计 算 机 研 究 与 发 展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220739
61(3): 799−807, 2024

# 面向主从区块链的多级索引构建方法

王俊陆　张桂月　杜立宽　李　素　陈廷伟
（辽宁大学信息学院　沈阳　110036）
（wangjunlu@lnu.edu.cn）

# A Multi-Level Index Construction Method for Master-Slave Blockchain

Wang Junlu, Zhang Guiyue, Du Likuan, Li Su, and Chen Tingwei
（*School of Information*, *Liaoning University*, *Shenyang* 110036）

**Abstract**　Master-slave blockchain is a novel information processing technology that is domain-oriented and uses efficient cryptography principles for trustworthy communication and storage of big data. With the exponential growth of the scale of domain data, the existing master-slave blockchain system has increasingly serious problems such as low query efficiency and long traceability time. To address these issues, we propose a multi-level index construction method for master-slave blockchain (MSMLI). Firstly, MSMLI introduces a weight matrix and partitions the entire master-slave blockchain based on the master chain structure, and the weight of each partition is assigned. Secondly, for the master blockchain in each partition, a master chain index construction method based on jump consistent Hash (JHMI) is proposed, which takes the key value of the nodes and the number of index slots as input and outputs the master chain index. Finally, a Bloom filter is introduced to improve the column-based selection function and a secondary composite index on the subordinate blockchain corresponding to each master block is built. Experimental results on three constraint conditions and two types of datasets demonstrate that the proposed method reduces the index construction time by an average of 9.28%, improves the query efficiency by 12.07%, and reduces the memory overhead by 24.4%.

**Key words**　blockchain; indexing; sharding; jump consistency Hash; improved Bloom filter

**摘　要**　主从区块链是一种面向领域的、采用高效密码学原理进行大数据可信化通信及存储的新型信息处理技术.随着领域数据规模的指数级增长,现有主从区块链系统存在的查询效率低、溯源时间长等问题愈发严重.针对这些问题,提出一种面向主从区块链的多级索引构建方法（multi-level index construction method for master-slave blockchain, MSMLI）.首先,MSMLI引入权重矩阵,基于主链结构将整个主从区块链进行分片,并对各个分片进行权重赋值;其次,针对每个分片内的主区块链,提出基于跳跃一致性哈希的主链索引构建方法（master chain index construction method based on jump consistent Hash, JHMI),输入节点关键值和索引槽位数量,输出主链索引;最后,引入布隆过滤器,改进基于列的选择函数,对各个主区块对

**收稿日期**: 2022-08-22; **修回日期**: 2023-04-19
**基金项目**: 国家重点研发计划项目(2021YFF0901004);辽宁省应用基础研究计划项目(2022JH2/101300250);数字辽宁智造强省专项资金(数字经济方向)(13031307053000568);辽宁省中央引导地方科技发展资金计划项目(2022JH6/100100032);辽宁省自然科学基金项目(2022-KF-13-06)
This work was supported by the National Key Research and Development Program of China(2021YFF0901004), the Applied Basic Research Program of Liaoning Province (2022JH2/101300250), the Digital Liaoning Intelligent Manufacturing Strong Province Funds for Direction of Digital Economy(13031307053000568), the Central Government Guides Local Science and Technology Development Foundation Project of Liaoning Province (2022JH6/100100032), and the Natural Science Foundation of Liaoning Province (2022-KF-13-06).
**通信作者**: 陈廷伟(twchen@lnu.edu.cn)

应的从属区块链构建2级复合索引. 在3种约束条件和2类数据集上的实验结果表明, MSMLI对比现有方法, 平均能够缩减9.28%的索引构建时间, 提升12.07%的查询效率, 同时降低24.4%的内存开销.

**关键词** 区块链; 索引; 分片; 跳跃一致性哈希; 改进布隆过滤器

**中图法分类号** TP311

区块链通过块链式数据结构存储并验证数据, 辅以密码学[1-2]方法保证数据传输和访问的安全性, 具有高可信[3]、可回溯、去中心化[4]等特点, 能够很好地解决数据存储对第三方的信任问题[5]. 随着区块链技术的发展以及各行业数据规模的累计, 传统的单链结构区块链系统已经无法满足愈发复杂的领域应用场景, 主从区块链(master-slave blockchain, MSBC)结构, 如星火链、COSMOS等, 开始受到领域专家学者的关注, 并逐步在教育、医疗、安全[6]等领域广泛应用[7-10]. 主从区块链通常包括主链和从属链2部分, 分别由主区块和从属区块组成, 每个主区块有且只有一个从属链. 各个主区块和从属区块之间分别通过前一个主区块和从属区块的哈希值相连, 主区块与从属链通过唯一的哈希值进行映射.

主从区块链结构可以应对复杂分类场景的应用. 如金融领域中, 采用主从区块链构建面向金融活动的企业区块链系统, 主链中存储金融企业属性信息, 对应的从属链存储其交易事件、金融活动等数据, 通过区块链的共识机制[11-12]保证数据不可篡改.

随着领域数据规模的不断增大, 主从区块链系统存在的查询效率低、溯源时间长等问题[13-14]愈发严重. 而现有区块链索引方法多只适用于单一链结构, 且查询效率和索引构建时间均较差. 因此, 如何针对主从区块链建立高效、可动态维护的索引结构, 成为领域研究的热点和难点.

针对上述问题, 本文提出一种面向主从区块链的多级索引构建方法(multi-level index construction method for master-slave blockchain, MSMLI). 主要贡献有4个方面:

1)综合节点负载、节点信用和网络质量构建权重矩阵, 提出一种基于权重矩阵的区块链分片算法(weighted matrix-based blockchain sharding algorithm, WMBS), 基于主链特征实现主从区块链结构的动态分片;

2)在此基础上, 针对现有区块链索引不适用于主从链结构的问题, 提出基于跳跃一致性哈希的主链索引构建算法(master chain index construction algorithm based on jump consistent Hash, JHMI), 通过主链的节点关键值与索引槽位映射, 实现主链信息的高效查询;

3)结合区块链的数据特点, 提出基于改进布隆过滤器的从属链索引构建算法(construction algorithm for subordinate chain indexes based on improved Bloom filters, IBF), 优化基于列的选择函数, 并给出索引查询方法;

4)在不同约束条件和数据集上与现有方法进行对比实验, 验证本文所提方法的有效性.

## 1 相关工作

目前, 许多学者对于区块链的索引构建问题进行了深入研究, 取得了一定研究成果.

文献[15]提出一种多维内存读取优化的索引方法Flood, 通过联合优化索引结构和数据存储布局, 自动适应特定的数据集和工作负载. 但是该方法无法检测到查询分布何时发生了足够大的变化, 需要定期评估当前布局的查询成本对不同的工作负载完全重建索引. 文献[16]提出一种基于B级树的索引方法EBTree, 支持对以太坊区块链数据的实时top-$k$、范围、等价搜索, 但是该方法的索引节点均在Level DB中单独存储, 且查询效率受节点大小影响较大. 文献[17]提出一种单通道学习索引RS(radix spline)方法, 可以在排序数据的单次传递中构建, 且只需要2个数据集, 对大多数数据集友好, 但是该方法会随着数据集的增长降低性能. 文献[18]提出一种基于子链账户交易链的索引方法SCATC, 将交易链划分为子链并在每个子链的最后一个区块的账户分支节点上添加哈希指针连接每个子链, 通过指针将遍历交易链的查询模式转化为子链查询来减少计算开销, 但是该方法仅对较长账户交易链的查询效率有所改善且只针对明文状态下的查询优化, 无法保证数据隐私性. 文献[19]提出一种用于大时间序列数据的可扩展分布式索引方法ChainLink, 设计了一个2层分布式索引结构, 使用单通道签名对数据进行散列, 利用分区级数据重组来实现查询操作, 但是该方法需要在本地对数据重组, 无法保障数据安全性. 文献

[20] 提出一种基于中间层的可扩展学习索引模型 Dabble, 使用 *K*-means 聚类算法, 根据数据分布将数据集划分为 *K* 个区域并分别使用神经网络学习和训练, 通过神经网络模型预测数据位置, 但是该方法在数据集更新时需要重新训练模型, 时效性较差, 并且 *K* 值对模型的准确性影响较大.

## 2 基于主从结构的区块链分片方法

为实现主从区块链结构的高效索引构建和查询处理, 基于主链特征对整个主从区块链分片, 并对各个分片赋予权重, 据此构建整个主从区块链结构的分片权重矩阵; 基于权重矩阵确定分片中的节点数目, 为主链和从属区块链构建索引提供支撑.

### 2.1 权重矩阵构建

设主从区块链的节点数目为 $x$, 主从区块链分为 $y$ ($y \ll x$)个分片, 第 $i$ 个分片为 $f_i(i=0,1,\cdots,y-1)$, 各个分片权重为 $\omega_i$. 分片权重由节点负载、节点信用、网络质量 3 个维度的权重决定, 其中节点信用和网络质量与分片权重呈正相关, 节点负载与分片权重呈负相关, 各项维度的权重由实验效果最佳的比例决定. 在进行分片权重计算前, 由于上述 3 个维度的单位不统一, 因此需进行归一化处理. 节点负载的归一化公式为

$$d_{ij}=\left\lfloor \frac{1}{\log(d'_{ij}+1)+1}\times\frac{x}{y}\right\rfloor, \tag{1}$$

节点信用和网络质量的归一化公式为

$$d_{ij}=\left\lfloor \frac{d'_{ij}+1-\min}{\max+1-\min}\times\frac{x}{y}\right\rfloor, \tag{2}$$

第 $i$ 个分片的权重为 $\omega_i$, 计算公式为

$$\omega_i=\sum_{j=1}^{3}\omega_{ij}d_{ij}, \tag{3}$$

式(1)(2)中的 $d'_{ij}$ 表示节点负载、节点信用和网络质量的原始数值, $d_{ij}$ 为归一化后的数值, max 和 min 分别为该项分量最大数值和最小数值. 式(3)中的 $\omega_{ij}$ 表示各项分量的权重(权重比例选择见 4.1 节).

设 2 维权重矩阵 $\boldsymbol{M}$ 的各个元素由分片权重组成, 获得各个分片的权重后, 使用各个分片权重 $\omega_i$ 构建 2 维权重矩阵 $\boldsymbol{M}_{p\times q}$(其中 $p\leqslant\sqrt{y}$, $q\leqslant\sqrt{y}+1$, $p$, $q$ 为整数). 对于任意一个分片 $f$, 都有 $M[f/p][f\%q]=\omega_f$, 矩阵中为空的元素置为 0.

### 2.2 基于权重矩阵确定片内节点数目

基于 2.1 节的权重矩阵, 确定各个分片内的节点数目. 首先, 对矩阵中的分片权重进行线性归一化处理. 其次, 对归一化后的所有分片权重按照比例离散, 设离散比例权重值区间为 [1,*Q*]. 最终获得分片比例权重 $Q-1$, 即该分片将对应 $Q-1$ 个节点.

将节点进行编号, 序号为 0, 1, …, $\sum_{i=0}^{x-1}\omega_i-1$, 则第 $k$ 个分片对应的节点序号为 $\left[\sum_{i=0}^{k}\omega_i,\ \sum_{i=0}^{k+1}\omega_i-1\right]$, 在获取节点编号后, 通过查表法得到分片的编号. 假定 $\omega_i$ 在进行归一化和离散化处理后得到的分片比例权重为 7, 则该分片将对应 7 个节点, 并将这 7 个节点进行编号, 编号顺序为 0, 1, …, 6.

线性归一化后的分片权重的公式为

$$\omega'_i=\frac{\omega_i-\omega_{\min}}{\omega_{\max}-\omega_{\min}}, \tag{4}$$

其中的 $\omega_{\min}$ 为所有分片权重 $\omega_i$ 中的最小值, $\omega_{\max}$ 为最大值.

基于 2.1 节的权重矩阵的构建过程, 提出了一种基于权重矩阵的区块链分片算法 WMBS, 其通过跳跃搜索(jump search, JPS)算法快速进行区块链分片与节点的映射, 实现了主从区块链结构的分片. WMBS 将节点关键值 *key*、随机数 *r* 和权重矩阵作为输入, 基于 JPS 使节点与分片逐一映射.*key* 在节点加入区块链时创立, 为 32 b 节点关键值, 其由 8 b 的分片地址码和 24 b 的节点随机码组成, 是节点的唯一标识; *r* 是一个在 [0,1] 区间内均匀分布的随机数, 由线性同余随机数发生器生成. WMBS 算法如算法 1 所示.

**算法 1.** WMBS 算法.

输入: 节点关键值 *key*, 随机数 *r* 和权重矩阵 $\boldsymbol{M}_{p\times q}$;

输出: 节点行列号 *row*, *colmun*.

① *WMBS*(*key*, *r*, $\boldsymbol{M}_{p\times q}$)

② int *i*, *j*;

③ for *i* = 0 to *p*−1

④ for *j* = 0 to *q*−1

⑤ *num_row* = *sum*(*M*[*i*][*j*]); /* 统计权重矩阵中每一列的行数 */

⑥ end for

⑦ *row* = *JPS*(*key*, *r*, *num_row*); /* 在每行均调用 JPS 算法确定节点所在行 */

⑧ *row* = *row*×*q*; /* 获得节点行号 */

⑨ end for

⑩ for *i* = 0 to *q*−1

⑪ *num_col* = *M*[*i*].*length*; /* 统计该行列数 */

⑫ *colmun* = *JPS*(*key*, *r*, *num_col*); /* 调用 JPS

算法确定节点所在列 */

⑬ end for

为优化分片内搜索节点的路径, 本文提出算法 JPS, 降低线性搜索算法的时间复杂度. 如算法 2 所示.

**算法 2.** JPS 算法.

输入: 节点关键值 *key*, 随机数 *r* 和权重矩阵的行数或列数 *num*;

输出: 节点行号或列号 *b*.

① *JPS*(*key*, *r*, *num*)

② int $b=0, j=0$;

③ *random.seed*(*key*, *r*); /*以 *key* 和 *r* 为种子生成随机数*/

④ while($j<num$)

⑤ $b=j$;

⑥ $j=floor((b+1)/random())$;

⑦ end while

⑧ return *find_in_table*(*b*). /* 通过查表法获得分片编号 */

## 3 多级索引构建方法

由于主从区块链结构的主链和从属链存储的数据规模和信息类型均不同, 通过 WMBS 算法分片后, 本文提出一种面向主从区块链的多级索引构建方法(MSMLI), 以满足主链和从属链的查询需求. 多级索引构建示意图如图 1 所示.

如图 1 所示, 在主链中, 通过 JHMI 建立 1 级索引. 在从属链中, 引入 IBF 建立 2 级索引, 2 级索引通过节点关键值进行关联. 当查询发生时, 首先在主链索引中检索节点关键值, 得到查询结果后, 基于该结果在从属链索引中通过节点关键值获得元素信息.

### 3.1 基于跳跃一致性哈希的主链索引构建

基于主链存储的数据特点, 本文引入跳跃一致性哈希算法, 提出 JHMI, 实现主链索引的快速构建. 首先, 根据各主链分片上的节点数量确定索引的槽位数量; 其次, 根据主链存储数据的哈希值确定各个节点关键值; 最后, 输入节点关键值和索引槽位数量, 输出主链索引.

当分片中节点数量发生变化时, 节点在索引中发生跳跃变化, 部分节点重新映射. 设产生跳跃变化的哈希映射函数为 *ch*(*key*, *num_buckets*), *key* 为节点关键值, *num_buckets* 为槽位数量, 可得: 1) *num_buckets* = 1 时, 只有 1 个槽位, 所有 *key* 都映射到 1 个槽位中, 即 *ch*(*key*, *num_buckets*) = 0, 所有节点都划分在 0 号槽位; 2) *num_buckets* = 2 时, 有 1/2 的节点在 *ch*(*key*, *num_buckets*) = 0, 有 *K* / 2 个 *key* 重新映射即 *ch*(*key*, *num_buckets*) = 1, 跳跃到 1 号槽位; 3) 当槽位数从 *n* 变为 *n*+1 时, 有 $n/(n+1)$ 的节点所在的槽位保持不变, 即 *ch*(*key*, *num_buckets*) = *n*−1, 有 $1/(n+1)$ 个 *key* 需要重新映射, 即 *ch*(*key*, *num_buckets*) = *n*.

设 *b* 为节点上一次跳变的结果, *j* 为发生跳变前最后一次槽位扩充时槽位的数量, 当分片内节点数量发生变化时, 节点重新映射的示意图如图 2 所示.

由图 2 可知, 对于任意的 $i\in[b+1, j-1]$, 增加节点数量且没有发生跳变的概率为

$$P(i)=\frac{b+1}{b+2}\times\frac{b+2}{b+3}\times\cdots\times\frac{i-1}{i}=\frac{b+1}{i}. \tag{5}$$

在 [0,1] 区间取一个均匀分布的随机数 *r*, 由式(5)可得, 当 $r<(b+1)/r$ 时, 节点就会跳变成 *j*, 则 *i* 的上界

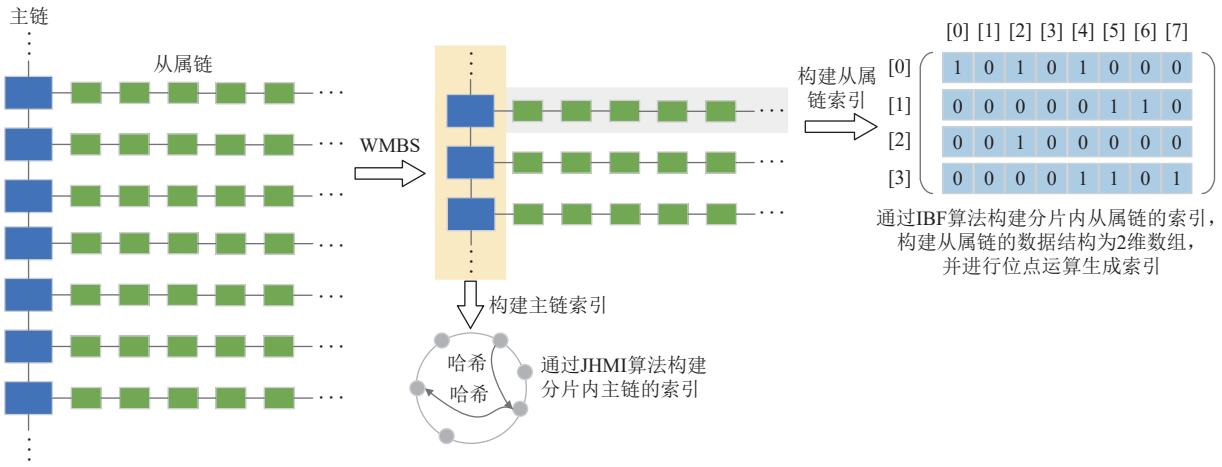


Fig. 1 Multi-level index construction schematic

图 1 多级索引构建示意图

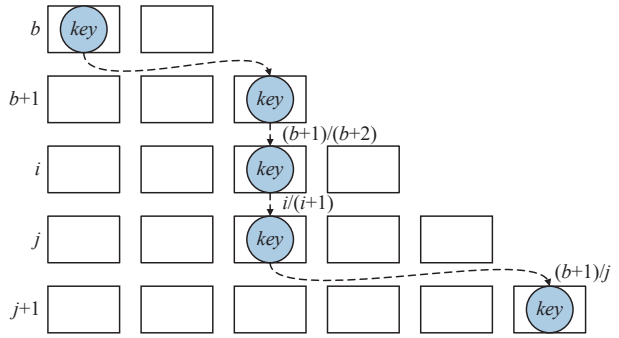


Fig. 2 Diagram of node remapping

图 2 节点重新映射图

为$(b+1)/r$. 由于对任意的$i$都有$j \geqslant i$, 则$j = floor((b+1)/r)$. JHMI 算法如算法 3 所示.

**算法 3.** JHMI 算法.

输入: 节点关键值 *key*, 槽位数量 *num_buckets* 和空主链索引 *s*;

输出: 主链索引 *s*.

① *JHMI*( *key*, *num_buckets*, *s*)

② int $b = 1, j = 0$;

③ while($j <$ *num_buckets*)

④ $b = j$;

⑤ $key = key \times 2862933555777941757ULL + 1$;

/* 根据存储数据的哈希值确定各个节点关键值 */

⑥ $j = (floorb/key)$;

⑦ $s[] = b$;

⑧ end while

⑨ return *s*.

### 3.2 基于改进布隆过滤器的从属链索引构建

从属链存储的数据具有规模大、多源异构的特点, 因此, 在从属链索引构建过程中, 本文重构布隆过滤器数据结构, 提出 IBF 算法.

首先构建基于改进布隆过滤器的从属链索引的数据结构为 2 维数组 $A[p][q]$, 其中 $p = 2^n$($n$ 为正整数), $q$ 中的数据长度为 $l_q$, 设 $l_q = 32/64$ b, 其取值由 CPU 中通用寄存器的缓存行长度决定, 以减少内存访问, 提高查询性能. 设改进布隆过滤器的 $K$ 个哈希函数为 $Hash(key)$, 其中, *key* 为节点关键值, 设一个改进布隆过滤器中可以存储的元素长度为 *len*, 则 *len* 计算结果为

$$len = p \times q = 2^n \times l_q. \tag{6}$$

在完成索引数据结构构建后, 将所有位点初始化为 0, 对每个主区块对应的从属链构建索引, 具体有 3 个步骤:

1)使用选择列的函数, 先将元素映射到对应列, 该元素将在对应列的位点;

2)通过 $K$ 个哈希运算函数获得位点;

3)将对应位点置为 1.

其中, 步骤 1 中的列选择函数产生了额外的计算开销, 为降低计算开销, 将优化哈希函数, 从属链上存储的交易哈希值是交易经过 SHA256 哈希函数运算得到的, 因此优化步骤 1 的选择列函数为以 SHA256 哈希函数为基础, 通过取模运算得到选择列的函数. 步骤 1 中的选择列函数可表示为

$$q_v = v\%2^n. \tag{7}$$

步骤 2 中的 $K$ 个哈希函数由 $K$ 个按位与运算组成, 可表示为

$$p_v{}^i = v\& \sum_{i=i-1}^{i\times\frac{k'}{k}-1} 2^{i-1}(0 < i \leqslant k), \tag{8}$$

式(7)中的 $v$ 为布隆过滤器中的元素, $q_v$ 为进行列选择后获得的列号. 在获得元素所在列后, 确定该元素在构建和查询时都将被限制在对应列. 式(8)中 $p_v{}^i$ 为在获得列号后在该列中通过 $K$ 次式(8)的运算获得的行号, 即对应位点, $k'$ 为在通用布隆过滤器中的数组长度. 基于改进布隆过滤器的从属链索引构建算法如算法 4 所示.

**算法 4.** IBF_Construction 算法.

输入: 从属链元素集合 $V'$, 元素 $v$, 改进布隆过滤器数组长度 $k$ 和通用布隆过滤器数组长度 $k'$;

输出: 元素位点 $p$ 和选择列 $q$.

① *IBF_Construction*( $V'$, $v$, $k$, $k'$);

② int $p = 0$, $q = 0$;

③ for $v$ in $V'$

④ $q = v \% 2^n$; /* 选择元素所在列 */

⑤ $p = v \& \sum_{i=i-1}^{i\times\frac{k'}{k}-1} 2^{i-1}$; /* 获得元素位点 */

⑥ end for

在构建完从属链的索引后, 提出一种根据从属链上节点关键值和选择列函数的基于改进布隆过滤器的从属链索引查询算法, 如算法 5 所示.

**算法 5.** IBF_Query 算法.

输入: 查询元素 $v$, 改进布隆过滤器中的元素 $V$;

输出: 判断结果.

① *IBF_Query* ( $v$, $V$);

② int $p = 0$, $q = 0$;

③ for $v$ in *IBF*

④ $q = v \% 2^n$; /* 选择元素所在列 */

⑤ $p = v \,\&\, \sum_{i=i-1}^{i\times\frac{k'}{k}-1} 2^{i-1}$; /* 获得元素位点 */

⑥ if($A[p][q] == 1$)

⑦ output yes;

⑧ end if

⑨ end for

## 4 实 验

本文实验环境为 16 台 4 TB 存储空间，128 GB RAM，16 核 24 线程 i9-12900KS CPU 的服务器集群，服务器之间通过高速局域网通信，每台服务器均部署 Ubuntu 18.04 操作系统. 实验采用 2 个不同的数据集进行实验验证：数据集 1 为公共以太坊网络中的前 3 000 000 个区块，数据集 1 中有 15 362 853 笔交易；数据集 2 为 Lognormal 人工数据集，Lognormal 数据集按照对数正态分布，以均值为 0、方差为 2 的方式采样了 500 万条不重复的数据. 本节将从索引构建时间、查询时间、内存消耗 3 个方面验证 MSMLI 的高效性和低内存的优点.

### 4.1 分片权重比例选择

在分片准备阶段将通过服务器构建 10 个分片，1 个服务器构建 1 个节点并将节点分配到对应分片. 分片中的节点容量设置为 500 个节点. 对比节点数目设为 100, 200, 300, 400 时，分片权重的 3 个维度节点负载、节点信用、网络质量的比例分别设置为 3∶3∶4, 4∶3∶3, 5∶2∶3 三种情况. 实验结果如图 3 所示.

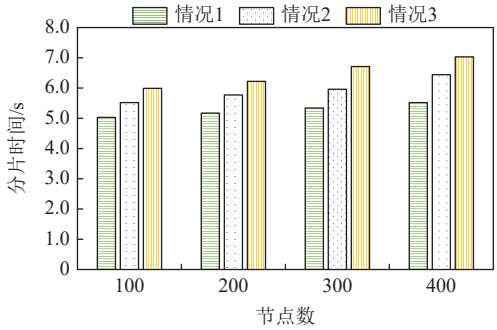


Fig. 3 Shard weight ratio

图 3 分片权重比例

由图 3 可知，在节点数量相同时，情况 1 的实验效果最好，且随着节点负载维度的比例增大时，分片时间增加. 随着节点数量的增多，情况 2 和情况 3 的时间明显增加，而情况 1 的时间增幅不大，维持在 5s 左右. 因此，本文将按照主区块链对区块链分片时节点负载、节点信用、网络质量的分片权重为 3∶3∶4 的比例进行.

### 4.2 索引构建时间对比

为在索引构建时间方面验证本文提出的 MSMLI 方法的高效性，将分别对比改进区块链结构的 EBTree 方法和引入神经网络的 Dabble 模型方法. EBTree 方法中的内部节点的能力设置为 128，叶节点的能力设置为 16. Dabble 模型中的 $K$=100，MSMLI 方法的 1 个分片中的节点数设置为 100. 在本节实验中，将分为 3 种具体情况讨论.

如表 1 所示，使用 2 个不同规模的数据集将 MSMLI 方法与 EBTree, Dabble 这 2 种方法进行索引构建时间对比. 索引构建时间对比实验结果如图 4、图 5 所示.

由图 4、图 5 可知，随着数据量的增大，MSMLI 方法与现有方法对比，在索引构建时间方面优化了约 9.28%，其中 Dabble 方法训练神经网络模型需要约 15 s，因此，MSMLI 方法大大优于 Dabble 方法.

### 4.3 查询时间对比

在本节实验中，首先将索引和数据加载进内存，为测试 MSMLI 方法的查询性能，将使用不同规模数据集和查询条件对比查询响应时间.

#### 4.3.1 大规模数据集查询响应时间对比

使用大规模数据集，数据集 1 为公共以太坊网络中的前 3 000 000 个区块. 对比 EBTree 方法和 Dabble 方法在主区块数目为 50 万、100 万、150 万、200 万、250 万、300 万，从属区块数目为 1 000 时的查询响应时间，实验结果如图 6 所示.

由图 6 可知，在大规模数据集上 MSMLI 方法与现有方法对比，在索引构建时间方面优化了约

**Table 1 Index Construction Time Comparison**

**表 1 索引构建时间对比**

| 情况 | 数据集 1 | | | 数据集 2 | | |
|---|---|---|---|---|---|---|
| | MSMLI | EBTree | Dabble | MSMLI | EBTree | Dabble |
| 情况 1 | 从属区块数据为空，主区块数目分别为 500，1 000，1 500，2 000 | | | 从属区块数据为空，主区块存储 500，1 000，1 500，2 000 条数据 | | |
| 情况 2 | 主区块数据为空，从属区块数目为 50 万、100 万、150 万、200 万、250 万、300 万 | | | 主区块数据为空，从属区块存储 2 万、4 万、6 万、8 万、10 万条数据 | | |
| 情况 3 | 主从区块数据均非空，主区块数目分别为 500，1 000，1 500，2 000，从属区块数目为 50 万 | | | 主从区块数据均非空，主区块存储 500，1 000，1 500，2 000 条数据，从属区块存储 2 万条数据 | | |

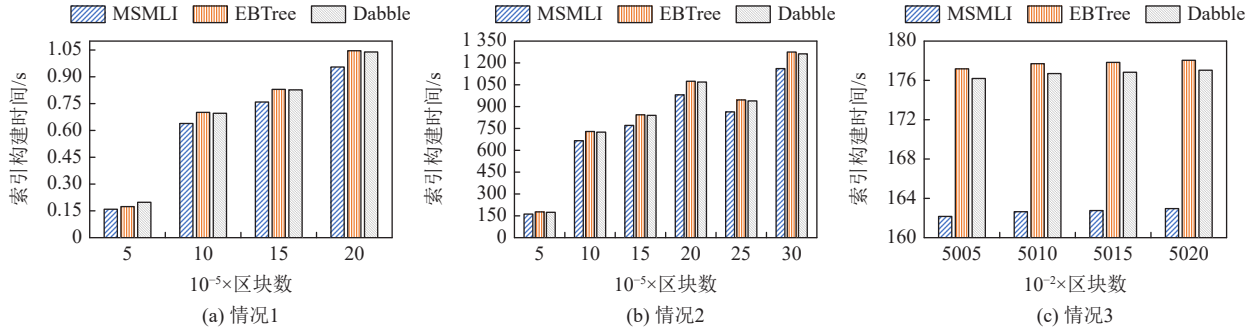


Fig. 4 Index construction time comparison on dataset 1

图 4 数据集 1 上索引构建时间对比

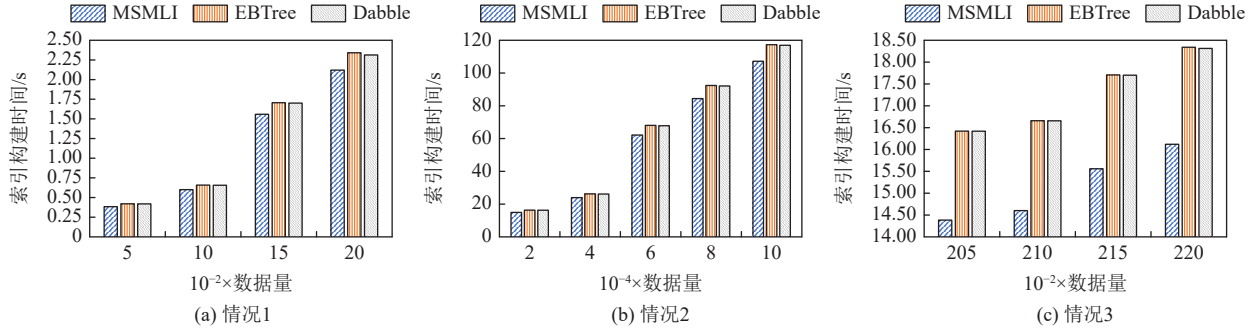


Fig. 5 Index construction time comparison on dataset 2

图 5 数据集 2 上索引构建时间对比

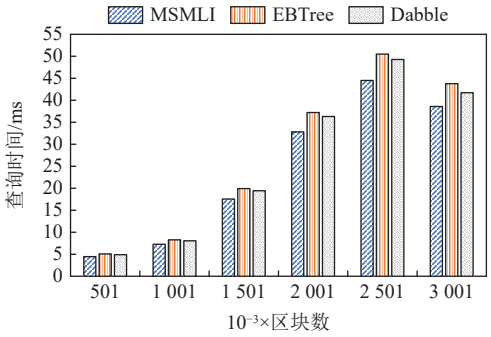


Fig. 6 Index query time comparison of large-scale dataset

图 6 大规模数据集索引查询时间对比

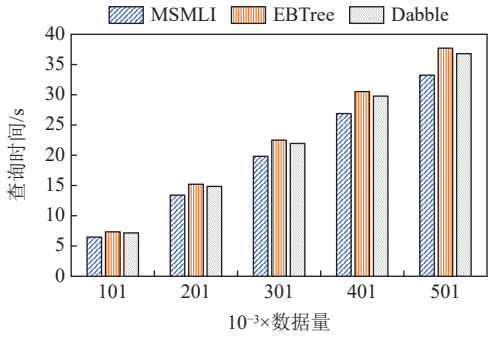


Fig. 7 Index query time comparison of small-scale dataset

图 7 小规模数据集索引查询时间对比

13.44%, 区块数量增大时 EBTree 方法优势更明显.

#### 4.3.2 小规模数据集查询响应时间对比

使用小规模数据集, 数据集 2 为 Lognormal 人工数据集, 其中具有 500 万条数据, 总大小约 24 MB. 对比 EBTree 方法和 Dabble 方法在主区块存储 10 万、20 万、30 万、40 万、50 万条数据, 以及从属区块存储 1 000 条数据时的查询响应时间, 实验结果如图 7 所示.

由图 7 可知, 在小规模数据集上 MSMLI 方法与现有方法对比, 在查询时间方面优化了约 10.71%. 实验结果表明 MSMLI 方法在数据量大时查询性能更好.

### 4.4 内存消耗对比

MSMLI 方法在分片阶段构建的权重矩阵几乎不占用内存, 主链基于跳跃一致性哈希算法构建索引, 跳跃一致性哈希对比经典的一致性哈希几乎没有额外内存消耗. 因此 MSMLI 方法中的内存开销主要考虑从属区块链的索引构建.IBF 的假阳性设置为 0.013 7, $l_q = 64$ b. EBTree 方法改写区块链结构, 内存消耗主要为区块数据. MSMLI 方法与 Dabble 方法的对比实验结果如表 2 所示.

由表 2 可知, Lognormal 数据集占用内存 24 MB; Dabble 方法占用内存 4 KB; 对于 MSMLI 方法, IBF 在保证假阳性可允许范围内仍仅占用内存约 2.048 KB, 即使将 IBF 与 BF 在相同假阳性要求下仍能保持相同数量级.

**Table 2 Memory Consumption Comparison**

**表 2 内存消耗对比**

| 索引方法 | Lognormal 占用内存/MB | 方法占用内存/KB |
|---|---|---|
| Dabble | 24 | 4 |
| MSMLI | 24 | 2.048 |

## 5 结 论

随着区块链技术的广泛应用,传统单链结构已逐渐无法满足日益增长的领域数据存储需求,主从区块链开始广泛应用于金融、教育、安全等领域.针对现有主从区块链系统查询效率低、溯源时间长的问题,本文提出一种面向主从区块链的多级索引构建方法(MSMLI).该方法首先将整个主从区块链结构按照主链将结构进行分片,并采用权重矩阵提高分片可维护性,为索引构建提供支持;在此基础上,针对主链和从属链数据规模不同的特征,提出基于跳跃一致性哈希的索引构建方法(JHMI),以及基于改进布隆过滤器的从属链索引构建方法(IBF),提高主从区块链查询效率.实验结果表明,本文提出的MSMLI方法对比现有方法,在构建时间、查询效率、内存占用方面具有很大优势,为主从区块链的快速检索提供了一条有效途径.

**作者贡献声明**:王俊陆提出了算法思路和实验方案;张桂月负责完成实验并撰写论文;杜立宽参与了审稿专家提出的关于查询实验部分的修改工作并补充了重要的参考文献;李素参与了审稿专家提出的关于文章具体内容上的修改工作,补充了文章框架和内容;陈廷伟提出指导意见并修改论文.

## 参 考 文 献

[1] Barbosa M, Barthe G, Bhargavan K, et al. SoK: Computer-aided cryptography [C] //Proc of the 42nd IEEE Symp on Security and Privacy (S&P). Piscataway, NJ: IEEE, 2021: 777-795

[2] Zhu Jianming, Zhang Qinnan, Gao Sheng, et al. Blockchain-based trusted federated learning model for privacy protection[J]. Chinese Journal of Computers, 2021, 44(12): 2464−2484(in Chinese)
(朱建明, 张沁楠, 高胜, 等. 基于区块链的隐私保护可信联邦学习模型 [J]. 计算机学报, 2021, 44(12): 2464−2484)

[3] Moudoud H, Cherkaoui S, Khoukhi L. Towards a scalable and trustworthy blockchain: IoT use case [C/OL] //Proc of IEEE Int Conf on Communications (ICC 2021). Piscataway, NJ: IEEE, 2021[2022-02-16].https://ieeexplore.ieee.org/document/9500535

[4] Li Chenxing, Li Peilun, Zhou Dong, et al. A decentralized blockchain with high throughput and fast confirmation [C] //Proc of USENIX Annual Technical Conf (USENIX ATC 2020). Berkeley, CA: USENIX Association, 2020: 515−528

[5] Połap D, Srivastava G, Jolfaei A, et al. Blockchain technology and neural networks for the Internet of medical things [C] //Proc of the 39th IEEE Conf on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway, NJ: IEEE, 2020: 508−513

[6] Wei Songjie, Li Shasha, Wang Jiahe. Cross-domain authentication protocols based on identity cryptosystems and blockchains[J]. Chinese Journal of Computers, 2021, 44(5): 908−920(in Chinese)
(魏松杰, 李莎莎, 王佳贺. 基于身份密码系统和区块链的跨域认证协议 [J]. 计算机学报, 2021, 44(5): 908−920)

[7] Bao Jiabin, He Debiao, Luo Min, et al. A survey of blockchain applications in the energy sector[J]. IEEE Systems Journal, 2020, 15(3): 3370−3381

[8] Bhaskar P, Tiwari C K, Joshi A. Blockchain in education management: Present and future applications [J/OL]. Interactive Technology and Smart Education, 2020[2022-03-21]. https://www.emerald.com/insight/content/doi/10.1108/ITSE-07-2020-0102/full/html

[9] Kalodner H, Möser M, Lee K, et al. BlockSci: Design and applications of a blockchain analysis platform [C] //Proc of the 29th USENIX Security Symp (USENIX Security 2020). Berkeley, CA: USENIX Association, 2020: 2721-2738

[10] Alladi T, Chamola V, Sahu N, et al. Applications of blockchain in unmanned aerial vehicles: A review [J/OL]. Vehicular Communications, 2020[2021-12-16].https://www.sciencedirect.com/science/article/abs/pii/S2214209620300206

[11] Huang Junqin, Kong Linghe, Chen Guihai, et al. Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism[J]. IEEE Transactions on Industrial Informatics, 2019, 15(6): 3680−3689

[12] Tsang Y, Choy K, Wu C, et al. Blockchain-driven IoT for food traceability with an integrated consensus mechanism[J]. IEEE Access, 2019, 7: 129000−129017

[13] Sui Yuan, Wang Wei, Deng Xue. High throughput verifiable query method for blockchain-oriented off-chain database[J]. Journal of Chinese Computer Systems, 2021, 42(6): 1304−1312(in Chinese)
(隋源, 汪卫, 邓雪. 一种面向区块链的链下数据库高吞吐量可验证查询方法 [J]. 小型微型计算机系统, 2021, 42(6): 1304−1312)

[14] Cai Lei, Zhu Yanchao, Guo Qingxing, et al. Efficient materialized view maintenance and trusted query for blockchain[J]. Journal of Software, 2020, 31(3): 680−694(in Chinese)
(蔡磊, 朱燕超, 郭庆兴, 等. 面向区块链的高效物化视图维护和可信查询 [J]. 软件学报, 2020, 31(3): 680−694)

[15] Nathan V, Ding J, Alizadeh M, et al. Learning multi-dimensional indexes [C]//Proc of ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2020: 985−1000

[16] Huang Xiaoju, Gong Xueqing, Huang ZhiGang, et al. EBTree: A B-plus tree based index for ethereum blockchain data [C] //Proc of Asia Service Sciences and Software Engineering Conf (ASSE 2020). New York: ACM, 2020: 83−90

[17] Kipf A, Marcus R, van Renen A, et al. RadixSpline: A single-pass learned index [C/OL] //Proc of the 3rd Int Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM 2020). New York: ACM, 2020[2022-02-15]. https://dl.acm.org/doi/10.1145/3401071.3401659

[18] Xing Xiaogang, Chen Yuling, Li Tao, et al. A blockchain index structure based on subchain query[J]. Journal of Cloud Computing, 2021, 10(1): 1−11

[19] Alghamdi N, Zhang Liang, Zhang Huayi, et al. ChainLink: Indexing big time series data for long subsequence matching [C] //Proc of the 36th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2020: 529−540

[20] Gao Yuanning, Ye Jinbiao, Yang Nianzu, et al. Middle layer based scalable learned index scheme[J]. Journal of Software, 2020, 31(3): 620−633(in Chinese)
(高远宁, 叶金标, 杨念祖, 等. 基于中间层的可扩展学习索引技术[J]. 软件学报, 2020, 31(3): 620−633)

**Wang Junlu**, born in 1988. PhD, lecturer. Member of CCF. His main research interests include blockchain technology, big data processing technology, and streaming data processing technology.

**王俊陆**, 1988 年生. 博士, 讲师. CCF 会员. 主要研究方向为区块链技术、大数据处理技术、流数据处理技术.

**Zhang Guiyue**, born in 1996. Master. Her main research interests include blockchain technology, big data processing technology, and machine learning.

**张桂月**, 1996 年生. 硕士研究生. 主要研究方向为区块链技术、大数据处理技术、机器学习.

**Du Likuan**, born in 1999. Master candidate. His main research interests include blockchain technology, big data processing technology, and machine learning.

**杜立宽**, 1999 年生. 硕士研究生. 主要研究方向为区块链技术、大数据处理技术、机器学习.

**Li Su**, born in 1997. PhD candidate. Member of CCF. Her main research interests include blockchain technology, big data processing technology, and streaming data processing technology.

**李　素**, 1997 年生. 博士研究生. CCF 会员. 主要研究方向为区块链技术、大数据处理技术、流数据处理技术.

**Chen Tingwei**, born in 1974. PhD, professor, master supervisor. Member of CCF. His main research interests include intelligent transportation and machine learning.

**陈廷伟**, 1974 年生. 博士, 教授, 硕士生导师. CCF 会员. 主要研究方向为智能交通、机器学习.