传感器 MDPI

文章

基于地理的区块链区域交易快速查询方法

常州[1] 慧梅陆1、勇翔2、*、吴景邦[3]和冯王[4]

[1]北京理工学院计算机科学技术学院，北京100081
[2]清华大学计算机科学技术系，北京100084
[3]北京技术与商业大学计算机与信息工程学院，北京102488
[4]密西西比大学计算机与信息科学系，牛津，38677，美国
*函件：xyong@mail。清华edu.cn

摘要：许多研究人员已经将区块链引入车辆互联网（IoV），以支持车辆之间的交易或其他认证应用。然而，传统的区块链不能很好地支持对发生在指定区域的事务的查询，这对车辆用户来说很重要，因为它们被绑定到地理位置。因此，事务的地理定位属性的查询效率对于基于区块链的应用程序至关重要。现有的工作不能很好地处理区块链中车辆的地理定位，因此查询效率存在问题。本文设计了一种区块链中区域交易的快速查询方法，包括数据结构和查询算法。其主要思想是利用Geohash代码来表示区域，并作为事务索引和查询的关键，并且地理位置被标记为区块链中事务的属性之一。进一步验证和

在著名的区块链以太坊实现的基础上，评估了该设计，结果表明，该设计获得了明显更好的查询速度比以太坊。

关键词：查询、区块链、地理信息系统、汽车互联网

check for updates

引文：周C.；陆H.；翔，Y。；吴J.；王，F。基于地理的区块链区域交易快速查询方法。传感器2022、22、8885。https://doi.org/10.3390/s22228885

学术编辑： Omprakash凯瓦提亚

收稿日期：2022年9月23日
接受日期：2022年11月15日
出版日期：2022年11月17日出版者注：MDPI对已出版的地图和机构附属机构中的管辖权主张保持中立。

版权所有：由作者提供的©2022。被许可方MDPI，巴塞尔，瑞士。本文是一个在条款和条件下发布的开放获取的文章知识共享的条件归属（CC BY）许可证。org/licenses/by/4.0/)。

1.介绍

近年来，车联网（IoV）的数据认证引起了[1]的广泛关注。g.，恶意车辆可以通过发送虚假的位置[2－6]来严重影响IoV信息的有效性。为了解决这一问题，许多工作都将区块链技术引入了IoV[7-13]。区块链是一种分布式账本技术，它通过散列将存储事务的块连接起来，并支持去中心化网络环境[14]中的数据完整性和可追溯性。一个典型的区块链结构是以太坊[15]，它是开源的区块链实现。以太坊的主要特点是，它使用默克尔帕特里夏树（MPT）来构建树结构来存储账户和交易信息。整个网络的事务信息构建成链结构，任何用户都可以查询发生的所有历史事务。

在IoV的场景下，车辆与地理位置相连，它们通常只关注本地信息[10,16,17]。与整个网络的信息相比，车辆周围局部区域的交易对用户来说更为重要。但是，传统的区块链结构不能很好地支持对发生在特定区域的事务的查询：用户需要回溯链结构上的所有事务，然后匹配交易发生的位置。此外，传统区块链上交易的地理定位数据的存储和访问，需要通过智能合约等外部应用程序来实现[7-9,18-21]，这进一步占用了额外的存储空间。

由于车辆的高动态性，车辆之间的数据链接的持续时间较短，区域信息的数据查询效率对应用[22－31]至关重要。因此，如何提高一个区域内信息的查询速度是一个关键问题。在IoV中尝试利用IoV中的链外结构来维护地理定位数据[7,9,18,32]，但是他们没有提出一种令人满意的区域信息查询方法。我们注意到，在电子地图中，地理区域通常由可变大小的[33,34]的地图瓷砖数据组成。最重要的是，我们的解决方案是使用地理块作为数据索引，来提高区域数据的查询速度。为了实现这一目标，我们使用了地理哈希代码[35]，这是一个一维的编码系统，可以有效地表示一个任意精度的网格。在这种情况下，存储在链中的与地理位置相关的数据通过地理位置代码进行索引，这可以很容易地通过地理位置进行查询。我们之前的研究[33]也表明，由Geohash代码编码的向量图数据结构在数据存储和数据压缩方面具有良好的性能。此外，为了更好地评价本文提出的数据结构和算法，我们修改了以太坊的核心结构，采用Geohash编码索引。

在本文中，我们提出了一种基于区块链的数据结构，用于快速查询发生在特定区域的事务。我们的主要贡献如下：

1. 基于默克尔-帕特里夏特里（MPT）[15]结构，我们设计了一个区域状态特里（RST），它使用地理哈什代码来索引层次区域的地理位置作为交易属性。
2. 在提出的RST的基础上，我们设计了一种分支查询方法来有效地查询层次区域的地理定位。此外，我们还设计了一个账户位置测试（ALT）来存储每个车辆账户的位置变化，以支持查询每个车辆的历史交易位置。

本文的其余部分组织如下。相关工作详见第2节。第3节概述了地理位置区块链结构，第4节将详细描述其方案设计。第5节讨论了实验和评价。最后，我们在第6节中总结了我们的论文。

2.相关工程

为了提高区块链数据查询[22－25]的效率，已经提出了一些工作。一般来说，它们可以分为四类。一个是链外查询。区块链仅用于维护数据安全，在区块链上建立独立的数据查询方法。g., IBM [23]为分类帐索引、状态数据、历史数据和块索引建立了四种不同的数据库，以实现分离查询。Oracle [25]使用支持丰富SQL查询的CouchDB作为存储数据库，用于为键和值创建索引，以提高查询效率。区块链的数据以虚拟表的形式存储在SAP [26]的数据库中，实现对区块链数据的链外查询。该方法主要用于数据库供应商。虽然查询效率有所提高，但数据被传输到区块链之外，其安全性在很大程度上依赖于可信实体。第二类是应用程序查询，它通过在区块链上构建不同的应用程序来实现不同的查询功能。沿着这条线，FlureeDB [27]是一个具有区块链核心和ACID标准的图形数据库，它为每个查询建立了一个与块或时间点相关的数据快照，以提高查询效率。Swarm [28]是一个区块链系统，用于处理各种类型的元数据的高级文件或分层集合，使用分层集的形式来提高查询效率。大链数据库[24]增加了基于大数据分布式数据库的区块链特性。节点根据查询需求建立不同的索引和查询api，实现不同的应用程序查询，提高查询效率。第三类是契约查询，它使用区块链的智能契约来优化查询功能。其中一个例子是vChain [22]，它使用智能契约来模拟区块链的底层存储结构，然后建立了模拟块的块间索引和块内索引。块和索引被重建

智能契约实现了链上的存储和查询，但与直接存储在区块链中相比，区块链中重构的数据量要大得多，区块链的内部查询方法没有改进。与vChain不同，G Gursoy等人。[29]没有重构块结构，而只是为智能契约中存储的内容建立了一个索引结构。第四类是分片查询，通过分片将大规模区块链划分为多个独立的部分，通过减少访问的数据量来提高查询效率。例如，chainSQL[30]将区块链划分为固定数量的碎片，并通过将事务分散到不同的碎片中来提高数据查询效率。一氧化物[31]将用户地址空间划分为多个区域，每个区域建立了一个独立的区块链，从而减少了每个区域内的数据量。该方法通过减少碎片内的数据量，提高了碎片内的查询效率，但其区域划分没有考虑实际的地理位置特征，不适用于地理位置相关信息的区域查询。一般来说，上述方法都依赖于信用实体，没有改进区块链的内部查询方法，也没有考虑到实际的地理位置。

一些作品已经试图通过智能合约[7,9,18]等高级结构，将IoV中的地理定位和区块链结合起来。表1是我们的方案与与地理定位和区块链相关的技术方案之间的比较。这些工作为提高区块链数据查询的效率以及将地理定位和区块链IoV应用结合起来做出了贡献。然而，它们没有很好地利用地理位置，因此区域查询效率值得怀疑。

表1。将我们的方案与与地理定位和区块链相关的技术方案进行比较。

文学	存储位置	编码方案	索引模式	功能
[7]	智能合同	经纬	–	合作的配置
[8]	智能合同	车辆之间的跳跃	–	可靠性
[36]	智能合同	经纬	水平的检查和垂直的检查	隐私位置查询
[9]	智能合同	经纬	–	提高达成共识的效率
[17]	交易	经纬	–	可靠证明
我们的计划	交易	土石灰	区域状态三角形（RST）	地区的消息查询

在[7]中，作者通过智能合约实现了车辆定位共享，以提高协同定位的准确性。Chukwoucha等。[8]利用车辆节点与事件位置之间的距离来衡量消息的可信度，并根据道路网划分多个重叠区域，以限制跨区域信息的可信度。Kudva等人。[9]读取了智能合约模拟的区块链交易中的地理定位和驱动距离，并提出了驱动共识机制以提高共识效率的证明，表明地理定位在IoV在区块链中的应用中具有很大的价值。张等人。[36]实现了基于区块链的隐私位置查询，设计了基于位置的匹配智能合同，并基于水平和垂直网格坐标检查返回匹配结果。这比一维地理位置查询方法的效率要低

由于从二维的角度进行查询。上述研究将地理定位存储在用户数据层中，不能使用区块链的安全机制来保证位置数据的安全性，因此不能保证区块链交易中地理定位的真实性。Shresta等人。[17]根据位置将全球区块链划分为多个独立的区块链，每个独立的区块链只接收其区域内的消息，从而提高了消息的可靠性，促进了车辆的可靠性验证。但是，本文并没有涉及查询区域信息的工作。

3.结构概述

如图1所示，以太坊的结构包括三种尝试：收据测试、世界状态测试和交易三种尝试。除此之外，我们还添加了一个区域状态trie（RST）来支持快速查询与分层地理位置相关的（区域）数据，以及一个帐户位置trie（ALT）来支持查询每辆车的历史交易位置。

以太坊采用了基于账户的数据模型。该块还包含事务执行日志的接收trie的根哈希和帐户状态数据的状态trie的根哈希，如图1中的块和区块链结构所示。

如图1所示，我们分别在块属性中添加了geohash编码的地理位置：帐户、事务和收据。我们将地理位置属性（帐户位置特里和区域状态特里）的索引结构的根散列添加到块中，以保持地理位置的一致性和可追溯性，便于地理信息查询和数据可信度验证。对于其他部分，包括共识机制，我们的结构遵循以太坊的现有设置。

1. 地理定位属性在图1中以蓝色的位置显示。每个车辆都与区块链账户有关联。我们将名为位置的地理位置属性添加到帐户中，以满足车辆的移动性，其中车辆位置按时间顺序记录，以方便历史位置查询。在事务中添加了地理位置，以实现向区块链添加地理位置属性的目的，区块链也用于验证和更新事务发送者的帐户位置。地理位置需要作为交易状态信息的一部分保存在收据中。
2. 地理位置属性的索引结构的根散列。与帐户余额一样，帐户位置特里也反映了帐户状态的变化。因此，它的根散列不需要写入块头，而是需要写入帐户状态。如图1的标头中的ALTRoot所示。区域状态特里记录地理区域内的全球变化，与账户状态的全球变化具有相同的状态。它的根哈希需要被写入块头中，如图1中的头的区域根所示。

为了便于查询与地理位置相关的属性，我们为它们构建了一个索引结构，并根据不同地理区域的查询需求设计了一种分支查询方法。

如果区块链中还没有账户位置的索引结构，那么车辆在执行位置验证或历史交易位置查询时，需要经历两个循环过程，即遍历块和遍历块中的交易。因此，应为每个账户建立一个账户位置特里。由于不同区域的车辆更注重其地理区域的信息，因此所提出的RST会根据地理区域的查询需求来存储地理信息。并进一步设计了一种分支区域信息查询方法，以降低了分支区域信息查询的开销，提高了分支区域信息查询的效率。

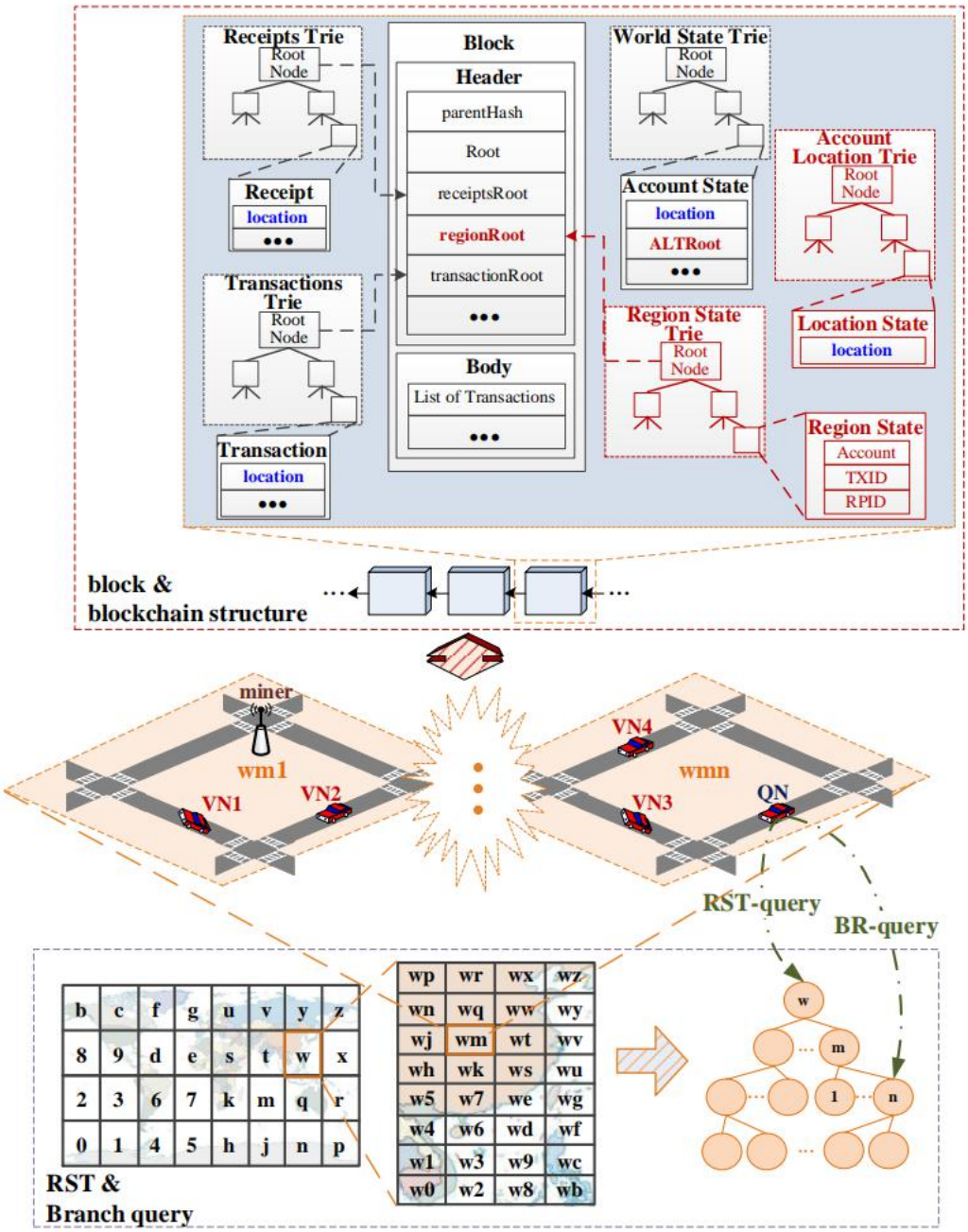

图1。提出的结构模型。其中，VN为车辆节点，QN为查询节点，查询节点RST-query表示查询节点根据RST方法查询区域信息，查询节点根据分支查询方法查询区域信息。

在提出的ALT中，为每个账户建立一个以时间为键、地理定位为值的MPT结构，实现区块链内支持地理区域信息查询的功能（详见4.1节）。在提出的RST中，一个区域状态的MPT结构，具有地理区域的层次关系，由指定地理区域内的交易量表示。我们使用地理位置的Geohash编码作为键，并将当前区域中的区域元素列表（包括帐户列表（帐户）、交易列表（TXID）和接收列表（RPID））作为此结构中的值（详见第4.2节）。

我们设计的分支查询方法要求根据节点的地理位置保存与地理区域对应的区域状态特里的分支索引结构，以缩小查询范围，提高查询效率（详见第4节）。.2.4

4.拟建结构的设计

4.1.账户位置测试（ALT）

为了支持查询每辆车的历史交易位置，我们将该交易位置视为区块链中相应账户的属性。与账户余额一样，所有账户位置也会作为状态信息写入每个账户的独立状态数据库中。

1.ALT的结构。

MPT作为以太坊的核心数据结构，是由默克树和帕特里夏树组合形成的单值识别密钥/值对的树状结构。MPT结构的优点是压缩前缀带来的查询效率和哈希合并带来的差异的快速定位。因此，我们使用MPT结构来构建ALT，以实现对地理信息的快速查询。

当事务发生时，ALT以帐户地理位置的记录时间作为索引，其中由miner节点验证的事务发送者的帐户地理位置存储在trie的叶节点中。然后，它将两个相邻叶子的哈希值组合成一个字符串，并将该字符串的哈希值存储在trie的父节点中。相邻的父节点重复上述哈希计算过程，直到获得根哈希值为止。为此，可以通过根节点来识别整个ALT的变化。如图2所示，ALT的存储由链上存储和索引结构存储组成。链上存储部分是块头中状态特里的根节点的散列值根。虚线框是索引结构，区块链的节点将在本地维护ALT的最新索引结构。帐户位置的更改将导致帐户状态ALTRoot中ALT的根节点的散列值的更改，这将导致块标头中的散列值root的更改。它仍然如图2所示，区块链添加了一个编号为N + 1的新块。由于将新时间为105且位置为wx4er5的新位置记录添加到ALT中，因此ALT的根节点的散列值从ALTRoot变为ALTRoot\，状态树根节点的散列值从root变为根.

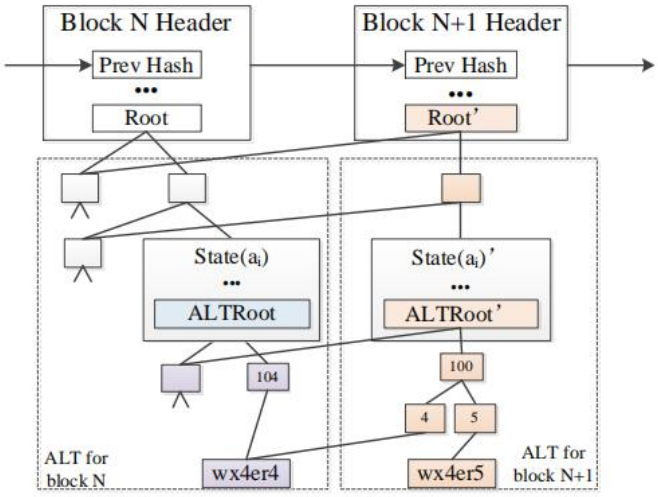

图2。ALT存储更换示意图。

2．正在查询帐户位置。

帐户位置查询可分为两种类型：最新的位置查询和历史上的位置查询。由于帐户的最新位置是帐户的一个属性，因此可以直接获取它。需要根据具体的时间来查询历史位置。首先查询最新的帐户状态，从帐户状态读取ALT的根节点，然后以指定的时间作为索引查询ALT，最后返回索引。

.2.4地区州界（RST）

ALT的目的是实现与指定帐户相关的位置查询。它还可以用于查询区域状态，而无需针对区域状态的独立查询函数。但是，由于基于账户和地理区域两个维度的查询过程，对区域信息的查询效率不高。我们设计了区域状态trie（RST），它使用地理位置编码作为MPT的索引，以支持分层地理定位（区域）状态的快速查询。RST记录某一地理区域内的区域状态信息，优化MPT索引方法，使其有效地支持区块链核心的前缀查询、最新的区域信息查询和分支查询。此外，我们通过提供一个供外部访问的接口来方便应用程序调用，从而减少了外部操作对区块链安全性的影响。

.2.1.4结构设计

图3为RST结构示意图。一般来说，我们使用地理哈什代码来索引地理区域、RST的分支节点和每个块的地理位置属性。我们进一步利用MBT（Merkle桶树）[23]设计了区域元素列表（Reel），在MBT的同一叶节点中存储多个区域状态信息。rel按时间顺序排列，方便查询最新信息。我们设计了一种前缀查询方法来查询RST中多个叶区域的信息。设计了一种分支查询方法，提高了指定分支区域内区域状态的查询效率。

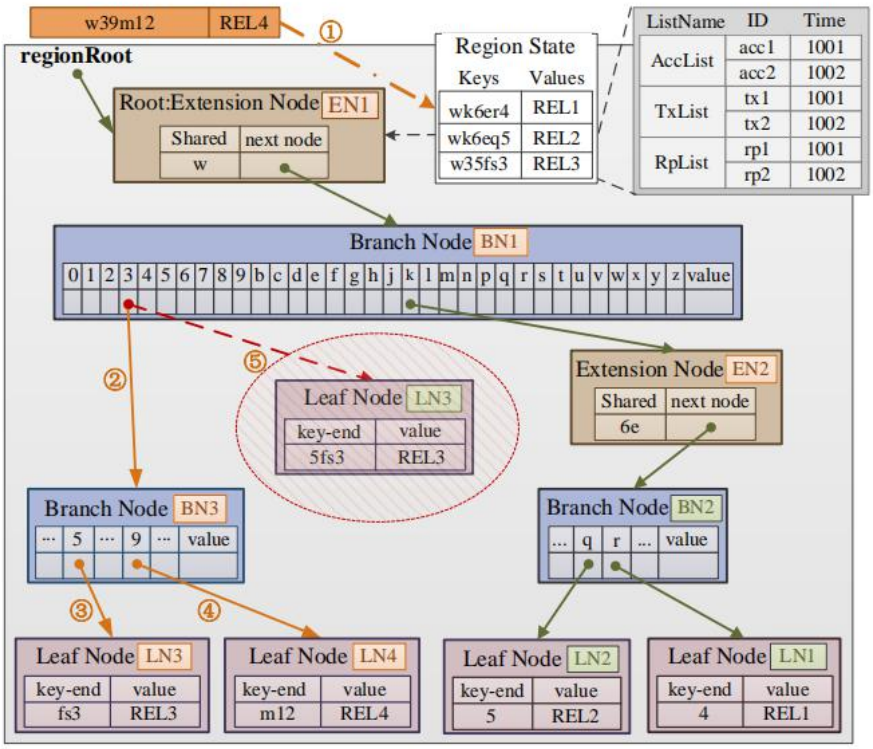

图3。区域状态三角结构示意图。

RST中的密钥用14字节的地理哈希编码，因为14字节的地理哈希代码可以满足地球上所有纬度的地理定位的厘米级精度要求。我们在RST中采用了递归长度前缀[15]的存储方法，以满足不同数据量的存储需求。

RST将该区域中的状态信息的REL作为值。在RST中有三种类型的节点，如下所述。

扩展节点：在从根节点遍转到此节点时，记录所有未写入节点的公共前缀（共享）和后面的分支节点索引（下一个节点）。作为键的一部分，公共前缀也使用地理哈希代码。构造扩展节点的目的是扩展树结构的分支，因此每个扩展节点将遵循一个分支节点。

分支节点：记录由指向此节点的不同键所遍历的分支路径。根据Geohash编码规则，该节点的前32项记录分支，第33项记录键结束标志。值得注意的是，如果在分支结之前没有组合的公共前缀，则在分支节点之前没有扩展节点。如图3中的分支节点BN3所示，上一个节点是分支节点
BN1。

叶节点：记录无分支的剩余键编码值和对应的REL。一个只有一个项的RST是一个叶节点。当RST只有一个项时，它是一个叶节点。

4.2.2.RST的构建和更新

如我们前面提到的，区域状态信息以REL的形式存储在RST中。REL包括帐户列表、交易列表和接收列表。区域元素包含两个属性： ID和时间。ID分别指账户列表、交易列表和收款列表中的账户ID、交易ID和收款ID。REL包含了地理区域中的所有区域元素，并且在任何两个REL中都没有重复的数据。对于REL中的帐户列表，只有帐户的最新交易时间被保留，以指示帐户的活动。

RST建设。RST的构建包括自上而下的存储路径规划和自下而上的节点值散列合并两部分。

1. 存储路径规划。存储路径规划是指根据地理定位编码REL，从根节点到RST的最终叶节点的所有分支节点的构建过程。RST的初始状态为空，第一个数据被写入根节点。当要规划的区域中存在REL时，将更新此REL；否则，将当前区块链所在的地理区域编码用作规划存储路径的公共前缀。算法1总结了RST的存储路径规划过程。
2. RST的节点哈希值合并过程与ALT的合并过程一致。它们都需要从叶节点开始，并组合相邻的节点来计算合并的哈希，并将其传递出去，直到获得根哈希值。

RST的更新。RST的更新过程包括三个部分：存储路径的更新、REL的更新和节点散列值的更新。当要更新的密钥生成的RST的存储路径与树中的现有路径不重叠时，需要建立RST的一个新分支，即存储路径更新。当要更新的键生成的RST的存储路径与树中的现有路径相同时，需要将新的REL更新到原始的REL，即REL更新。以上两个更新过程需要伴随着节点哈希值的更新过程。

以图3中要更新的项目（步骤O1），键为w39m12，值为REL14为例，说明RST的更新过程。当查询分支节点BN1的分支3时，记录不是空的，需要比较叶节点的键端。由于REL4的密钥编码被读取到9m12，原叶节点LN3的密钥为5f s3，且第一次编码不同，因此它将进入存储路径更新

②过程，即建立一个分支节点（步骤），然后分别创建键为f s3和m12的叶节点LN3和LN4（步骤和）。③④然后进入节点值哈希更新过程，即将叶节点LN3和LN4的哈希值分别写入分支节点BN3的5个和9个分支。最后，删除原始叶节点LN3（步骤），并完成更新过程。⑤

算法1：RST的存储路径规划算法

要求：R，w

```
1: 函数STORAGE_PATH_PLANNING（R，w）
2: //w是R所属的地理哈希区域编码
3: //cur_节点是前缀为w的最长的节点
4: //sub_reean是w去掉cur_node和w的前缀
5: 如果sub_区域。len > 0然后
6: 叶节点[子区域（1：结束）]=R
7: 分支节点[sub_区域(0)]=
                LeafNode[cur_node。键（1：结束）
8: 如果cur_node。≠然后键入分支节点
9: 叶节点[cur_node]。键（1：结束）]=cur_node
10: 分支节点[cur_node]。键(0)]=
                LeafNode[cur_node。键（1：结束）
11: 如果结束
12: 其他的
13: 如果cur_node。然后键入==分支节点
14: 分支节点。value=R
15: 其他的
16: R被存储在REL中
17: 如果结束
18: 如果结束
19: 结束函数
```

4.2.3.前缀查询

RST支持MPT查询和前缀查询。MPT查询要求查询区域的编码字节长度和构造树的键的编码字节数相同。且查询结果为空，或为叶节点的值。为了根据地理位置编码的前缀来查询不同级别的区域信息，我们设计了一种前缀查询方法。前缀查询要求要查询的区域的地理哈什编码作为RST键的前缀。且查询结果为空，或为前缀所在分支的所有叶节点值的组合值，如图4中的粉红色部分所示。前缀查询包括两部分： RST查询和查询状态缓存。

1．RST查询。设计了一种前缀匹配方法来实现RST的前缀区域查询。我们使用输入来表示要查询的区域的编码，并使用cur_key来表示在前缀匹配过程中要查询的当前密钥的编码。整个前缀匹配过程是从输入编码的第一个字节逐字节比较过程，然后得到cur_key。当第一次与RST的根节点进行比较时，输入为cur_key。当父节点的键是cur_key的前缀时，cur_key是没有前缀的父节点的其余部分。如果遍历输入后没有到达叶节点，则将父节点子节点的键设为cur_key，逐个遍历子节点，直到遍历父节点的所有叶节点。

2．查询状态缓存。获取IoV的区域地理信息最终需要查询REL。当待查询的REL没有更改时，缓存查询状态可以提高REL的查询效率。我们设计了一种查询状态缓存方法来临时保存REL的查询结果。如图4中的蓝色部分所示，我们使用QSC来表示查询状态缓存列表，使用区域作为区域状态缓存的键来记录查询的区域编码

请求时，REL存储与该区域对应的REL。query_hash是查询状态高速缓存的哈希值。在查询区域信息时，我们首先检查该元素在查询状态缓存列表中是否存在。如果不是空的，则将查询状态缓存的哈希值与RST的节点进行比较。如果RST对应元素的哈希值与查询状态缓存列表中对应元素的哈希值一致，则直接返回查询状态缓存结果。如果元素不在查询状态缓存列表中，或者元素存在但不是最新状态，则需要完成RST的查询过程，然后应该更新查询状态缓存列表的哈希值和缓存中的查询状态缓存。

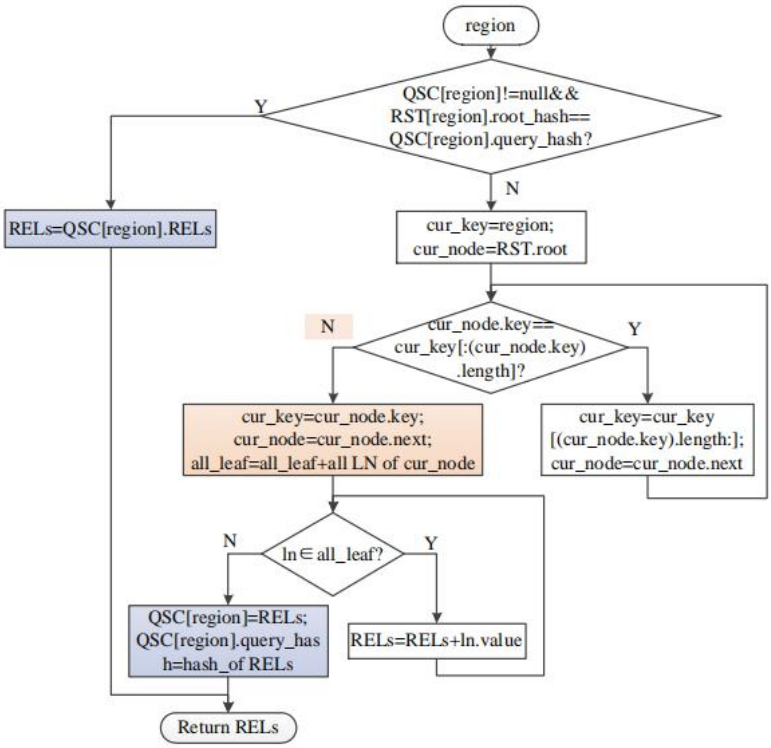

图4。RST的前缀查询的流程图。

.2.4.4分支机构查询

与区域外的地理信息相比，车辆需要区块链节点提供的自身区域的地理信息，以满足IoV服务的需求。我们提出了一种分支查询方法，其中区块链节点缓存其自身地理区域的区域状态数据，以提高分支查询的效率。

1.分支区域缓存。分支区域高速缓存的内容包括根节点、该节点所在的地理区域的RST分支，以及分支节点的前缀。本文将分支节点前缀定义为该节点在全局RST中缓存的RST分支的所有父节点键的编码长度。图5显示了分支查询方法的建立和更新的示意图。在图中，编码区域ID的地理区域为空的节点采用全局RST查询方法，编码区域ID为wm1的节点采用分支查询方法。图中蓝色虚线框表示区块链节点在某一点的存储状态，包括两部分：橙色框表示当前节点分支区域的状态缓存，下部表示区块链的链链关系。在图5中为节点wm1建立分支区域缓存的过程如下： 01读取的根节点

②③④⑤最新块B3的全局RST，查询全局RST中wm1的分支，获取当前区域的根节点，即键为11的b_reg_root（wm1），获取分支区域节点pre_num（wm1）的前缀编码字节，计算RST分支根节点的哈希值（wm1）。

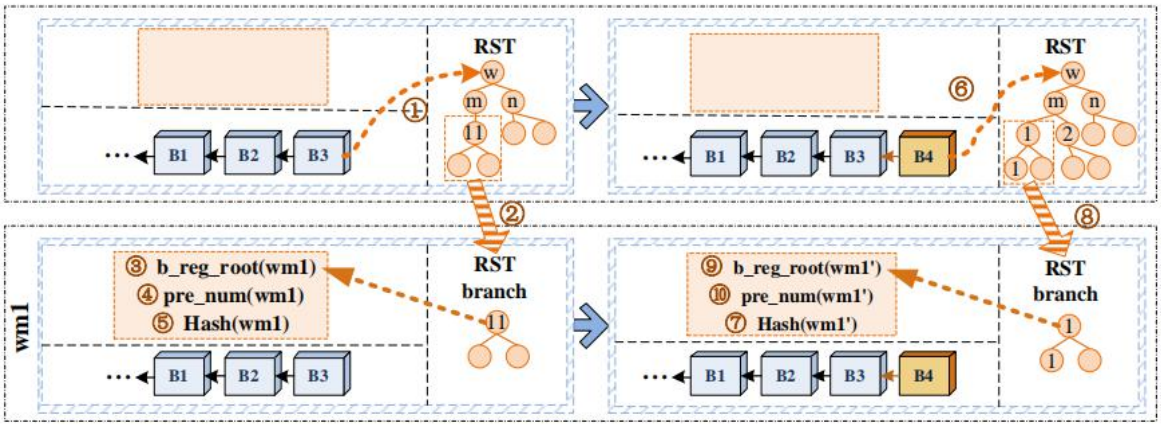

图5。分支查询方法的示意图。

使用分支查询方法的节点退出后，其分支区域缓存无效，在重新启动时需要重新建立该缓存。算法2显示了寻找分支区域状态的根节点和分支节点的前缀的算法。

算法2分支区域状态缓存

```
输入：RST。根，区域ID
输出： b_reg_root，pre_num
1: v_reg = regionID
2: pre_num=0
3: b_reg_root = RST .根
4: 而v_Reg则不是b_reg_root的前缀。键做
5: pre_num = pre_num +伦(b_reg_root。钥匙
6: v_reg = v_reg删除了b_reg_root.key的前缀
7: b_reg_root = b_reg_root .下一个的
8: 结束时
9: 返回b_reg_root，pre_num
```

2．分支区域更新。当使用分支查询方法的节点的区域ID对应的区域状态分支发生变化时，需要更新本地分支区域状态缓存。随着RST的结构的改变，区域ID对应的分支的哈希值和前缀也会改变。因此，需要同时更新分支节点的哈希值和分支节点的前缀。如图5所示，在节点wm1完成块B4的同步过程后，生成一个新的RST（步骤06），然后生成RST hash的分支的根节点哈希(wm1\)需要进行计算（步骤07)。哈希(wm1\)≠散列（wm1）表示分支已经更改，重新搜索节点的RST分支（步骤08），并将分支区域状态缓存更新为b_reg_root（wm1\）和pre_num (wm1\)⑨⑩（分别为步骤和0）。

3.分支查询。如果分支区域的地理哈什编码和要查询区域的地理哈什编码前缀相同，则可以通过使用该区域的本地缓存的RST分支来实现对分支区域状态的快速查询。不需要从RST的根节点启动查询，而只需要从高速缓存的分支节点启动查询。如图5所示，wm1的节点查询wm1或wm11范围的区域状态，这是一个分支查询。当节点没有出现时

设置要查询的区域，或者当实际查询区域是节点区域的上部区域或未作为节点当前分支区域前缀的其他区域时（因为没有相应的区域状态缓存），需要在对应区域状态的全局RST中进行查询。

4.3.关于数据安全的探讨

本节将分析地理定位区块链的存储安全性和查询安全性。

存储安全。

1．区块链采用了非对称加密算法，可以抵御许多传统的安全攻击，其分布式存储结构保证了存储的安全性[7-12]。

2．位置通过事务写入块，可以根据分布式共识机制和同步机制实现一致性和抗篡改性。

3．地理定位属性被添加到区块链的核心中。虽然存在暴露位置的风险，但通过向交易添加位置属性来增加交易的真实性，另一方面，这提供了安全保证。

查询安全性。

1．隐私。原始的区块链不提供位置信息，也没有暴露该位置的风险。我们在地理定位区块链中添加了位置属性。虽然交易的真实性可以通过位置来证明，但如果位置访问不受限制，就存在隐私泄露的风险。因此，根据位置信息的应用场景，我们根据位置访问权限是否为交易中的双方一方，将位置访问权限分为内部索引使用和外部使用。在区块链的内部索引期间，对位置信息的访问是无限的。在其他情况下，只有双方才能访问特定的位置信息，以达到限制位置信息使用的目的。

2．输入是不可重复的。这是因为在同一帐户中一次只能发生一个事务和一个位置输入，特别是ALT以时间作为键，并且键不会重复。也就是说，没有任何情况下，一个时间会对应于ALT中的两个位置。

3．查询可靠性。当查询同一区域的状态时，任意两个节点都会返回相同的查询结果。假设有两个节点查询相同的区域状态信息。由于整个网络中的诚实节点占了大多数，所以所有节点的RST的根节点哈希都是相同的，所以不会有不同的分支。这与假设相矛盾，这也证明了查询的可靠性。

通过上述安全性分析可以看出，我们的解决方案对区块链的安全性没有影响，同时也保证了查询的可靠性。尽管存在暴露帐户隐私的风险，但它可以通过添加简单的查询限制来进行改进。

4.4.讨论查询时间

我们分析并讨论了本小节中在无位置索引方法、ALT方法、全局RST方法和分支查询方法中查询区域状态信息的查询时间。

1.无位置索引方法。

现有的区块链，如以太坊，没有针对地理定位的索引方法，我们称之为无位置索引。有两种方法可以向以太坊添加位置信息。一是使用智能合同来存储办公地点，但他们

不能与普通事务相关联。如果使用智能合同来实现链上的存储和访问数据，数据存储和存储成本将会增加。另一种方法是以额外数据的形式在事务中存储位置信息。位置数据需要以递归长度前缀编码格式存储，然后写入普通事务。查询过程主要包括块查询和事务查询的循环搜索过程。在获取事务位置时，需要完成递归长度前缀解码过程，然后与要查询的范围进行比较。我们选择后者进行比较分析。

假设块数为m，交易量为n，取平均值

在每个块中打包的事务量为，则查询时间的复杂度

为O (m) = O (n)。

2. ALT方法。

通过ALT查询区域状态的步骤如下：首先，获取所有帐户id。为了从ALT的角度（主要是指交易的数量）来获取整个区域的状态，我们首先需要统计该区域内的所有历史账户信息。第二，获取该账户的所有交易信息。我们需要获取查询的开始时间和结束时间。开始时间是帐户发送第一个交易的时间，结束时间是帐户发送最新交易的时间。由于事务的随机性，我们可以从块生成时间来确定它。我们选择方块0，i。e.，将起源块的时间戳作为开始时间，因为起源块不包含任何事务。我们选择当前区块链中最新块的时间戳作为结束时间，以及时实现完全覆盖。第三，交易记录位置过滤。在步骤2的过程中，还需要根据查询区域来过滤事务。根据Geohash编码规则，只需要检查查询区域是否为事务位置的前缀。

假设块数为m，交易金额为n，账户数量为a，则每个账户的平均交易金额为。ALT是一棵Merkle树，其查询时间复杂度为O（logN），N为叶节点数[37]。ALT以时间为关键，每个账户都有一个

独立的ALT。因此，叶节点的数量是的事务处理数量

因此，每个帐户的查询时间复杂度是日志，和整体

查询时间复杂度为O（a log）O（log n）。因此，查询时间复杂度

ALT法小于无位置索引法。

3. 全局RST方法。

由于RST的存储路径规划过程中没有循环过程，这在算法1中被总结出来，其时间复杂度为O (1)。规划过程独立于查询过程，不会影响全局RST方法的查询时间复杂度。

RST也是一个Merkle树，而Merkle树的查询时间复杂度为O（logN），其中N是叶节点的数量。由于RST存储在叶节点的桶结构中，我们假设在RST的叶节点的REL中存储的事务数为S。则RST中的叶节点数为N /S，全局RST的查询时间复杂度为O（log（N/S））。因此，全局RST方法的查询时间复杂度小于ALT方法。

4. 分支查询方法。

算法2中总结的寻找分支区域状态的根节点和分支节点的前缀的算法的时间复杂度取决于待查询的分支区域的编码长度和RST中公共前缀的长度。在最坏的情况下，如果RST中每个分支的编码长度为1，则算法2的时间复杂度为O (n)。由于此过程是一个可以在节点启动时完成的操作，因此它不需要等到

查询区域信息，不会增加对区域信息的查询时间。

假设Global RST方法的叶节点数为N/S，且RST每层最多有L个分支。根据算法2，我们假设在分支区域查询方法中缓存的前缀区域码的长度为P。那么在分支区域查询方法中缓存的RST的叶节点数为$\frac{1}{L^{P}}*\frac{N}{S}$，小于Global RST方法。我们可以得出这样的结论

分支查询方法的查询时间复杂度为$O(记录\frac{1}{L^{P}}*\frac{N}{S})$，这是较少的比全局RST方法更好。

在具有相同事务内容和相同数据量的区域状态查询中，查询时间的顺序为无位置索引方法> ALT方法>全局RST方法>分支查询方法。与文献[37]中Merkle树的查询时间复杂度相比，我们的查询方法与文献[37]的数量级相同，这表明我们的查询方法虽然添加了位置信息，但对原始Merkle树的查询优势没有影响。

5.实验与评价

我们已经基于以太坊实现了所提出的数据结构。在本节中，我们首先介绍在本工作中使用的实验环境和实验数据。然后，我们用区域状态查询函数来评估地理定位区块链结构的性能。

5.1.实验设置

1. 实验环境。我们的原型框架是基于以太坊版本1.9.12（2020年3月16日）实现的，该版本是用Go语言编写的。该实验的物理环境是一台拥有4核英特尔酷睿i5-4690k 3.5 GHz处理器和7.7 GB内存的台式电脑，运行在Ubuntu 14.04上。
2. 数据生成。设定车辆每200 ms发送一次位置交易，一般城市道路的限速为30-50 km/h [7]，因此我们设定平均车速为40 km/h作为车速。车辆轨迹是在指定区域内按指定速度连续生成的14字节地理位置编码所表示的位置序列。车辆节点将车辆位置写入事务中，以记录车辆的移动。无位置索引区块链以普通事务的附加数据的形式将位置数据存储到区块链中，地理定位区块链结构以事务的位置属性的形式进行存储。每个车辆节点每200 ms向区块链发送一个事务。
3. 数据采集。为了评估原型框架的性能，我们主要使用三个指标： (1)将相同内容和相同交易数量的时间作为构建时间，比较构建不同区块链时的时间成本。(2)取写入区块链的内容相同、交易量相同的链上数据和本地数据量，比较构建不同区块链时的空间成本。由于不同区块链节点存储的数据内容不同，我们分别从链上的数据量和本地数据量进行统计。链上的数据量：只计算数据块本身的数据量。本地数据量：统计本地保存的完整数据量，包括状态数据。(3)将同一区域内事务量的查询时间作为查询时间，表示不同类型的查询方法对区域信息的查询效率。对于构建时间和数据量的统计，我们分别计算事务量为8000、16000、24000、32000、40000时的情况。对于查询时间的统计，我们设置查询节点，每10秒完成一次区域信息查询，并取200个连续查询的平均值。

4.实验场景。四个车辆节点代表四个不同区域的车辆，在各自的区域生成车辆位置(每个节点位置的移动范围为4个5字节Geohash编码区域，区域约为
80 km2)，每200 ms发送一次交易。一个固定节点充当矿工，负责打包事务。有一个查询节点负责查询工作。

5.比较协议。与前一节一样，我们的比较协议包括了原始的无索引法、ALT法、全局RST索引法和分支区域查询法。
原始无索引法（ORG）如第4.4节O第一段所述。在原始的以太坊中，交易的位置数据以附加信息的形式存储。由于区块链中没有针对附加信息的索引，因此需要使用块和块内事务来完成查询。这个没有专用位置查询的方法被称为原始的无索引方法。
ALT指数法（ALT）如第4.4节第二段所述。在地理位置区块链中，ALT用于从帐户的角度查询区域状态。这种方法称为ALT索引方法。
全局RST索引方法（RST）如第4.2.1－4.2.3节中所述。在具有RST的地理定位区块链结构中，节点使用全局查询方法来完成区域状态查询工作，称为全局RST索引方法。
分支区域查询方法（BR）如第4.2.4节所述。在带有分支查询方法的地理定位区块链结构中，节点使用分支查询方法来完成对当前分支区域状态的高速缓存的查询工作。

5.2.实验结果

在本节中，我们将从不同的方面来评估支持区域状态查询的地理定位区块链结构原型的性能。首先，我们测量了无位置索引区块链和地理位置区块链的构建时间和数据量这两个性能指标。然后，对在不同交易量下查询不同地理区域时的原始无索引方法（ORG）、ALT指数法（ALT）、全局RST索引法（RST）和分支区域查询方法（BR）的查询时间进行了定量分析。最后，进一步分析了节点区域范围和查询范围对分支区域查询方法的影响。

1.构建时间。我们将相同数量和相同交易内容的成功打包时间作为构建时间，分别对8000、16000、24000、32000、40000个案例的构建时间进行统计。图6显示了构建时间的比较。地理定位区块链结构和无位置指数区块链的构建时间和时间增长趋势基本相同，两者总体上都随着交易量的增加而增加。与无地理位置指标区块链相比，地理位置区块链结构的构建时间平均增加约0.31%，说明虽然索引结构添加到地理位置区块链结构中，但构建时间没有显著增加。
由于我们的研究没有修改共识机制，在发送事务速度一致的情况下，单矿和多矿对施工时间测试的影响很小，所以本文没有涉及多矿实验。

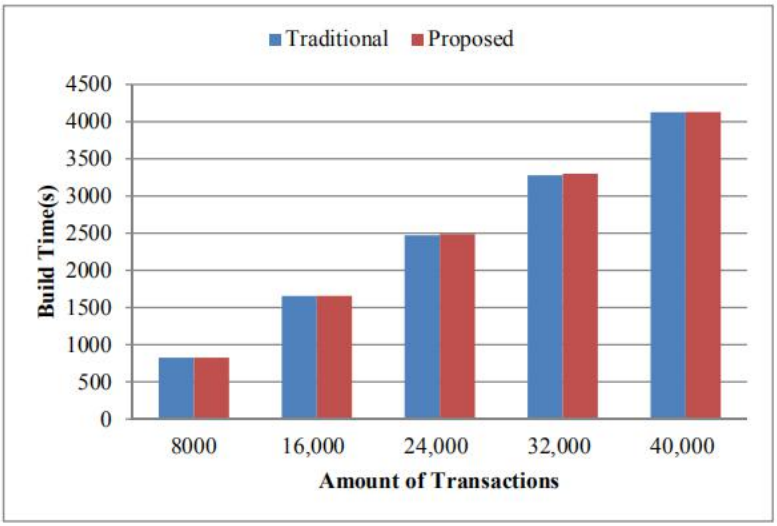

图6。使用不同数量的事务来构建时间。

2．数据量。数据量统计的实验设置与建筑时间统计的实验设置相同。数据量的比较结果如图7所示。

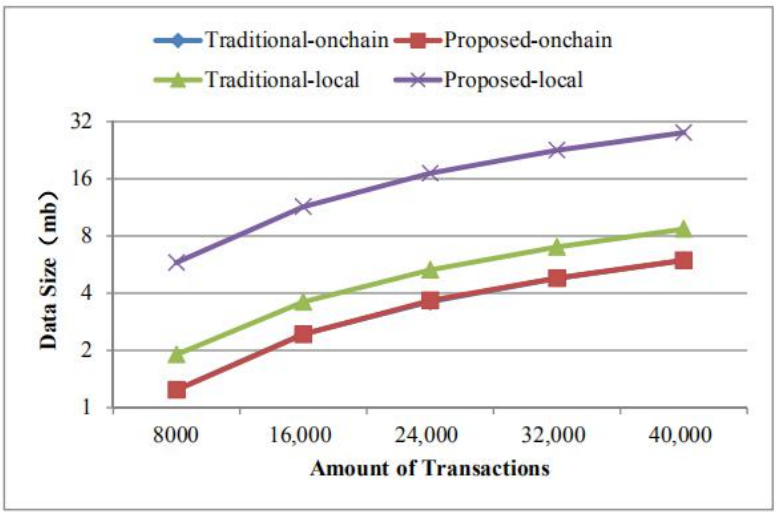

图7。具有不同交易量的数据量。

对于链上的数据量，如图7所示，在这两种情况下，链上的数据量都随着事务数量的增加而增加。与无位置指数区块链相比，地理位置区块链结构的链上数据量略有增加，平均增长约0.27%。这是因为地理定位区块链结构只在块头中添加了ALT和RST的根节点，并且块体中包含的事务中包含的数据量保持不变。因此，地理定位区块链结构对链上的数据没有显著影响。

从本地数据量的角度来看，粗略地说，两种方法的数据量都随着事务处理数量的增加而增加。地理定位区块链结构的数据量显著增加，平均约为无位置指数区块链的3.2倍，说明地理定位区块链结构的数据库数据量明显高于无位置指数区块链的数据量。原因如下：(1)每个账户建立的ALT的存储继续增加；(2)RST的继续存储

根据该区域交易数量的增加而扩大。数据库数据量的统计结果表明，地理位置区块链结构的索引结构随时间交换空间。

3. 查询时间。查询时间是指当链上的事务量相同时，使用不同的索引方法查询同一地理区域内的事务量所花费的时间。这里我们根据固定的交易间隔量计算累积平均值，并分别以3-6字节的地理哈编码查询范围计算地理区域内交易数量的查询时间。统计结果如图8所示。图8中分支查询索引的区域节点所选择的分支区域与查询区域所选择的Geohash范围相同。并且，在图8a-d中的四个查询结果图中，与分支区域索引对应的节点的查询范围也有所不同。

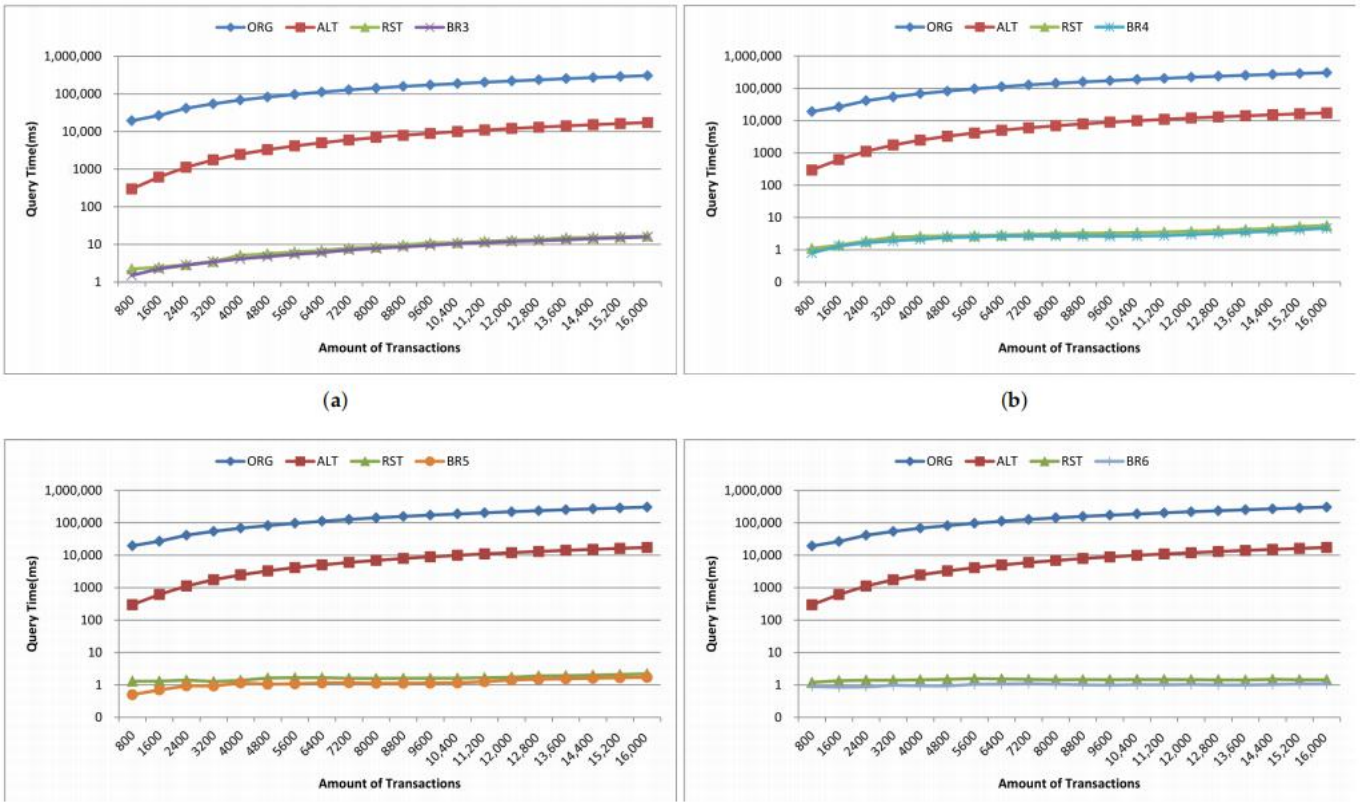

图8。不同地理区域内不同事务量的查询时间。(a)查询范围是3字节地理哈希编码的地理区域。(b)查询范围是4字节地理哈希编码的地理区域。(c)查询范围是5字节地理哈希编码的地理区域。(d)查询范围是6字节地理哈希编码的地理区域。

总的来说，当查询范围为3字节的地理哈希范围时，由于该区域的大规模，该区域中包含的事务量会快速增长。因此，四种查询方法的查询时间随着此范围内事务量的增加而大大增加。当查询范围为6字节的Geohash范围时，查询时间会增加，但速度很小，因为该区域很小，且该区域内包含的事务数量增长缓慢。

比较ALT索引与原始无索引方法，当查询区域在3-6字节Geohash范围内时，原始无索引的平均查询时间为19s-306s，ALT索引的平均查询时间为0.3s-17s。ALT指数的平均查询时间约为原始无索引的5.3%，说明ALT指数具有很大的差异

与原始无索引的查询区域状态相比，提高了查询效率。

比较全球RST指数和ALT指数，当查询区域在3-6字节的范围内，全球RST指数的查询时间平均为1.4毫秒-9.2毫秒，至少比ALT指数少99.8%，表明RST的全球RST指数有更好的区域查询效率。

比较分支区域索引和全局RST索引，当查询区域在3-6字节地理值范围内，区域索引的查询范围也在相应的3-6字节地理哈范围内时，分支区域索引的平均查询时间为1ms-8.4ms，比全局RST指数低约21.7%，说明分支区域索引也提高了RST的效率。

这些实验结果验证了第4.4节中对查询时间的讨论。

4．分支区域查询方法的分析。

我们进一步评估了在同一查询范围内的不同区域节点的地理区域下的分支区域索引查询方法。统计结果如图9所示。图9中的BR3-BR6表示在区域状态信息查询过程中用于表示不同区域的Geohash编码的长度。

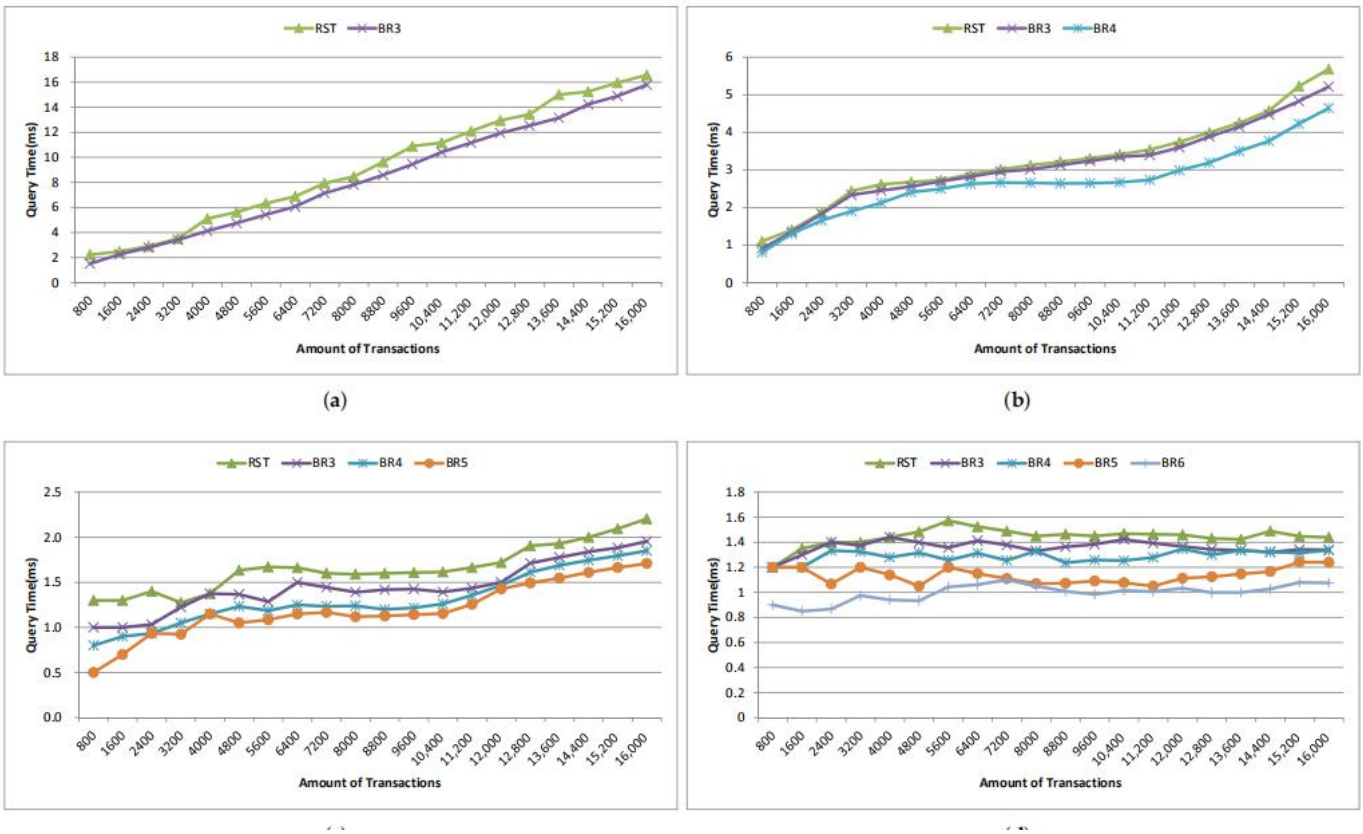

图9。指定地理区域内不同事务量的分支区域查询方法的查询时间。(a)查询范围是3字节地理哈希编码的地理区域。(b)查询范围是4字节地理哈希编码的地理区域。(c)查询范围是5字节地理哈希编码的地理区域。(d)查询范围是6字节地理哈希编码的地理区域。

当事务量较小（约1200）时，全局RST索引的区域查询时间与3字节地理网格编码时分支区域索引的区域查询时间相同。随着事务量的增加，分支区域查询方法的查询优势逐渐提高。它也可以从

结果表明，查询时间受查询范围内的事务量的影响很大。当查询范围内的事务量变化很大时，查询时间几乎相同。例如，当图9b中的事务量为5600-10,400时，BR4的查询时间基本相同。另外，在图9d中，由于查询范围较小，该区域内的事务量较小，且变化较小，使得查询时间变化不大。

当节点区域与查询范围一致时，分支区域查询方法的查询时间比全局RST索引平均减少了约21.7%。当查询范围在4-6字节地理哈什编码范围内，区域索引范围为3字节时，分支区域查询方法的查询时间平均比全局RST索引少约7.6%。当查询区域在5-6字节地理编码范围内，分支区域查询方法的范围为4字节时，区域查询方法的查询时间平均比全局RST索引少约16.2%。当查询区域在6字节地理哈希编码范围内，分支区域查询方法的范围在5字节范围内时，区域查询方法的查询时间比全局RST索引的查询时间平均减少了约21%。结果表明，当区域查询方法的范围一致时，分支区域查询方法的查询效率高于全局RST索引，当查询范围与区域范围大致相同时，分支区域查询方法的查询效率更高。这对区域状态查询和区域节点的区域选择起着指导作用。

5.3.讨论

以上实验结果表明，在建设时间和链上数据量方面，地理位置区块链较无位置指数区块链的平均增长仅为0.31%和0.27%。在本地数据量方面，由于地理位置的索引结构，地理位置区块链的数据量平均为无位置索引区块链的3.2倍，说明地理位置区块链使用存储空间来换取查询效率。对于查询时间，地理位置区块链的ALT索引的查询效率为原始无索引查询的5.3%。与ALT索引相比，全局RST索引的查询时间减少了99.8%。与全局RST索引相比，分支区域索引的查询时间减少了21.7%，说明地理定位区块链在区域状态查询方面具有明显的优势。此外，我们还进一步得出结论，节点的地理定位范围与分支查询范围越一致，查询效率就越高。通过对数据安全性、查询时间和实验结果的讨论，验证了地理位置区块链在区域信息查询中的有效性。

6.结论

本文为了在IoV中快速获取区域状态信息，设计了一种基于Geohash编码系统的区域状态测试方法，以对块中的分层地理定位属性进行索引。我们进一步设计了一个帐户位置特里，以支持历史交易位置的查询。此外，我们还提出了一种分支查询方法来提高区域查询的效率。我们的设计是在开源以太坊的基础上实现的，而不是使用在上层建立智能契约等逻辑结构，这更方便未来的开发和应用。

目前，对区域状态的查询只涉及一个全职查询，目前还没有对该时段的查询进行进一步的研究。它也没有考虑区块链的物理存储、动态结构调整和分区共识等问题。这些都是实现物理多链的有趣的研究方向
适合IoV。

作者贡献：概念化与方法论，C。Z.和Y。X.；软件，验证和调查，C。Z.；写作-原稿准备，C。Z.；写作-评论和编辑，Y。X.， H.L.， C.Z.， J.W.和F。W.所有作者均已阅读并同意该手稿的出版版本。

资助：本研究由北京市教委研发计划项目KM202210011012资助，国家自然科学基金项目61972010资助。

机构审查委员会的声明：不适用。

知情同意声明：不适用。

数据可用性声明：不适用。

利益冲突：作者声明没有利益冲突。

参考文献

1. Kaul，A.；Altaf，我。Vanet-TSMA：一种针对车辆自组网中智能道路交通的交通安全管理方法。Int。J.通勤。西斯特。2022， 35， e5132.CrossRef
2. 格罗弗，J。使用区块链的车辆专用网络的安全性：一个综合的综述。Veh。通勤。2022， 34， 100458.CrossRef
3. W.；西、C；李源Z；淑芬Z；道顺，W。车联网秘密共享中的一种多秘密声誉调整方法。安全。通勤。网络。2022， 2022， 1413976.
4. 马斯基。R.；Badsha，S.森塔；哈利勒，我。以区块链的VANET应用智能：以事故验证为案例研究。影响过程。马纳格。2021， 58， 102508.CrossRef
5. 赵，L.；德彪，H.；信义，H.；Neeraj，K.；金光雷蒙德，C。BCPPA：一种基于区块链的车载专用网络条件隐私保护认证协议。IEEE跨。知识翻译。西斯特。2020， 22， 7408 - 7420.
6. 吴J.；陆H.；翔，Y.；王，F.；李，H。SATMAC：基于自适应tdma的MAC协议。IEEE跨。知识翻译。西斯特。2022， 23， 21712 - 21728.CrossRef
7. 李C.；傅Y.；余，F。R.；Luan，T。H.；张。车辆位置校正：一个基于车辆区块链网络的GPS错误共享框架。IEEE跨。知识翻译。西斯特。2021， 22， 898 - 912.CrossRef
8. c；p；Thulasiram，R。K.在VANET上建立信任和可扩展的基于区块链的消息交换方案。对等网络。应用程序。2021， 14， 3092 - 3109.CrossRef
9. Kudva，年代；Badsha，S。；森古普塔，S.；哈利勒，我；Zomaya，一个。为基于区块链的VANET达成安全和实际的共识。影响科学。**2021， 545， 170 - 187.**CrossRef
10. R.Shrestha；南。Y.区域区块链的车辆网络，以防止51%的攻击。IEEEAccess2019,7,95033-95045。CrossRef
11. 杨z。杨K.；雷L.；郑K.；梁，V。C.M.基于区块链的车辆网络中的分散信任管理。IEEE互联网东西J。2019， 6， 1495 - 1505.CrossRef
12. 康，J.；熊，Z.；Niyato， D.；也门 D.；基姆 D.I.；赵，J。面向安全的支持区块链的车联网：利用声誉和契约理论优化共识管理。IEEE跨。Veh。技术。2019， 68， 2906 - 2920.CrossRef
13. 艾哈迈德，W。；迪，W。；Mukathe，D。在vanet中，基于区块链的身份验证和信任管理。IET Netw.2022， 11， 89 - 111.CrossRef
14. 中本，S。比特币：一个点对点的电子现金系统。可在线获得：http://bitty。org/bitcoin.pdf（已于2022年9月20日访问）。
15. 木材，G。以太坊：一个安全的分散化的广义交易分类账。2014.可在网上获得：https://以太坊。吉图布。io/yellowpaper/paper.pdf（已于2022年9月20日访问）。
16. 梅杰斯，J.；米哈洛洛斯，P.；年代；张；张；Veneris，A.；雅各布森，H。A.区块链for V2X：应用程序和架构。IEEE开放J。Veh。技术。2022， 3， 193 - 209.CrossRef
17. Shrestha，R.；巴，R.；Shrestha，A。P.；南。Y.一种用于VANET中安全消息交换的新型区块链。数字。通勤。网络。**2020， 6， 177 - 186.**CrossRef
18. Alladi；查莫拉，v；新；Guizani，M。区块链在车辆网络安全中的应用综述。IEEE通勤。服务。导师。2022， 24， 1212 - 1239.CrossRef
19. Jamil，F.；易卜拉欣，M.；乌拉，我；金，S.；Kahng，H。K.；金，D。H.基于物联网区块链网络的农业自主温室环境最优智能合约。压缩。电子农业。2022， 192， 106573.CrossRef
20. 易卜拉欣；李Y；KahngH。K.；金姆，年代；金姆，D。H.基于区块链的智慧城市发展停车共享服务。压缩。电气化。雕刻2022， 103， 108267.CrossRef
21. 易卜拉欣m.；JamilF.；李Y；金D。基于区块链的健身服务安全负载均衡任务调度方法。压缩。母亲康廷。2022， 71， 2599 - 2616.CrossRef
22. 王，H.；徐， C.；张先生， C.；徐，J。一个确保查询完整性的区块链系统。在2020年ACM SIGMOD国际数据管理会议会议记录上，波特兰，或，美国，2020年6月14-19日；pp。2693 - 2696.

23. 美国国际商用机器公司IBM区块链解决方案。在线可用： https://www。ibm.com/区块链/（已于2021年9月10日访问）。
24. BigchainDB有限公司。区块链数据库。在线可用： https://www。bigchaindb.com/whitepaper/ bigchaindb-whitepaper.pdf（已于2021年9月10日访问）。
25. 神示所甲骨文区块链平台。在线可用：https://文档。神示所云/区块链/云/what-are-advantages-oracle-blockchain-platform.Html（已于2021年9月10日访问）。
26. 舒斯特。SAP HANA区块链-一个技术介绍。在线可用：https://博客。sap.com/2018/06/13/sap-hanablockchain-technical-introduction/（已于2021年9月10日访问）。
27. 流体。区块链和Fluree的账簿。在线提供： https://docs.flur。ee/guides/1.0.0/architecture/blockchain(访问时间为2021年9月10日。
28. 蜂群群的书面形式。在线可用：https://文档。ethswarm.org/swarm-whitepaper.pdf（已于2021年9月10日访问）。
29. Gursoy，G.；布兰农，C。M.；格斯坦，M。使用以太坊区块链，通过智能合约来存储和查询药物基因组学数据。BMC Med.基因组。2020, 13, 74.CrossRef
30. Peer安全。区块链写作论文v3.0。在线可用： http://www。chainsql.net/PDF（已于2022年8月2日访问）。（中文）
31. 王，J.；王，H。一氧化物：扩展具有异步共识区域的区块链。第16届USENIX网络系统设计与实现研讨会（NSDI 19），波士顿，美国马萨诸塞州，2019年2月26-28日；USENIX协会：美国马萨诸塞州波士顿，2019年；页。95 - 112.
32. 张H.、刘J.、赵H.；王P.；加藤，N。基于区块链的车联网信任管理。IEEE跨。艾默格。顶部压缩。2021, 9, 1397 - 1409.CrossRef
33. 周，C.；陆H.；向Y.；吴J.；王，F。基于地理哈希的矢量地理数据显示方法。Int。J.Geo-Inf.2020, 9, 418.CrossRef
34. 陈X.；陈S.；徐T.；阴B.；彭J.；梅X.；李，H。基于生成对抗网络的半监督样式图生成方法。IEEE跨。地球科学。远程传感器。2021, 59, 4388 - 4406.CrossRef
35. 尼梅耶，G。土石灰。可在线获得： http://geohash。org/（已于2022年8月2日访问）。
36. 张先生， C.；朱，徐， C.；张先生， C.；谢里夫，K.；吴，H.；韦斯特曼，H。BSFP：区块链启用的智能停车场，具有公平、可靠性和隐私保护。IEEE跨。Veh。技术。2020, 69, 6578 - 6591.CrossRef
37. Szydlo，M。日志空间和时间中的默树遍历。在密码学的进展-欧洲隐窝，2004年；卡钦，卡梅尼施，J。L., Eds.；计算机科学课堂讲稿；施普林格：柏林/海德堡，德国，2004年；第3027卷，页。541 - 554.