特币中的 Merkle 树, 以太坊中 Merkle Patricia 树和超级账本中的 Bucket 树. 为了消除其他因素的影响而直接分析不同 Merkle 的性能, 我们利用统一的语言 (Java) 对 3 种树的结构进行实现.

在数据方面, 为了更好的适配 MPT 结构, 我们统一使用账户模型. 如图 8 所示, 在本实验中, 我们通过随机产生的方式, 生成了数量为 1000、10 000、50 000 和 100 000 的键值对. 这些键值对的 key 值不固定大小, 其长度从 4 到 20 不等; value 的值也不固定大小, 长度从 6 到 30 不等.

针对 3 种 Merkle 树结构, 具体的实验过程为:

在 Merkle 树的实验中, 将 key 和 value 的值利用“--”字符串进行合并, 组成一个新的字符串. 之后将新的字符串的 Hash 值, 作为 Merkle 树的叶子节点.

在 Merkle Patricia 树的实验中, 首先要对 key 值进行 Hash 运算, 将 key 编成 16 字节的固定长度. 然后利

用 Hex 编码进行编码, 之后采用 MPT 树的相关算法实现完整的 MPT 树结构.

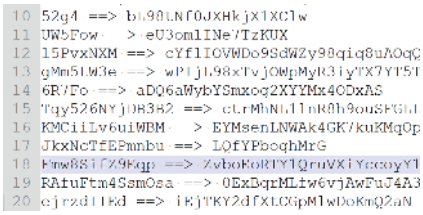

图 8 数据集格式

在 Bucket 树的实验中, 确定 100 个 Hash 桶, 每个节点最多有 3 个子节点. 然后将所有的键值随机分配到 Hash 桶中, 保证每个 Hash 桶的键值对应数目基本均匀, 最后构建 Bucket 树.

4.2 实验结果

在本实验中, 我们首先对 3 种不同 Merkle 树架构的存储进行分析. 如 3.2 节所述, 在 Merkle 树存储分析时, 我们利用树的规模作为反映存储空间的指标. 即利用树的节点数和树的深度来进行表示. 通过实验, 表 2 和表 3 表示在不同数据规模下 3 种不同 Merkle 树结构的节点数和深度.

表 2 不同规模树节点数比较 (个)

结构	1000	10000	50000	100000
Merkle树	2001	20004	99938	199750
MPT树	1709	16455	82491	167670
Bucket树	153	153	153	153

表 3 不同规模树深度比较

结构	1000	10000	50000	100000
Merkle树	11	15	17	18
MPT树	32	32	32	32
Bucket树	6	6	6	6

从表 2 中可以看出, 在 1000、10 000、50 000、100 000 组数据的情况下, Merkle 树的节点数均为最多, 分别为 2001、20 004、99 938 和 199 750 个; MPT 的节点数居中, Bucket 树的节点数最小, 均为 154 个.

再看树的深度, 从表 3 可知, 在 1000、10 000、50 000、100 000 组数据的情况下, Merkle Patricia 树的树深均为 32, Merkle 树的深度分别为 11, 15, 17 和 18, Bucket 树的深度均为 6. 因此, 在树的深度方面, Merkle Patricia 树的树深远远大于 Merkle 树和 Bucket 树两种算法. 同时 Merkle Patricia 树和 Bucket 树的树深比较稳定, 不会随着节点数的增加而增加.

其次, 如 3.1 节所述, 我们进一步对 3 种不同 Merkle 树的构建时间进行分析. 如表 4 所示, 表明了 3 种不同的 Merkle 树分别创建 1000、10 000、50 000 和 100 000 组数据的数据结构所需的时间.

表 4 不同规模树创建时间比较 (ms)

结构	1000	10000	50000	100000
Merkle树	7	42	165	330
MPT树	71	1043	23184	110663
Bucket树	2	22	305	1143

由表 4 可以看出, 在相同规模的数据下, Merkle Patricia 树的创建时间最多, 远大于其他两种 Merkle 结构; Merkle 树的创建时间居中, Bucket 树的创建时间最短.

最后, 我们需要评测 3 种 Merkle 树在做 SPV 验证. 在实验中, 我们分别取了 3 种 Merkle 树的 Merkle 路径, 如表 5 所示.

表 5 不同规模树 Merkle 分支 (个)

结构	1000	10000	50000	100000
Merkle树	11	15	17	18
MPT树	32	32	32	32
Bucket树	18	108	508	1008

从表 5 可以看到, 可以看到比特币, 以太坊和超级账本在不同规模数据下的 Merkle 路径的长度, 相比之下 3 种系统的规模有着明显的差异. Merkle 树的 Merkle 路径所需要的节点最少, 其次是 MPT 树. Bucket 树我们采用和 Merkle 树相同 SPV 验证的方法进行统计 Merkle 路径, 所需要的 Merkle 路径总体来说最长.

4.3 实验总结与分析

如 4.2 节所示, 在存储方面, 如果从节点数目来看, Merkle 所拥有的节点较多, 即所占存储较大, 其次是 MPT 树, Bucket 树所占空间最小. 从之前的理论上分析来看, 由于 Merkle Patricia 树和 Bucket 树均采取措施压缩数据, 其中 MPT 树使用前缀树算法, Bucket 树使用 Hash 桶, 故其节点数均小于 Merkle 树. 值得注意的是, MPT 树的节点数其实与 Merkle 树较为接近.

从树的深度来看, MPT 树最大, Merkle 其次, Bucket 中树的深度最小. 从理论上分析, 由于在生成 MPT 树的过程中, 首先要将 key 值 Hash 运算为 16 的长度, 同时采用 Hex 编码, 会扩大树的深度, 因此 MPT 应该是一个非常狭长的树状结构; 但是由于 key 的长度一定, 因此其深度永远不会进行增长; 而 Merkle 树的深度是数据个数的 log(n), 会随着数据的增多而增加; Bucket 树中底层的 Hash 桶数量不会发生改变, 因此其树的深

度很小, 同时其不会因为数据规模的增加而改变.

从构建时间来看, 同样是 MPT 的构造时间最长, Bucket 树的构造时间最短. 出现这种情况是由其算法的特殊性导致的: 首先, 为了提高节点的查找效率和减少存储空间浪费, MPT 树对单独存在的 key 进行了合并, 这需要占用一定时间; 其次, 为避免树中出现很长的路径并提高 MPT 树的安全性, 需要对每个 key 求 Hash 值, 这一过程需要花费大量时间. 而 Bucket 只需要进行 Hash 桶的排序和求 Hash, 上层的 Merkle 树的规模较小; 而 Merkle 树只需要进行结构的构建. 因此, MPT 树创建时间远大于其他两种算法.

从 SPV 来看, 在 3 种 Merkle 树中, 比特币中 Merkle 路径最短, 超级账本最多, 以太坊居中. 其原因是因为比特币和超级账本中只需要获取相应的路径上的节点即可, 因此其 Merkle 路径的长度应该和对应 Merkle 树的深度成正比; 而 Bucket 树底层是对有序的 Hash 桶求 Hash, 所以如果进行 SPV 验证即不仅仅需要传递路径上的数据, 同时也需要 Hash 桶中其他的数据. 因此, 在三种区块链结构中, 比特币应该最适合于在手机移动端使用, 超级账本不适合在移动端实现.

5 结论与展望

在本文中, 我们首先分析了比特币, 以太坊和超级账本这 3 种主流区块链中 Merkle 树的相关结构和核心操作, 同时根据 Merkle 树的作用和特性, 提出了相应的性能指标. 然后利用统一的语言进行了实现, 并进行了相关性能指标的检测. 实验结果表明, 比特币的 Merkle 树结构会造成更多的存储消耗, 以太坊的 MPT 结构会造成更多的时间消耗, 超级账本的 Bucket 树 SPV 验证最难. 本文的操作分析和实验环境为我们接下来的工作也奠定了基础, 接下来我们将针对 Merkle 树的特性和不足, 希望提出新的 Merkle 树结构, 在插入时间、存储或者 SPV 的验证上取得更好的发展.

参考文献

1 Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. 2009.

2 马昂, 潘晓, 吴雷, 等. 区块链技术基础及应用研究综述. 信息安全研究, 2017, 3(11): 968–980. [doi: 10.3969/j.issn.2096-1057.2017.11.003]

3 袁勇, 王飞跃. 区块链技术发展现状与展望. 自动化学报, 2016, 42(4): 481–494.

4 Sun HL, Hua S, Zhou EC, *et al*. Using ethereum blockchain in Internet of Things: A solution for electric vehicle battery refueling. Proceedings of the First International Conference on Blockchain. Seattle, WA, USA. 2018. 3–17.

5 张栋珀. 基于区块链的电能交易平台设计与实现 [硕士学术论文] 成都: 电子科技大学, 2018.

6 ElMessiry M, ElMessiry A. Blockchain framework for textile supply chain management. Proceedings of the First International Conference on Blockchain. Seattle, WA, USA. 2018. 213–227.

7 Swan M. Blockchain: Blueprint for A New Economy. Sebastopol: O'Reilly, 2015.

8 Crosby MA, Pattanayak P, Verma S, *et al*. Block Chain technology: Beyond bitcoin. Applied Innovation, 2016, 2: 6–10. [doi: 10.21626/innova/2016.1/01]

9 Zheng ZB, Xie SA, Dai HN, *et al*. An overview of blockchain technology: Architecture, consensus, and future trends. Proceedings of 2017 IEEE International Congress on Big Data (BigData Congress). Honolulu, HI, USA. 2017. 557–564.

10 Wood G. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 2014, 151: 1–32.

11 Cachin C. Architecture of the hyperledger blockchain fabric. Proceedings of Workshop on Distributed Cryptocurrencies and Consensus Ledgers. Chicago, IL, USA. 2016. 4.

12 Bonneau J. EthIKS: Using Ethereum to audit a CONIKS key transparency log. Proceedings of FC 2016 International Conference on Financial Cryptography and Data Security. Barbados. 2016. 95–105.

13 Szydlo M. Merkle tree traversal in log space and time. Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques. Interlaken, Switzerland. 2004. 541–554.

14 Delgado-Segura S, Pérez-Solà C, Navarro-Arribas G, *et al*. Analysis of the Bitcoin UTXO set. Proceedings of the FC 2018 International Conference on Financial Cryptography and Data Security. Nieuwpoort, Belgium. 2018. 78–91.

15 Dinh TTA, Liu R, Zhang MH, *et al*. Untangling blockchain: A data processing view of blockchain systems. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(7): 1366–1385. [doi: 10.1109/TKDE.2017.2781227]

16 邵奇峰, 金澈清, 张召, 等. 区块链技术: 架构及进展. 计算机学报, 2018, 41(5): 969–988. [doi: 10.11897/SP.J.1016.2018.00969]

17 Linux. Hyperledger Fabric. https://www.hyperledger.org/projects/fabric. 2019.