第二，利用 RocksDB[9]将数据以链式结构保存在本地文件，将运行时区块链数据结构转为数组结构，以提高查询效率。

第三，优化了溯源数据的上链流程。引入了“缓存池”的概念，先将未构成完整溯源链的数据存在池中，只有构成完成溯源链的时候才将完整的溯源数据上链。这样查询溯源数据的时候就不需要从后往前推出所有溯源数据，只需要一次查询便可以完成取出全部溯源数据，极大提高了溯源查询效率。

第四，以商品的溯源为背景设计实现了商品溯源系统。

第五，设计实验，将本文中优化后区块链查询方法与传统的区块链查询方法进行比对，验证优化方案可行性。

2 改善区块链结构的查询优化

在本章将深入讨论传统区块链的底层结构与查询性能之间的联系，以及其改进方法。

2.1 传统区块链查询方法

2.1.1 传统区块链结构

目前主流区块链系统的区块基础结构[10]如图 1 所示，在区块头中包含三组数据。第一组是通过哈希计算得出的 Current Hash 和指向父区块的 Pre Hash，其值为父区块的 Current Hash。通过 Current Hash 和 Pre Hash 链接区块链中的当前区块和前一区块；第二组由随机数（Nonce）、难度目标（Bit）和时间戳组成。这组数据与区块挖掘有关。随机值定义了区块挖掘难度，难度目标是指给定难度目标的 Hash 值，时间戳是指区块生成的时间，精确到秒；第三组数据是 Merkle 树根，它是本区块所包含的所有交易对应的 Hash 值两两循环计算得出的 Hash 值。

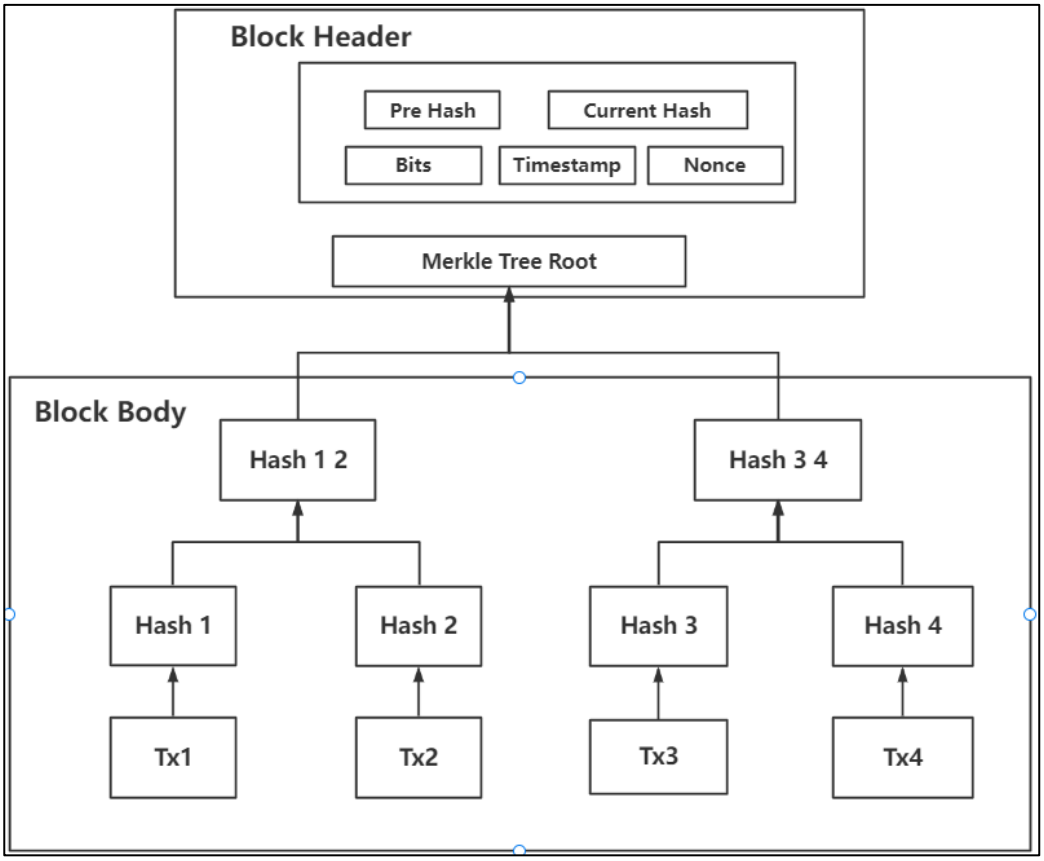

图 1 比特币区块结构图

在进行交易验证时可以通过计算 Merkle Tree Root 的 hash 值来确认区块中的数据是否合法，有无被篡改。假设交易 Tx1 交易出了问题时，则可以通过 Merkle Tree Root 到 Hash1 2 到 Hash1 最后到 Tx1 的这条 Merkle proof 来进行非法数据的定位。

2.1.2 传统区块链查询方法

在传统块索引结构的基础上，深入研究了传统区块链查询方法[11]，并给出了算法实现。通过对时间复杂度的计算，指出了传统区块链查询方法的不足之处，与之后的优化方法进行对比，体现优化后的方法具备的使用价值。

如图 2 所示，为区块链结构。区块链是由区块的 Hash 值相链接一起的一种链表结构。当前的区块链索引结构只支持相对简单的查询，并且只支持基于唯一标识 Hash 值的查询。在区块链的链表结构中的查询某一数据最坏情况需要历遍整条区块链的数据才能查到数据，最优的情况只需要访问

最新区块就可以查到，故其时间复杂度为 O(N)，其中 N 为区块数量。找到对应区块后需要历遍区块体中的 Merkle 树结构的叶子结点，最后找到对应数据。在历遍 Merkle 树叶子结点需要的时间复杂度为 O(2(n-1))其中 n 为交易数量。故总的查询时间复杂度为 O(2(n-1)•N)，忽略常数项则查询时间复杂度为 O(n•N)。

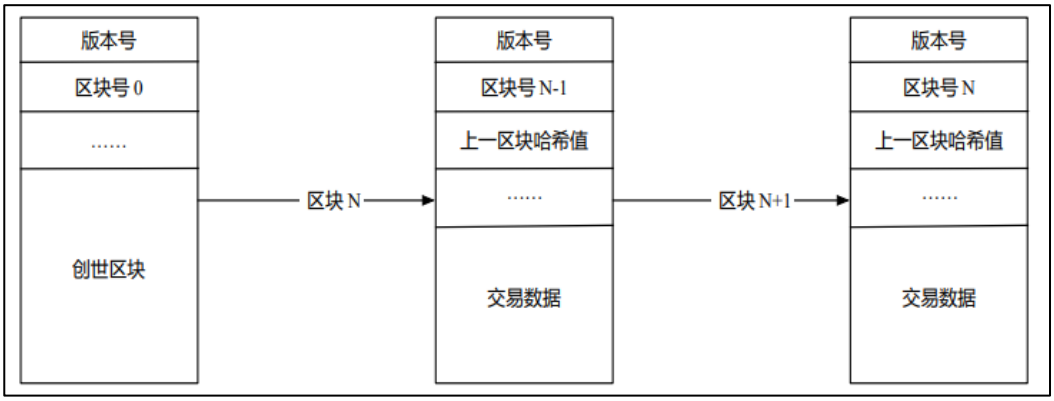

图 2 区块链结构示意图

区块链的传统查询算法如，算法 1 所示

算法 1: 基于传统区块链结构的溯源查询

```
Require: Hash value h of transaction ticket number of transaction
Ensure: Details of transaction result
1. initialize the storage transaction details array result;
2. n = Block().num;
3. while n > 0 do
4.       for j = 0 to block[n].transaction.num then
5.             if  h == block[n].transaction[j].hash then
6.                 result.add( Block[n].transaction[j]);
7.                 n--;
8.             end
9.       end
10.       n = n  -  1;
11. end
12. if result.num == 0 then
```

```
13.     return None;
14. else
15.     return result;
16. end
```

算法 1 先初始化了存储结果数据的数组 result；在之后的外层 while 循环进行对区块的历遍，在内层的 for 循环中历遍该区块包含的交易数组。若找到结果则将数据存入 result 中。若 result 数组包含数据量为 0 则返回空的结果，表明未查询到数据，否则输出 result 数组。

2.2 基于优化区块链底层结构的溯源查询方法

为了在优化区块链查询性能的目标基础上保证区块内数据的不可篡改性与可验证性，同时不增加区块链系统的复杂度。本文选择从区块链底层的数据结构入手，以改善区块链的溯源查询性能，拓展其查询功能。

2.2.1 优化区块链结构的溯源查询方法

区块链的链表结构主要是通过前后区块的 Hash 指针值进行匹配链接，是为了保证区块链的不可篡改性与安全性。而区块对应的 Hash 值是通过区块内部本身数据计算得出的，再加上区块链上的区块是按照时间顺序添加到区块链上的。由此可以将“链”这一结构抽象化，把区块视为一个整体对象，区块链为一个包含合法区块的可变长数组。如图 3 所示：

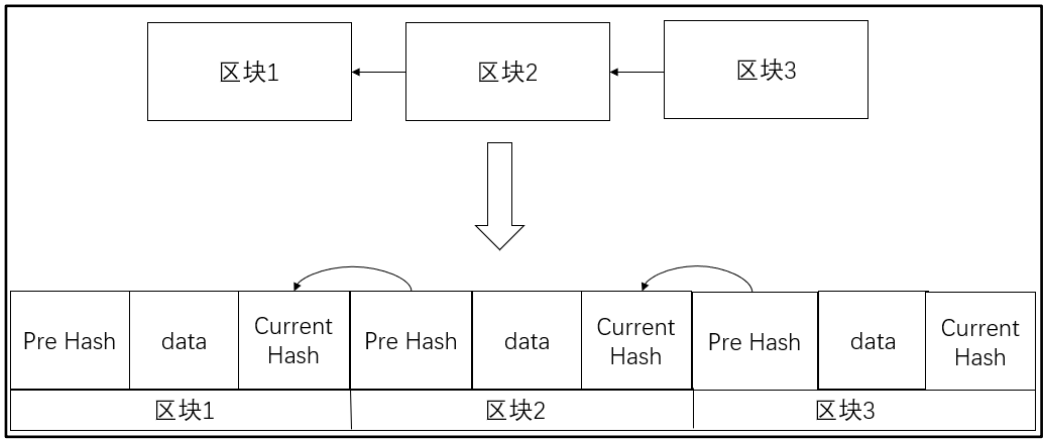

图 3 区块链结构转换示意图

将链表结构转为数组结构，数组中存储“区块”对象，根据区块头中的前区块 Hash（Pre Hash）和当前 Hash（Current Hash）进行抽象的“链接”。将“链”的概念抽象化，由于 Pre Hash 和 Current Hash 都是根据区块数据计算得出的，所以该方案同时也保证了数据的不可篡改性。其中区块所存数据（data）具体为如图 1 所示。

由于将区块链底层改成了数组结构，访问区块链任意区块的时间复杂度的从 O(c1•n)变为 O(c2•n)，n 为区块链包含的区块数量，c1，c2 为常数项，其中 c1 小于 c2，优化了区块链的查询性能，还使得其有了随机访问任意区块能力。

同时在区块链系统中引入了缓存池的概念，假设一件商品有生产、分销、零售等流程。则设立生产缓存池，分销缓存池，分别保存商品从生产数据，分销数据。当有商品成功零售的时候，则将对应商品的生产缓存池和分销缓冲池中的商品数据取出与零售数据构成完整的溯源数据再将完整的溯源数据进行上链。这样在进行溯源查询时无需历遍全部区块的数据，只需要找到存在的唯一完整的溯源数据，减少了区块的查询次数，优化了区块链溯源查询性能。

2.2.2 优化 Merkle 树结构的溯源查询方法

Merkle 树[12]本质是一种二叉树结构，曾经在文件系统和 P2P 网络系统中广泛应用。

如图 4 所示，Merkle 树具备以下特点：Merkle 树的叶子结点存储数据或其对应 Hash 值;Merkle 树的非叶子结点存储由其子结点中数据通过 Hash 计算得出的 Hash 值。

Merkle 树在很多应用场景下都有着极为亮眼的性能表现，具体有以下三种：

第一，比较大量数据：Hash 值的计算快，而 Merkle 树通过叶子结点，从下往上循环计算至根结点的 Hash 值进行构建，上层结点 Hash 依赖于下层结点的 Hash 值。故只需比对构建出的 Merkle 树的树根 Hash 值与原 Merkle 树根 Hash 值是否一致，就可以在短时间内比较数据。

第二，快速定位修改：在图 4 中，数据集中的 L1，L2，L3，L4 都有唯一的 Merkle proof 到 Top Hash，若 L1，L2，L3，L4 任意数据发生改变，都会影响到 Top Hash。当 Top Hash 值发生变化的时候，就可以沿着 Top Hash 最多使用 O(logn)的时间通过 Merkle proof 定位到发生变化的数据。

第三，零知识证明：如下图 4 所示，可以通过 Merkle proof 证明数据集（L1，L2，L3，L4）的某个数据 L2 存在，而不用暴露其他数据。

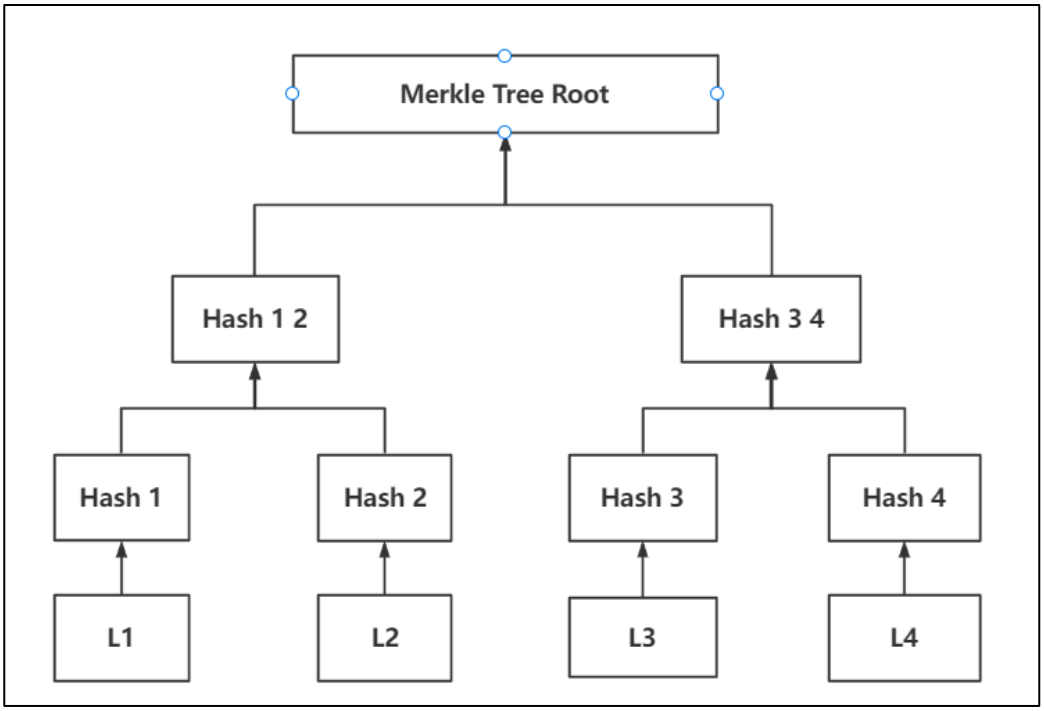

图 4 Merkle 树示例图

为了优化区块链的溯源查询性能，本文对区块链中 Merkle 树的结构进行了优化，引入了布隆过滤器（Bloom Filter）[13]这一数据结构。

Bloom Filter 本质上是一个很长的二进制向量和一系列随机映射函数，用于快速确定一个元素是否在一个集合中。在初始状态 Bloom Filter 的所有位都被置为 0，如下图 5 所示：

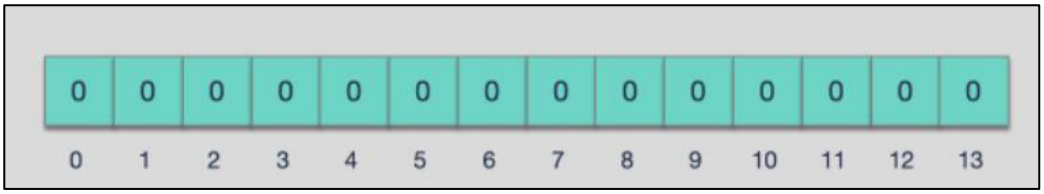

图 5 Bloom Filter 初始状态

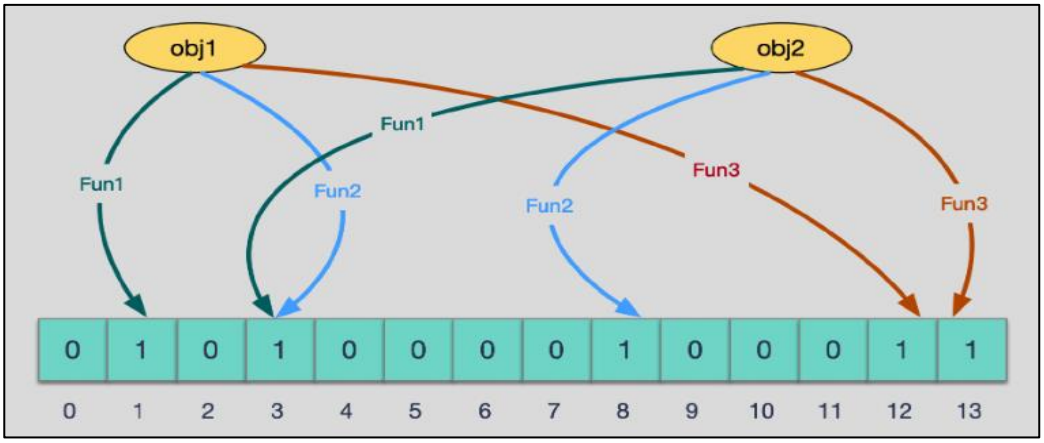

图 6 Bloom Filter 映射数据到位数组示意图

将变量添加到集合中时，该变量将通过 K 个映射函数映射到位数组中的 K 个点，并将其值设置为 1（假设有两个变量，每个变量通过三个映射函数）。根据 Bloom Filter 的特性，如图 6 所示在判断变量 obj1 是否存在集合中只需看二进制向量数组中下标为 1、3、12 对应值是否为 1。若其中任意一个下标对应值为 0，则 obj1 变量不存在该集合中，若对应值全为 1，则变量很可能存在。因为映射函数本质上是哈希散列函数，存在哈希碰撞，如图 6 中，变量 obj1 和变量 obj2 通过映射函数 Fun2 都会映射到二进制向量数组下标为 3 的元素中。

同时与其他数据结构相比，Bloom Filter 在空间和时间上都有很大的优势。Bloom Filter 存储空间和插入及查询时间是常量的 O(k)，k 为常量。Bloom Filter 不需要存储元素本身，这有效保证了数据的隐私性、安全性。Bloom Filter 与其他数据结构相比最大的一个特点是可以表示数据全集，基于这个特点，可以丰富区块链溯源查询性能的语义查询功能让其不只仅支持交易的 Hash 等语义简单的查询。

优化后 Merkle 树结构如图 7 所示，其中 BF 为 Bloom Filter 简写，BF（1,8）表示该结点的 Bloom Filter 完成了交易 Tx1 到 Tx8 信息到位数组的映射。（Tx1…Tx8)为交易数据列表，（Hash1…Hash8）为叶子结点。该 Merkle 树的叶子结点存储真实数据对应的 hash 值，非叶子结点存储其左右子结点计算得出的 hash 值。

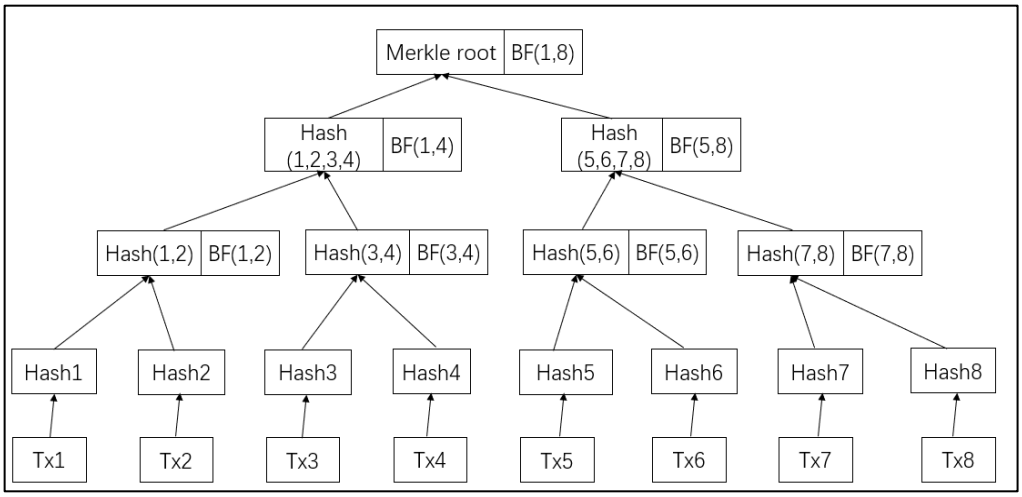

图 7 结合 Bloom Filter 的 Merkle 树结构

优化后 Merkle 树构建算法如算法 2 所示

算法 2: 优化后 Merkle 树构建算法

Require: Transaction list T

Ensure: Improved Merkle tree

```
1. for i = 0 to T.num do
2.     MerkleTree.leafnode = <Hash(T[i])>
3. end
4. for i = leafnode.Size() to 1 do
5.     i = (i+1)/2;
6.     for j = 1 to i do
7.         Hash(i,j) = Hash(Hash(Hash.l|Hash.r)|BF(i,j));
8.         MerkleTree.node = <Hash(i,j),BF>
9.     end
10. end
11.return MerkleTree;
```

该算法的 1-3 行计算 Merkle 树叶子结点对应的 Hash 值，4-10 行根据叶子结点的 Hash 值循环构建上层非结点直至 Merkle 树的根结点，最后输出结构优化后的 Merkle 树。

由于该方案只在 Merkle 树本身结构上增加了 Bloom Filter 结构，不影响 Merkle 树本身的特性，故其仍具有防篡改、比对数据、快速定位、零

知识证明等功能。

根据结构优化后的 Merkle 树设计查询算法[14]，如算法 3 所示：

算法 3: 基于优化 Merkle 树的查询算法

```
Require: transaction index
Ensure: Details of transaction result
1. for i = (Block.num - 1) to 0 do
2.     temp = Block[i].MerkleRoot;
3.     if temp.BF.contain(index) then
4.         while (temp not LeafNode) do
5.             if temp.LeftNode.BF.contain(index) then
6.                 temp = temp.LeftNode;
7.             else
8.                 temp = temp.RightNode;
9.             end
10.        end
11.    else
12.        continue;
13.    end
14.    if Hash(index) == temp.Hash then
15.        result = temp.information
16.    end
17. return result;
18. end
```

在算法 3 中，通过外层 for 循环历遍全部区块，在历遍区块链时查看每个区块的 Merkle Root 的 Bloom Filter 进行数据的快速判断检测，如果不包含数据则跳过当前区块，去查询下一区块。不需要历遍区块内全部数据，提高了查询效率。若 Merkle Root 中 Bloom Filter 判断存在查询数据则进入 Merkle 树结构，从上往下访问子结点的 Bloom Filter 判断数据在左子树或右子树，进行一个剪枝操作，提高查找效率。

3　基于区块链的溯源系统的设计与实现

本文该部分内容结合商品的溯源流程与真实场景的应用，分析、设计、实现了基于优化后区块链底层数据结构的商品溯源查询系统。

3.1　系统设计

3.1.1 系统业务流程

在商品溯源场景中，业务主要由生产商、分销商、零售商参与，生产商与分销商将数据写入该区块链系统的对应“缓存池”中，当零售商成功将商品售卖出去时则，将零售数据与“缓存池”中对应的商品数据组合成完整的溯源数据信息，并将其上链。具体业务流程如图 8 所示：

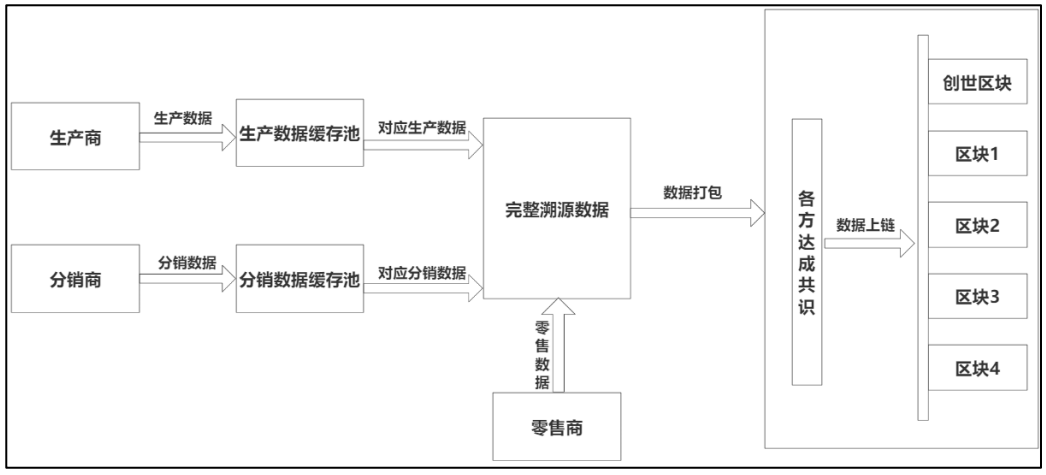

图 8 商品溯源查询系统业务流程示意图

在生产区块与构建 Merkle 树中，由于 SHA-256 算法具有单向性与独立性。本文采用了 SHA-256[15]这一哈希加密算法，以保证数据的完整性。具体加密流程如图 9 所示：

图 9 哈希加密流程图

同时在数据上链过程中，本文采用了 ECDSA 算法[16]这一非对称加密算法，给每一个环节的数据加了一个签名，主要进行生产商、分销商、零售商与普通用户的角色判定，证明消息的来源。具体流程如图 10 所示。

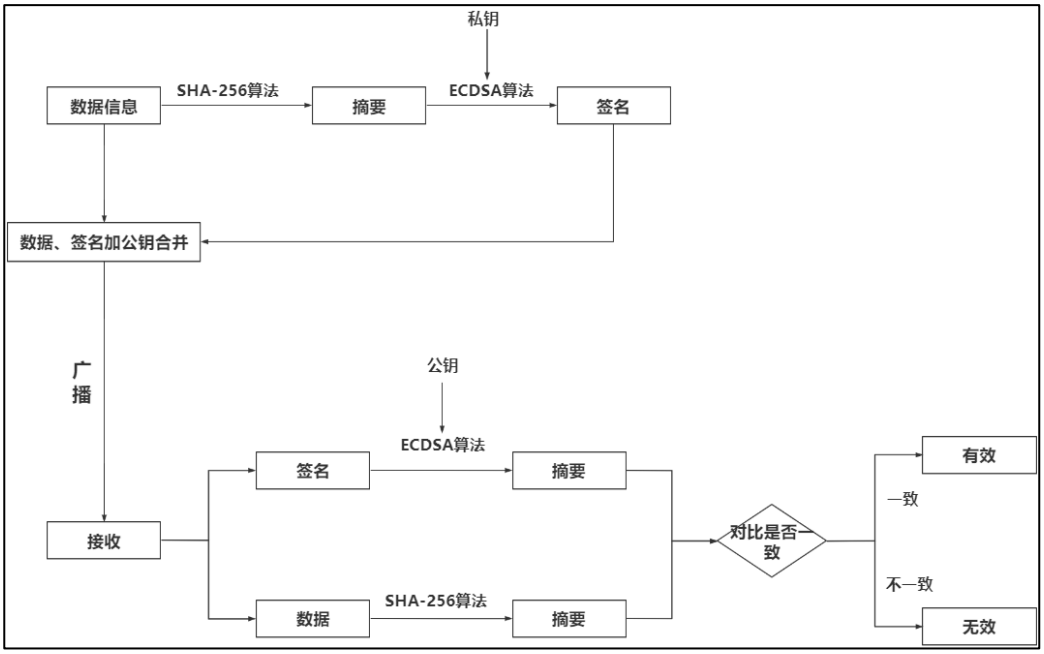

图 10 数字签名验证流程图

3.1.2 系统总体架构

本文主要探索对区块链底层数据结构优化，以改善区块链的溯源查询性能，故本溯源查询系统的总体架构[17]为图 11 所示，移除了共识层。只有生产商、分销商、零售商有增加数据权限，普通用户只有查询权限。

在本区块链系统中，取消了“挖矿”这一 POW 共识机制[18]，依靠数字签名进行用户权限的认证，新增区块的判断由系统本身设定值，当交易量到达一定值时自动验证数据并生成区块。在利用 RocksDB 将数据以物理链式结构写入本地文件，加载区块链时将 RocksDB 中的数据写入运行内存的一个动态数组中，将物理的链表结构变为抽象概念上的链式结构。

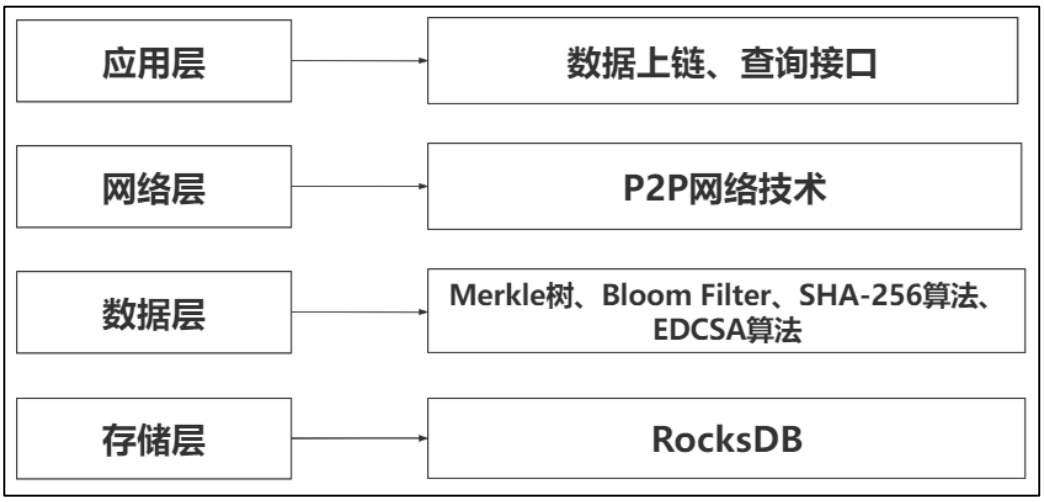

图 11 本系统架构示意图

4 实验与结果分析

本章基于上一章设计实现的商品溯源查询系统，完成对商品传统溯源查询性能与优化后的查询性能进行比对，通过实验结果验证了优化区块链底层数据结构的优化方案的可行性。

4.1 实验环境

本文主要通过优化区块链底层数据结构进行区块链的溯源查询性能优化，为排除网络延迟等各种因素干扰，本次实验只设立了一个主节点，即本机开发环境。

具体本地实验环境如表 1 所示：

表 1 实验环境配置表

类型	名称	描述
本地硬件环境	开发环境	操作系统：Win10 CPU：Intel Core 8th i5 内存 16GB
本地硬件环境	测试环境	操作系统：Win10 CPU：Intel Core 8th i5 内存 16GB

表 1 实验环境配置表（续表）

软件环境	Java	运行环境：JDK1.8
	IDEA	运行版本：2022.1.3
	PostMan	运行版本：8.12.1
	RocksDB	运行版本：7.03

4.2 实验结果

在本地测试环境下进行一定量数据的分析测试,对本溯源查询系统的各方面性能进行了定性、定量的分析，得到了预料中的结果，实验取得成功。

4.2.1 定性分析

由于 Bloom Filter 存储空间和插入及查询时间是常量的 O(k)，且算法 2 构建优化后 Merkle 树相比于传统 Merkle 树构建算法只在非叶子节点额外添加了 Bloom Filter 结构，额外花费的时间和空间都为(n-1)·O(k)，n 为该区块的信息数量，O(k)为常数。故构建优化 Merkle 树，额外花费的时间空间都是很小的。

根据上文算法 1 基于传统区块链的溯源查询算法和算法 3 基于优化 Merkle 树的查询算法进行该系统定性分析。设区块链的区块数目为 n，区块最大数据数量为 m，包含所查询数据的区块数量为 k。

算法 1 历遍所有区块的时间复杂度为 O(c1·n)，算法 3 的时间复杂度为 O(c2·n)，c1 和 c2 为常数项。因为优化后的 Merkle 有 Bloom Filter 结构，可以直接判断区块是否包含所查询数据，若无则不进入该区块的 Merkle 树结构，故 c1 大于 c2。

算法 1 在进入区块后需要查询数据集合，其时间复杂度为 O(k·m)，传统区块链中溯源查询的数据可能分布在多个不同区块中，故需要进行 k 次查询。算法 3 在进入区块后，由于 Merkle 树的结点中包含了 Bloom Filter 可进行剪枝操作，且由于溯源数据统一上链 k=1，故该过程时间复杂度为

O(logm)。

同时将区块链底层改成了数组结构，历遍全部区块的时间复杂度的从O(c1•n)变为O(c2•n)，n为区块链包含的区块数量，c1，c2为常数项，其中c1小于c2，还使得其有了随机访问任意区块能力，在常数项上优化了区块链的溯源查询性能。

算法1时间复杂度为O(c1•k•n•m)，算法3时间复杂度为O(c2•n•logm)算法3的时间复杂度小于算法1，故该优化方案在溯源查询效率上有了极大的提高。

4.2.2 定量分析

设置一定数量的区块和数据集，通过本地实验得出以下数据图表，展示了传统溯源查询性能与优化后溯源查询性能的差距。根据图12、图13、图14所示，该方案对区块链溯源查询性能有着不错的效率提升，同时额外耗费的Merkle树构建成本低。

如图12所示，为固定每区块包含16个交易数据时，查询时间与区块数量关系折线图。

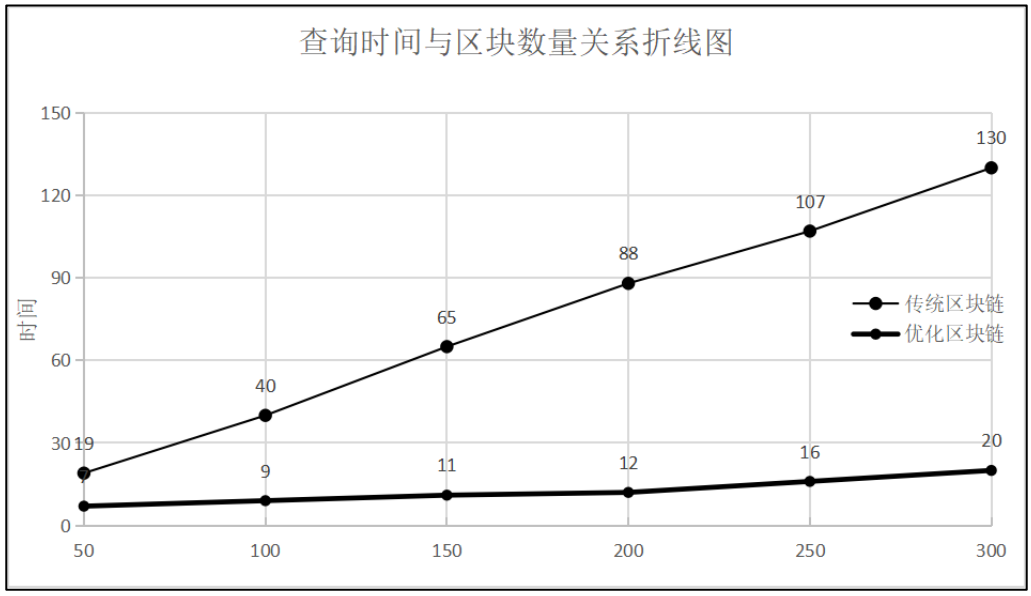

图 12 查询时间与区块数量关系折线图

如图 13 所示，该图为 Merkle 树构建成本与当前区块包含交易数量关系折线图。

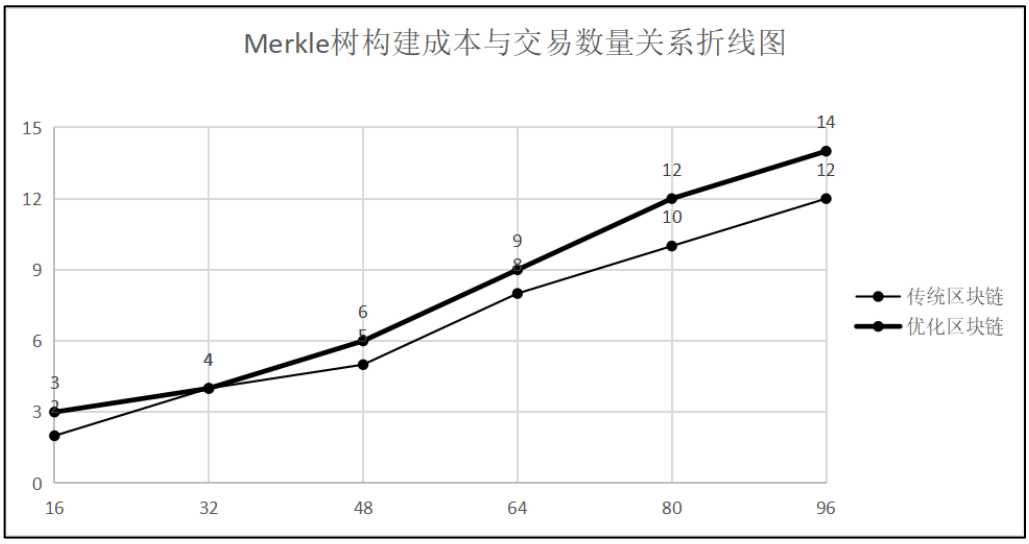

图 13 Merkle 树构建成本与交易数量关系折线图

如图 14 所示，为固定 16 个区块时，查询时间与交易数量关系折线图。

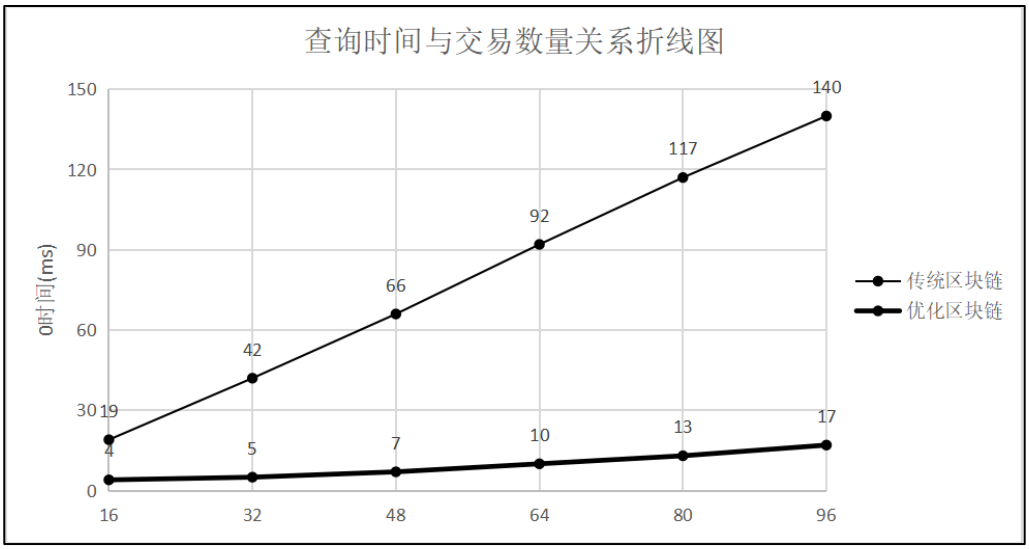

图 14 查询时间与交易数量关系折线图

5 总结与展望

在当今社会中，由于信息被篡改、不一致而导致的问题层出不穷，而利用区块链技术的不可篡改性则可以很好地解决这类问题。由于区块链本身查询效率低、支持语义查询功能贫乏等原因，区块链技术往往很难有效地

被用于各行各业。

在本优化方案中，不仅优化了区块链的溯源查询性能，还新增了 Bloom Filter 这一可以代表全集的数据结构，拓展了区块链根据语义查询功能，可以根据具体场景将需要的数据索引映射到 Bloom Filter 中。针对不同的业务场景，可以基于本优化方案设计不同的区块链平台，使区块链技术更好地在各行各业发挥其作用。