Computer Science and Application 计算机科学与应用, 2021, 11(4), 803-813
Published Online April 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.114082

Hans汉斯

基于改进区块存储结构的高效数据检索模型

梁保陈，张兴兰

北京工业大学信息学部，北京
Email: maillbc@qq.com

收稿日期：2021年3月9日；录用日期：2021年4月6日；发布日期：2021年4月13日

摘　要

区块链技术具有去中心化和不可篡改性等特征，能安全有效的降低新任成本且安全的存储数据，是目前计算机领域研究的热点。然而，目前区块链系统难以满足大量数据下的查询访问需求，当前研究大多是从数据同步存储到外部数据库中，通过外部数据库进行查询，增加了大量的通信量，没有解决区块链本身的问题。本文根据B+树这种结构的查询稳定性和查询路径短的优点，提出了一种解决方案，通过改进区块的存储结构，利用B+树和默克尔树各自优势，改进了默克尔树，在保证区块链验证效率的情况下增加区块链的查询性能，并且支持范围查询，减少了通信量。然后设计了基于改进区块结构的构建算法和查找算法；最后通过实验验证了所改进结构的可行性和有效性。

关键词

区块链，默克尔树，区块结构，查询

Efficient Data Retrieval Model Based on Improved Block Storage Structure

Baochen Liang, Xinglan Zhang

Information Department, Beijing University of Technology, Beijing
Email: maillbc@qq.com

Received: Mar. 9th, 2021; accepted: Apr. 6th, 2021; published: Apr. 13th, 2021

Abstract

Blockchain technology has the characteristics of decentralization and unforgeability, which can effectively reduce the new cost and store data safely, which is a research hotspot in the computer field. However, the current blockchain system is difficult to meet the demand of query and access under a large amount of data. Most of the current research is from synchronous storage of data to

文章引用: 梁保陈, 张兴兰. 基于改进区块存储结构的高效数据检索模型[J]. 计算机科学与应用, 2021, 11(4): 803-813.
DOI: 10.12677/csa.2021.114082

external database, query through external database, which increases a lot of traffic, and does not solve the problem of blockchain itself. In this paper, according to the advantages of B+ tree such as query stability and short query path, a solution is proposed. By improving the storage structure of blocks, taking advantage of the respective advantages of B+ tree and Merkel tree, the Merkel tree is improved. The query performance of block chain is increased under the condition of ensuring the verification efficiency of blockchain, and the range query is supported, and the traffic is reduced. Then the construction algorithm and search algorithm based on the improved block structure are designed; finally, the feasibility and effectiveness of the improved structure are verified by experiments.

Keywords

Blockchain, Merkel Tree, Block Structure, Query

1. 引言

区块链[1]技术起源于中本聪发表的论文《比特币：一种点对点电子现金系统》，奠定了加密货币的基础，自此比特币以及其他很多加密货币已成为行业和学术界的流行语。作为最成功的加密货币之一，比特币获得了巨大的成功，其资本市场在 2016 年已达到 1000 千万美元。区块链技术作为比特币的底层支持技术，具有公开透明不可篡改等特性，主要由密码学的技术保证安全性和可靠性。在安全性方面，主要包括数据签名技术、公钥私钥体制思想、时间戳机制；在可靠性方面，主要由共识机制来实现节点和数据的验证。解决了拜占庭[2]将军问题，例如工作量共识机制、股权证明等分布式共识机制。解决了传统的信任[3]问题与双花攻击问题，减少了传统的信任经济成本；在数据存储方面，主要采用默克尔树技术。Androulaki 等人[4]介绍了超级账本体系结构：一个可扩展的真正区块链系统，支持模块化共识协议和身份认证等，允许系统针对特定信任模型和应用场景进行定制化使用，除此之外，也是第一个支持运行标准通用编程语言运行的区块链系统，支持更多高级编程语言进行合约开发部署，并且将“智能合约”规则放在容器中运行。

文献[5]-[10]中，研究人员对区块链上数据存储的安全性问题和数据检索效率方面进行了探讨，主要是基于链上链下存储这一特点展开研究。McConaghy 等人[11]提出了 BigChainDB，这是一个大型的分布式数据库，将分布式控制和区块链技术的不变性相结合，将区块链技术引入传统数据库，提高了数据安全性，并提出了区块链数据库完整性策略。关于区块链存储容量，数据容量已增加以支持更大的数据存储。Wilkinson 等人[12]提出了端到端的区块链存储网络，用户独立于第三方数据提供者发送和共享数据，取消中央控制，并使用常规密码检查文件的可用性和完整性。Dinh 等人[13]搭建了一套系统框架用于测试区块链系统，对数据模型、共识算法和超级分类账进行了全面评估，把数据存储划分为一层、数据执行划分为一层，把共识层单独划分，各个层次独立扩展，并且测试了 UStore 数据的查询能力，帮助开发者识别系统瓶颈并相应地改进平台。Tsai 等人[14]分析了区块链应用系统在一致性、可扩展性、数据库方面的需求，以可扩展性为目标，开发了双链架构模式的“北航链”。Li 等人[15]提出一种支持排序的高效检索方法，用户可以根据需要返回 top-k 条最相关的数据，采用 MongoDB 作为存储库，并且支持范围查询，能满足不同场景的查询需求，但是 Li 等人也是利用链上链下数据库结合的方式把区块链的数据以

key-value 的形式把数据存储到数据库，利用其特性进行查询，并没有解决区块链本身的查询效率问题，并且增加了通信量。贾等人[16]提出一种存储容量可扩展模型的检索方法，设计了四层查询模型，提出 B-M 树区块结构提高了查询效率，并且根据节点的存储能力分工存储，增加缓存机制，加快了查找的速度。

本文根据目前区块链存储和检索存在的效率问题，从区块结构出发，首先提出了基于区块链系统的三层检索模型；其次，根据 B+树的查找稳定性能，查询路径短的优点，结合默克尔树的优良验证性能，改进了默克尔树的存储结构，在节点上添加相应的索引数据，在保证验证效率的前提下提高了数据检索的能力，并且支持范围查询，设计了对应的构建算法和查询算法；最后，通过实验测试了提出树结构的有效性和可用性，有良好的查询性能和稳定性能。

2. 预备知识

2.1. 区块链结构

区块链是明显的防篡改数字账本，以分布式方式(即没有中央存储库)并且通常没有中央授权机构(即银行，公司或政府)来实现。区块结构如图 1 所示，一般区块链中，它们使用户能够在区块链网络内共享交易，从而在区块链网络的正常运行下，一旦发布交易就无法更改。狭义上讲，区块链是一种按照时间顺序将数据区块以顺序相连的方式组成的一种链式结构，并以密码学方式保证的不可篡改性和不可伪造的分布式账本。广义上讲，区块链技术是利用块链式数据结构来验证和存储数据、利用分布式节点共识算法来生成和更新数据、利用密码学的方式保证数据传输和访问的安全性、利用由自动化脚本代码组成的智能合约编程和操作数据的一种全新的分布式基础架构与计算范式。区块链是多种已有技术的集成创

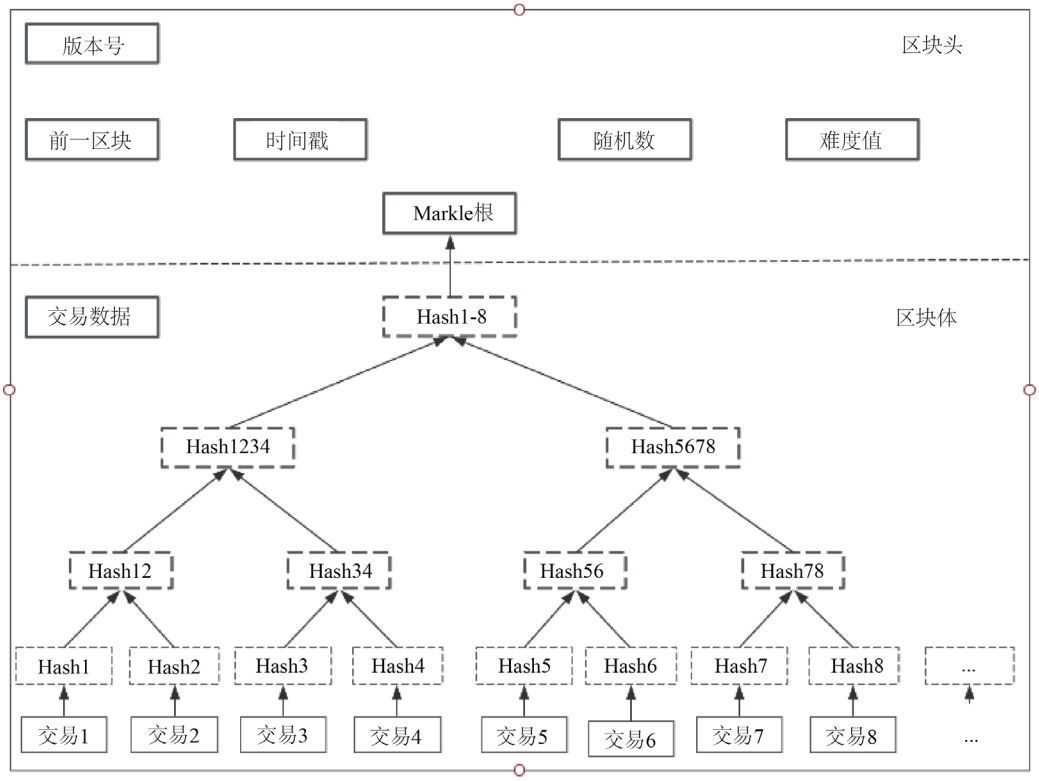

Figure 1. Block structure
图 1. 区块结构

新，主要用于实现多方信任和高效协同，主要具有透明可信、防篡改可追溯、隐私安全保障以及系统高可靠等特性。

2.2. B+树

B+树是 B 树的变体，应用比较广泛，用户的文件索引、专用数据库和通用访问方法都是基于 B 树和 B+树提出和实现的。在 B+树中所有的键值都位于叶子节点，上层非叶子节点包含一个索引，能够快速定位数据位置。叶子节点通过链表从左向右链接。

一棵 m 阶 B+树的定义如下：

- 每个节点最多可以有 m 个子树(分支节点)；
- 分支节点最少有$\lceil m/2 \rceil$棵子树(根节点除外)；
- 节点的子树个数与关键字个数相等；
- 叶节点根据关键字排序通过指针相连。

如图 2 所示为一棵 4 阶 B+树结构示意图。

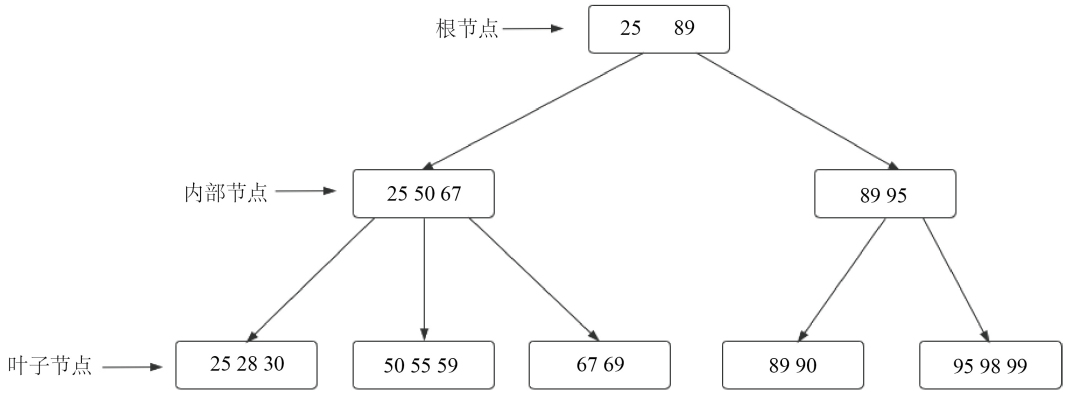

Figure 2. Structure diagram of fourth order B+ tree
图 2. 四阶 B+树结构示意图

3. 模型设计

数据检索模型主要划分为 3 层，分别包括用户层、查询层和存储层。如图 3 所示。

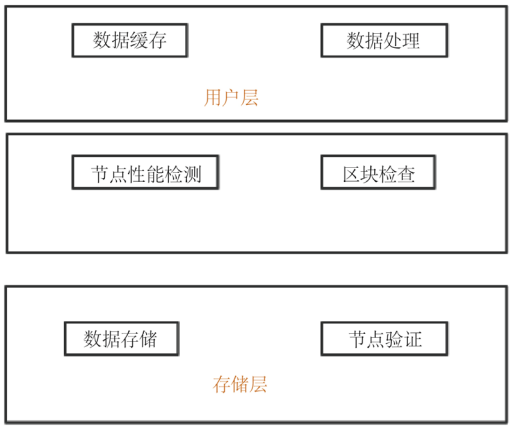

Figure 3. Model design
图 3. 模型设计

3.1. 用户层

用户层主要处理数据和接收查询请求。收到数据搜索请求后，一个完整的搜索从用户层开始，先从用户层的历史查询数据中查看是否已有查询记录和结果，如果存在可以直接返回所需数据，搜索完成；如果不存在则需要进入查询层继续查询。

传统的数据库使用 B+树作为查询结构，并且将一次查询的结果认为下次可能也会再次被查询，所以采用缓存机制来提高性能，所有缓存的数据基本都存在内存中，相比于去读写磁盘操作，减少了 IO 时间，数据访问结果返回更快。基于这个思想，在用户层设置缓存模块，用户层在节点完成查询之后，把结果缓存在缓存模块中，方便下次查询相同数据时，增加检索速度。

数据处理模块主要是用户节点完成查询对查询结果的数据进行处理，比如一次转账交易的参与方地址，交易金额等数据进行处理，方便下次查询。

3.2. 查询层

查询层根据节点的稳定性以及计算能力分工处理查询业务和普通的节点验证业务，查询数据返回给用户层。

基于 P2P 网络中的节点查找技术，并且对区块链网络中的节点划分为两种，超级完全节点和普通验证节点。超级节点通过以下公式计算产生，主要通过在线时间和稳定性计算，为节点计算评分(score)，拥有评分高的节点选举为超级节点：

$$\text{score}[i]=\alpha\times\text{Capacity}[i]+\beta\times\text{Age}[i](\alpha+\beta=1)$$

Capacity 代表节点的计算能力和稳定性，Age 代表节点的在线时间。其他参数信息代表的意义是：BandWidth-带宽、CpuSpeed-Cpu 速度、Storage-存储空间、Netspeed-节点网络环境速度、Time-节点在线时长、*t*-节点上线次数。

$$\text{Capacity}[i]=a\times\text{Bandwidth}[i]+b\times\text{CpuSpeed}[i]+c\times\text{Storage}[i](a+b+c=1)$$

$$\text{Age}[i]=\left(1/\text{Netspeed}[i]\right)\times\left(\text{Time}[i]/t[i]\right)$$

当进行查询操作时，由于超级节点具有稳定性、计算能力强的特点，仅使用超级节点查询，发现算法在超级节点之间转发。普通计算节点负责节点稳定性的检测，主要包括节点评分的计算，为超级节点选择候补节点。

3.3. 存储层

存储层位于最底层，主要负责存储区块链数据，对查询层的查询请求做出响应。针对区块的存储结构，提出了基于改进的默克尔树的存储结构，结合了默克尔树和 B+树的特点，默克尔树能很快的验证数据是否存在树中以及是否数据被篡改，B+树能在极短路径下查询到数据，并且支持范围查询，提高查找效率，具体改进结构在下一节详细介绍。

4. 改进的区块存储结构

目前，区块链中的数据存储结构主要是默克尔树，默克尔树的主要思想是基于单向密码哈希函数构造一棵树，每个叶节点都可以通过验证路径信息进行验证，由于仅计算哈希函数，因此验证的成本非常低。因为默克尔树的特性。是每层是数据的哈希值，无法利用二叉树的特点优化查找，只能遍历全部数据。随着区块链链上数据快速增长，查询一条数据的效率越来越慢，无法满足要求。因此根据 B+树这种结构的查询稳定性和查询路径短的优点，提出了改进的区块存储结构，利用 B+树和默克尔树各自优势，

改进了默克尔树，在保证区块链验证效率的情况下增加区块链的查询性能，并且支持范围查询，设计了构建算法和查询算法。

基于 M_H+树结构的区块链存储结构结合了默克尔树的高效验证和 B+树的快速检索的特点，数据结构如图 4 所示。节点主要改进了默克尔树的存储结构，主要有哈希值和索引地址和端点索引值大小几个部分组成。

Figure 4. New data structure
图 4. 新数据结构

建立 M_H+树的过程主要是，首先确认系统在一个区块里写入的数据，即打包的交易信息；然后根据索引数值的大小构建一棵树，这里比如以每个用户的身份地址数值大小作为索引，建立起一个 B+树；根据建立的树的结构，从底层开始把叶子节点哈希值合并，计算组成新的哈希值，同时记录用户地址的最大值和最小值；重复直到生成一个树根哈希，得到的根哈希作为区块头的一部分数据保存。

由于 B+树结构只有叶子节点存储数据信息，其他节点存储的都是索引信息，通过索引信息查找到叶子节点，叶子节点在最后一层通过链表的形式相连，并且使有序的，在范围查找时只要找到第一个符合条件的叶子节点，顺序往后遍历即可查找到所有节点，节省了查询时间和查询次数。根据其存储结构的特点，索引信息通常占用的空间小，因此相对来说整体的节点占用的空间缩小，能存储更多的数据。

算法：B+树插入操作

这里设插入的关键字为 key。

1) 如果是空树，则新建一个叶子结点，把记录插入叶子节点，此时这个叶子结点也是根结点，插入操作结束；

2) 对于节点类型为叶子的：通过比较 key 值大小定位到要插入叶子节点的位置，判断插入当前节点后叶子节点关键字的个数是否会影响 B+树的性质，即若关键字个数不大于 $m-1$，插入成功，结束此次操作；如果关键字个数大于 $m-1$，需要把当前节点一分为二，其中一个新节点的关键字为前 $m/2$ 个，另一个新节点插入余下的关键字，需要在上一层索引节点即父节点插入第 $m/2+1$ 个记录作为新的索引，新索引的左右指针分别指向左右节点，进行下一步操作；

3) 对于节点类型为索引的：判断该节点的关键字个数是否大于 $m-1$，如果关键字个数不大于 $m-1$，此次操作完成，如果关键字个数大于 $m-1$，该节点一份为二，此时分成两个索引节点，其中一个索引包含 $(m-1)/2$ 个关键字，另一个节点包含 $m-(m-1)/2$ 个关键字，同时需要在上一层索引节点即父节点插入第 $m/2$ 个关键字作为新的索引，新索引的左右指针分别指向左右节点，重复执行直到完成插入。

如图 5 为 B+树插入示意图，依次插入关键字 40、45、62。

算法：M_H+树建立算法

Input：:构建好的 B+树结构；

Output：M_H+树结构存储模型；

1. 确认构建好的 B+树结构；
2. 据 B+树的结构，从最底层节点处理数据;
3. for(B+树节点 N){
4. if(叶子节点){
5. 哈希数据；

6. }
7. else{
8. 最小值索引获取；
9. 最大值索引获取；
10. 记录此节点的 minval、maxval 值；
11. 数据哈希；
12. }
13. N = N − 1；
14. }
15. 最后得到的根作为结果保存在区块头中；

以用户地址为 25 28 30 50 55 59 67 69 89 95 98 99 说明构建 M_H+树的过程，该节点构成的 B+树结构如图 6 所示。

根据 B+树构建的 M_H+树如图 7 所示：

根据带索引信息的节点，并且根据默克尔树的校验特征，可以快速校验信息检索结果正确与否。

查询算法如下，如果有节点发起检索请求操作，首先检查节点是否为稳定的且计算能力强的超级节点，如果满足则直接查询；如果不满足，则请求超级节点查询。在一个节点查询数据时，首先按照区块

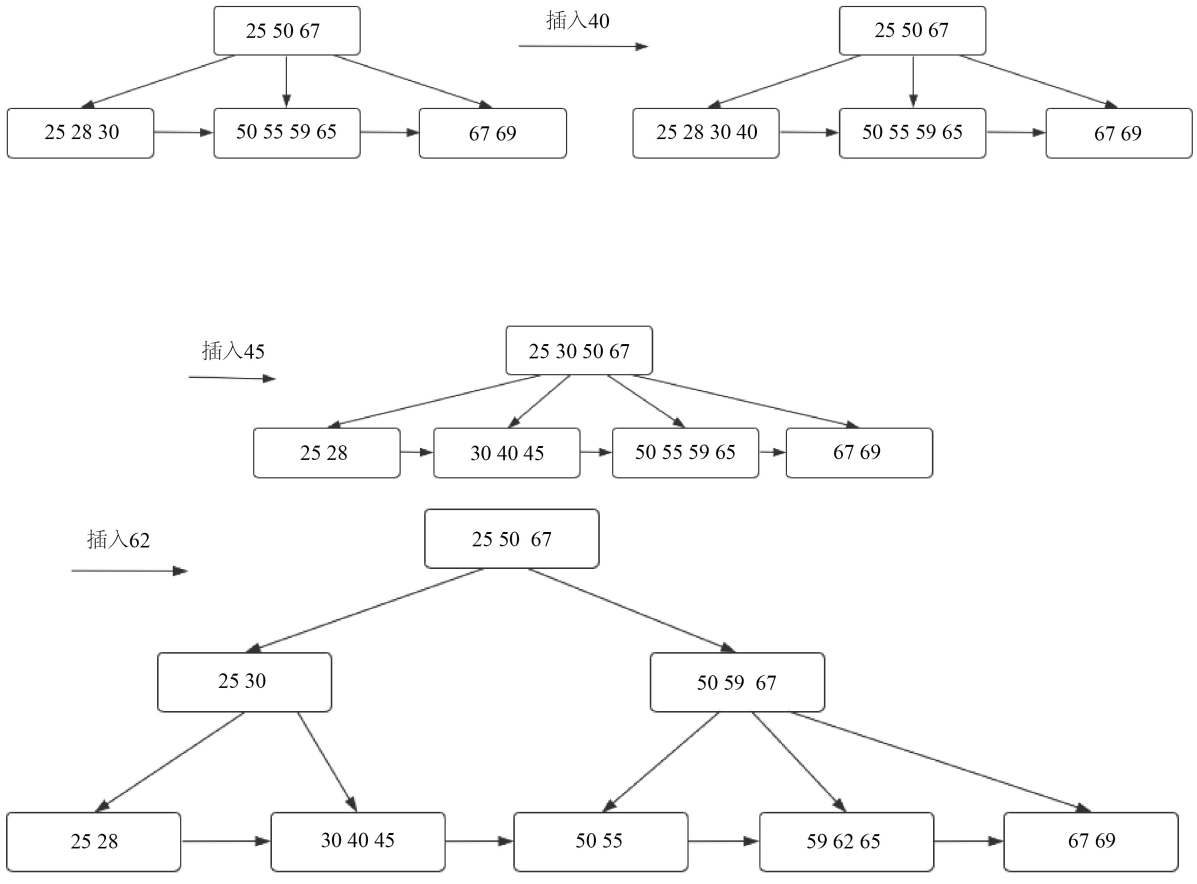

Figure 5. Schematic diagram of B+ tree insertion
图 5. B+树插入示意图

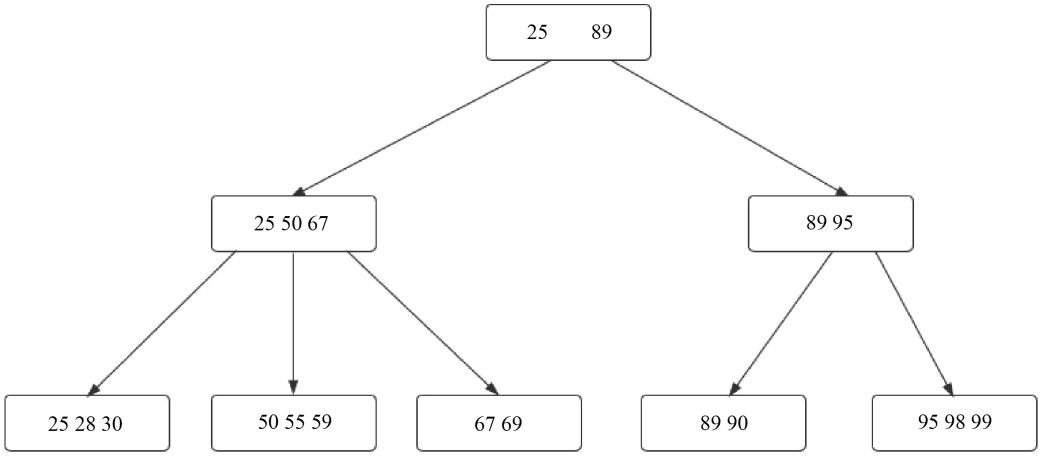

Figure 6. B+ tree structure
图 6. B+树结构

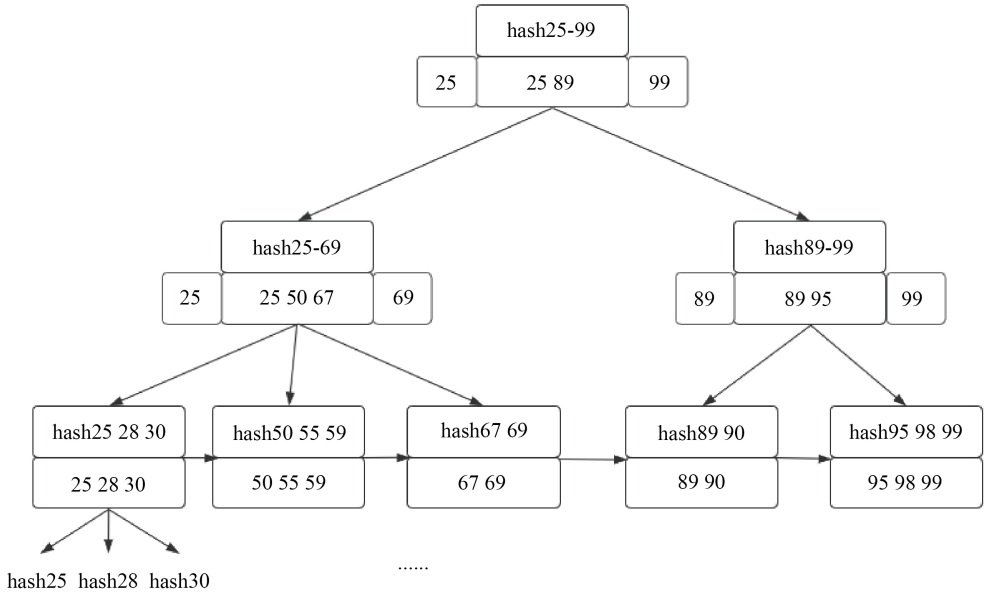

Figure 7. M_ H+ tree structure
图 7. M_H+树结构

顺序遍历区块头中的数据，找到 M_H+树根，根据根节点中的索引信息大小(索引的最大值和最小值)判断查询的数据是否在这个区块中：如果要检索的数据索引不在这个范围内，继续查询下一个区块，如果满足在这个范围内，则根据索引的指向地址继续往下一层检索，直到找到叶子节点是否满足条件，搜索一颗完整的树结构，如果未查询到相关结果，搜索下一个区块。

如图 7 所示的 M_H+树中，要查找索引地址为 69 的数据，首先从根节点判断，索引范围 25~99，满足 69 所在的区间，根据访问根节点(25 89)，寻找第一个孩子节点，找到子树的节点范围 25~69 的节点，满足区间，根据(25 50 67)节点信息，找到第三个子节点，遍历找到结果并返回数据信息给查找节点。

算法：M_H+树查询算法

Input：区块链数据，查询条件索引 Index；

Output：交易结果。

```
1. If (节点不满足稳定条件){
2.  找到邻近的超级节点
3. }
4. For (遍历区块){
5.  访问区块头树根 minval, maxval 值；
6. while(minval≤Index≤maxval){
7.      If(叶子节点){
8.          For(遍历叶子节点){
9.              If(找到 index){
10.                 Return 数据；
11. }
12. }
13. }
14.  获取新的节点信息；
15.  更新 minval、maxval;
16.   }
17.  访问下一个区块；
18. }
```

5. 分析

对于区块链一次查询操作时，时间复杂度为$O(n_P)$，n_P 为区块链系统中节点个数。基于本文设计的三层查询模型，通过向超级节点发起检索请求，由超级节点进行检索操作，时间复杂度为$O(n_{sP})$，n_{sP} 为超级节点个数。在数据存储层，基于 M_H + 树的结构查找时间复杂度为 log 对数级别，普通区块链查询需要遍历完整的区块数据，时间复杂度为$O(n_c)$，n_c 为一个区块内交易的总数。在通信量方面，与采用链上链下结合的区块链相比，减少了去链下寻找数据的通信次数，很大程度减少了通信量，节约计算成本。

实验的配置为 Intel SkyLake 6161 2.2 GHz CPU 和 2 GB 运行内存的云服务器，操作系统为 centos 8.0 64 位，借助 Bitcoin 的源码搭建了一个简单的区块链系统，通过修改区块的块内部结构，主要是区块头和区块体的结构，交易的组合形式由默克尔树改为第四章提到的改进结构 M_H+树；保留了非对称加密和 UTXO 的思想。主要测试区块链改进数据结构的查询方法的正确性和高效性。本节中，我们设计多组实验测试所提出的检索性能，主要参考为运行的时间，并把算法运行 50 次平均值作为算法的运行时间。

在生成区块时，需要对交易两两哈希得到默克尔根，本模型把结构换成 M_H + Tree，并且根据 B+树的特性，添加了相应的索引数据。设计了实验测试当交易数量为 128，256，512，1024，2048，4096，8192，16384，32768，65536 时生成树结构的时间，如图 8 所示，默克尔树和 M_H+ Tree 的构造时间都随着数量增加而线性增加，性能相当，说明在可接受的时间范围内实现了树的构建，增加了范围查询的功能，并且减少了通信量。这里由于 M_H+树在生成的同时增加了一部分索引信息的生成，所以时间代价多于默克尔树，不过相差甚微，可以忽略。

比特币系统查询只能根据交易哈希查询，在一个区块内顺序遍历所有数据，时间复杂度为 O(n)，随着数量增大呈线性增长，而基于 M_H+树结构的查询时间比较稳定，并且低于默克尔树的时间，如图 9

所示，这是由于所构建的树结构，在给定实验数据的交易量大小对查询时间的影响可忽略不计。

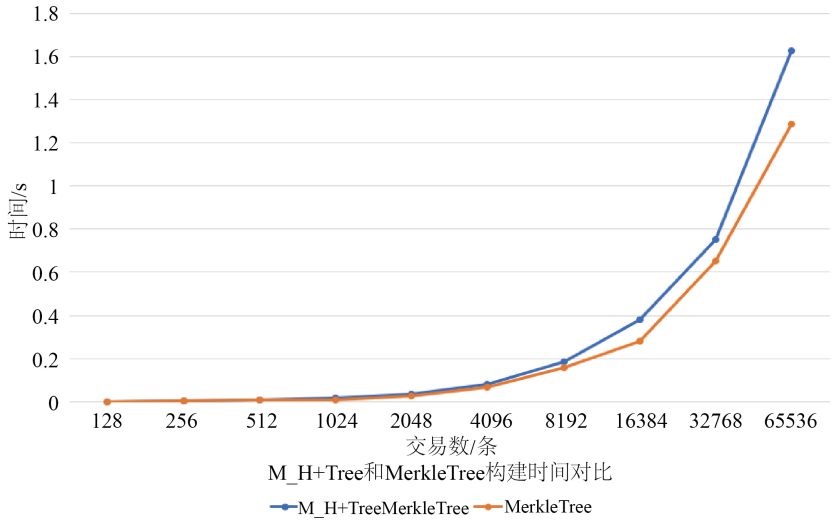

Figure 8. Construction time comparison
图 8. 构建时间对比

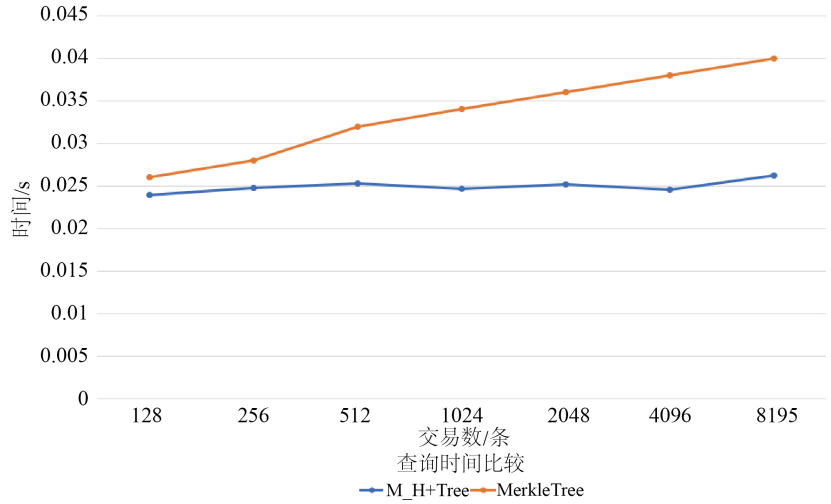

Figure 9. Query time comparison
图 9. 查询时间对比

6. 总结

本文提出了基于默克尔树改进的区块存储结构，主要在不影响区块链交易验证效率的情况下提高交易数据的检索效率，其次设计了基于区块链的查询三层模型，在用户层增加了缓存机制，用于处理已查询数据的。和传统的利用区块链和链下数据库结合的存储方式，减少了数据的通信量，并且提高了检索效率，具有良好的性能。区块链凭借着安全不可篡改的优势，应用场景越来越广，对数据的存储和查询效率要求越来越高，未来研究中将继续深入研究区块链的查询优化技术。

基金项目

基金项目：国家自然科学基金(61801008)。

参考文献

[1] Nakamoto, S. (2019) Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Lamport, L., Shostak, R. and Pease, M. (2019) The Byzantine Generals Problem. In: Malkhi, D., Ed., *Concurrency*: *The Works of Leslie Lamport*, Association for Computing Machinery, New York, 203-226. https://doi.org/10.1145/3335772.3335936

[3] Anjum, A., Sporny, M. and Sill, A. (2017) Blockchain Standards for Compliance and Trust. *IEEE Cloud Computing*, **4**, 84-90. https://doi.org/10.1109/MCC.2017.3791019

[4] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., *et al*. (2018) Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the* 13*th EuroSys Conference*, Porto, April 2018, Article No. 30. https://doi.org/10.1145/3190508.3190538

[5] Wang, Z., Fu, Y., Zhong, L. and Dai, F. (2020) Research on Blockchain Availability Modeling in P2P Network. *International Journal of Advanced Network, Monitoring and Controls*, **5**, 36-43. https://doi.org/10.21307/ijanmc-2020-006

[6] Chen, Y., Ding, S., Xu, Z., Zheng, H. and Yang, S. (2019) Blockchain-Based Medical Records Secure Storage and Medical Service Framework. *Journal of Medical Systems*, **43**, Article No. 5. https://doi.org/10.1007/s10916-018-1121-4

[7] 余涛, 牛保宁, 樊星. FabricSQL: 区块链数据的关系查询[J]. 计算机工程与设计, 2020, 41(10): 2988-2995.

[8] 吕建富, 赖英旭, 刘静. 基于链上链下相结合的日志安全存储与检索[J]. 计算机科学, 2020, 47(3): 298-303.

[9] Sahoo, M.S. and Baruah, P.K. (2018) HBasechainDB—A Scalable Blockchain Framework on Hadoop Ecosystem. 2018 *Asian Conference on Supercomputing Frontiers*, Singapore, 26-29 March 2018, 18-29. https://doi.org/10.1007/978-3-319-69953-0_2

[10] Hofmann, A. (2020) Building Scalable Blockchain Applications—A Decision Process. 2020 *International Conference on Design Science Research in Information Systems and Technology*, Kristiansand, 2-4 December, 309-320. https://doi.org/10.1007/978-3-030-64823-7_28

[11] McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McMullen, G., Henderson, R., *et al*. (2016) BigchainDB: A Scalable Blockchain Database. BigchainDB, Berlin.

[12] Wilkinson, S., Boshevski, T., Brandoff, J. and Buterin, V. (2014) Storj: A Peer-to-Peer Cloud Storage Network.

[13] Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C. and Tan, K.-L. (2017) BLOCKBENCH: A Framework for Analyzing Private Blockchains. *Proceedings of the* 2017 *ACM International Conference on Management of Data*, Chicago, May 2017, 1085-1100. https://doi.org/10.1145/3035918.3064033

[14] Tsai, W., Yu, L., Wang, R., Liu, N. and Deng, E.Y. (2017) Blockchain Application Development Techniques. *Journal of Software*, **28**, 1474-1487.

[15] Li, Y., Zheng, K., Yan, Y., Liu, Q. and Zhou, X. (2017) EtherQL: A Query Layer for Blockchain System. 2017 *International Conference on Database Systems for Advanced Applications*, Suzhou, 27-30 March, 556-567. https://doi.org/10.1007/978-3-319-55699-4_34

[16] 贾大宇, 信俊昌, 王之琼, 郭薇, 王国仁. 存储容量可扩展区块链系统的高效查询模型[J]. 软件学报, 2019, 30(9): 2655-2670.