区块链系统的数据存储与查询技术综述

王千阁 何 蒲 聂铁铮 申德荣 于 戈
(东北大学计算机科学与工程学院 沈阳 110169)

摘 要 目前,以比特币和以太坊为代表的区块链系统已经日趋成熟,区块链技术成为学术界与工业界的研究热点。然而,这些区块链系统在实际应用中因数据存储模式限制而普遍面临着查询功能简单、查询性能较低等严重问题。文中重点对区块链系统的数据存储与查询技术的研究进展进行综述与展望。首先,介绍当前流行区块链系统中使用的数据存储机制和查询处理策略。然后,详细介绍在现有区块链系统基础上扩展查询处理功能的两种方法,并从查询效率、写性能优化、存储空间占用、数据安全性和可用性 5 个方面对其进行对比和分析。最后,分析了未来区块链系统的查询技术发展趋势,探讨了其主要的研究方向。

关键词 区块链,数据库,存储管理,查询处理,索引结构
中图法分类号 TP315 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.12.002

Survey of Data Storage and Query Techniques in Blockchain Systems

WANG Qian-ge HE Pu NIE Tie-zheng SHEN De-rong YU Ge
(School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China)

Abstract At present, blockchain systems, represented by Bitcoin and Ethereum, are becoming more and more mature, and blockchain technology has become a hot topic in academic and industrial circles. However, in practical applications, these systems are generally faced with tough problems such as simple query function and low query performance due to the limitation of data storage scheme. This paper presented the survey and prospect of the research progress on data storage and query technology of blockchain systems. First, the data storage mechanism and query processing strategy used in current popular blockchain systems were introduced . Then, two methods of extending query processing functions on the existing system were introduced in details. The features of their query efficiency, write performance optimization, storage space occupancy, data security and availability were compared and analyzed in detail. Finally, the trend of the query technology development in the future block chain system was analyzed, and the main research direction was discussed and explored.

Keywords Blockchain, Database, Storage management, Query processing, Index structures

1 引言

近年来,随着比特币、以太坊等区块链系统的成功应用,区块链技术受到学术界与工业界的广泛关注。区块链系统因具有去中心化、防篡改、数据共享、隐私保护等诸多特性,被认为将会给传统的互联网应用和金融领域带来变革[1]。区块链技术是一种集成创新技术,它由多种已有的成熟技术整合而成,包括密码学、点对点网络、高性能数据存储和共识协议等。在工业界,大量的成熟区块链系统已经投入实际应用,更多的企业正在加大对区块链领域应用的投入。然而,学术界对区块链技术的研究仍然处于起步阶段,研究内容主要集中在系统性能分析[2]、前沿应用探索[3-4]、安全保密和共识机制[5-8]等方面,新架构[9]针对区块链底层数据管理系统和查询技术的研究则相对较少。在当前的实际应用中,区块链系统在查询处理方面已经暴露出许多不足。

1)查询效率低

大多数区块链系统底层的数据存储系统使用的是LevelDB[10],它是一种针对写密集型应用而设计的数据存储系统,以牺牲数据读性能为代价换取写性能的提高。然而,在实际应用中,区块链系统单位时间内的数据写入量并不大。例如,以太坊系统的交易写入量为每秒 7~10 笔左右[11],比特币系统目前的交易写入量为每秒 1 笔左右,LevelDB 的高速

到稿日期:2017-12-22 返修日期:2018-04-20 本文受国家自然科学基金课题(U1435216,61433008),中央高校基本科研业务费(N150408001-3,N150404013),辽宁省自然科学基金(2015020018)资助。

王千阁(1994-),男,博士生,主要研究方向为大数据存储与分布式计算,E-mail:wangqiange@stumail.neu.edu.cn;**何 蒲**(1992-),男,硕士,主要研究方向为云计算与区块链系统;**聂铁铮**(1980-),男,博士,副教授,主要研究方向为分布式数据库、大数据管理;**申德荣**(1964-),女,博士,教授,主要研究方向为分布式数据库、大数据集成;**于 戈**(1962-),男,博士,教授,CCF 会员,主要研究方向为数据库理论与技术、分布与并行式系统,E-mail:yuge@mail.neu.edu.cn(通信作者)。

写入优势无法体现出来。随着区块链系统中数据的增加和应用的扩展，往往需要处理频繁的查询，其底层存储系统的写性能过剩但是读性能不足，这成为了限制查询性能的主要瓶颈。

2）查询功能有限

区块链系统使用的数据存储系统大多是基于 Key-Value 模型的非结构化数据存储系统，如 LevelDB。这些系统仅支持基于 Key 值的插入与查询，并不支持复杂查询的关系操作。电子金融、电子商务等区块链应用领域对关系查询和分析型查询有着迫切的需求。因此，现有区块链系统的查询功能极为有限，难以满足实际应用的需求。

3）读写性能难以调整

由于区块链系统的应用领域不同，对读写性能的需求也不尽相同。例如，在电子商务支付系统中，单位时间产生的交易数量较大，需要频繁地执行写入操作，并不存在大规模的查询请求。而银行业的结算系统中，交易数量有限，对数据存储系统的写入性能要求不高。但由于需要处理复杂的审计流程和频繁的分析运算，用户需要执行大量的查询，对系统的读性能的要求较高。而现有区块链系统的性能难以满足对读写性能调整的要求。

4）数据存储缺乏灵活性

目前，区块链主要应用于电子金融与电子商务，区块链中保存的数据的字段数量少，结构相对固定，查询处理逻辑简单。而随着应用领域的扩大，应用的数据结构将更加复杂。现有系统无法增加和管理用户自定义字段，难以为新的应用提供有效支持。

以上问题成为了限制未来区块链系统应用的瓶颈。针对以上问题，本文做了以下工作：1）总结了现有流行区块链的底层数据存储系统和查询处理引擎设计；2）总结了现有区块链系统的数据查询技术，提出在现有流行区块链扩充中查询处理引擎的解决方案；3）对未来区块链的底层存储系统和查询技术的研究方向进行了展望。

本文第 2 节介绍现有区块链系统使用的数据存储技术；第 3 节介绍现有区块链系统对查询功能的支持；第 4 节讨论在现有区块链系统中增加查询处理引擎的方法；第 5 节分析区块链查询技术的难点并指出今后的研究方向；最后总结全文。

2　区块链系统的数据存储技术

目前，各种主流区块链系统的具体体系结构的设计不尽相同，但基本可以划分为三大模块：数据存储模块、共识协议模块和查询处理模块[12]。其中，数据存储模块在系统中主要承担区块链数据的持久化、数据的编解码与存取访问任务。底层使用的数据存储系统决定了数据存储模块的查询功能和读写性能。因此，本文首先介绍区块链系统使用的数据存储系统。

2.1　基于 LevelDB 的数据存储管理

目前，区块链系统的数据存储大多使用的是 LevelDB。LevelDB 是一种基于日志排序合并树（LSM-Tree）的单机 Key-value 数据存储系统。随着硬件性能和传感器性能的提高，以及大数据时代的到来，相比过去人们在单位时间内可以获得更多的数据，新一代数据库面对的首要挑战就是在单位时间内保存和处理更大体量的数据。传统的关系数据库系统的优化策略，主要是通过维护索引结构和保证数据的聚簇存储的方式来提高查询性能。这种策略需要在数据插入阶段执行多次随机磁盘读写来维护索引，以牺牲数据的写入速度为代价。而现代数据库的应用场景已经从读密集型转化为读/写并重。显然，传统数据管理系统难以应对单位时间内的大量数据写入。在大数据分析场景中，系统要处理的数据操作是成块数据的连续读写，传统数据库管理系统的查询优化策略并不能带来明显的性能提升。为了处理这一情况，谷歌开发了针对写操作优化的新一代数据存储系统 LevelDB。LevelDB 系统具有极高的写性能，这主要归功于 LSM-Tree 结构[13]。与传统的针对读优化的多路平衡查找树不同，这种结构的核心策略是：在数据写入阶段降低由于更新索引结构所产生的大量随机读写开销，在内存中完成固定大小的分片数据的排序，然后将其一并写入磁盘；而后在后台将小的有序文件逐步合并为大的有序文件进而提高查找效率。与之相对比，B+树在写入阶段需要在磁盘上多次随机读写，而 LevelDB 在写入阶段仅产生极少的内存排序开销，文件的排序在后台进行，而且该操作也仅仅带来顺序读写磁盘的开销。这种策略可以大幅提升数据的写入效率。下面介绍具体的数据存储过程。

1）写日志与内存排序

系统在内存中通过 skiplist 结构维护一个 Memtable，数据在 Memtable 中按照 Key 值有序存储。数据到达之后，首先写日志，然后将其插入到 Memtable 中。当 Memtable 达到一定大小后，将 Memtable 转化为 Imemtable 准备写入磁盘，而后系统产生一个新的 Memtable。

2）Imemtable 持久化与后台合并

系统在磁盘上将存储区划分为 N 级：level 0，level 1，…，level $n-1$。当 Memtable 达到一定大小后，将其转化为一个不可被写入的 Imemtable。随后，系统将其保存为一个 sstable 文件并持久化到 level 0 级存储区中。在 level 0 级存储区中，所有的 sstable 内的数据都是按照 Key 值有序排列的，但是各个 sstable 之间可能出现 Key 重叠。当 level 0 级存储区达到给定上限后，系统在后台将 level 0 级存储区内的所有 sstable 与 level 1 级存储区中保存的所有 sstable 按照 Key 值排序合并存储到 level 1 级存储区中。这一过程会将两个存储区中的数据排序合并，并删除其中的重复键值，具体原理此处不再赘述。同理，如果 level 1 级发生溢出，则合并处理到 level 2 级存储区，向下逐步进行相应操作，插入过程如图 1(a)所示。

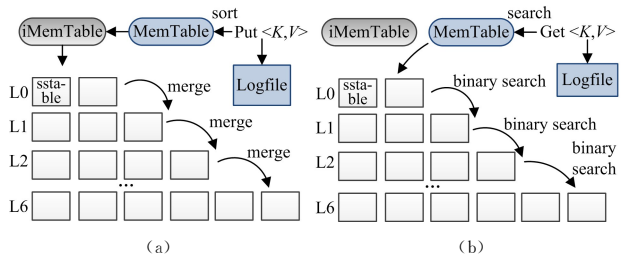

图 1　LevelDB 的写入过程与读取过程

Fig. 1　Writing and reading processes of LevelDB

除了 level 0 级存储区，level $m(m>1)$级存储区内部任意两 sstable 之间的 Key 值范围不会产生交叉。这种在逻辑上将存储区划分为 level 的方法有效保证了数据的写入效率，并且通过后台的维护保证了文件在各个 level 中有序，便于后续的数据查找。

2.2 区块链系统存储方案的比较

目前，绝大多数区块链系统使用了以 LevelDB 为代表的 Key-value 数据存储系统。比特币的早期版本使用了 Berkley DB[14]，2012 年，Berkley DB 系统停止维护，因此开发人员在后续更新中将 Berkley DB 更换为 LevelDB。以太坊的开发时间相对较晚，其底层使用 LevelDB 作为数据存储系统。超级账本在研发之初使用 LevelDB 作为数据存储系统[15]。随着版本的迭代，系统可以由不同的数据存储系统提供支持。在 fabric 0.6 版本及之前，系统底层使用 LevelDB 存储系统。随后，在 fabric1.0 版本的更新中系统给用户提供了 LevelDB 与 couchDB 两种存储系统作为选择，需要注意的是系统并未强制用户停用 LevelDB。Storj 是一种基于区块链的云存储系统[16]，其区块链数据的底层存储也使用了 LevelDB，由于使用 Javascript 语言开发，系统需要使用 LevelUP 接口作为系统与 LevelDB 之间的连接层。Filecoin 是另一种基于区块链的云存储系统[17]，系统部分继承了比特币的源码，同样使用LevelDB 作为底层存储。需要注意的是，Storj 与 Filecoin 中的区块链用于存储系统的元数据而非用户数据。BigchainDB 是一种支持区块链特性的分布式数据库系统，它使用 rethinkDB 和 mongoDB 两种数据库作为后端数据库，并在此基础上设计了丰富的查询语言。

严格意义上讲，区块链系统与基于区块链系统有着本质上的不同。在前 3 种系统中，区块链直接提供给用户使用，系统将用户数据保存在区块链上，并且提供相应的查询接口。底层的数据存储系统需要直接处理查询逻辑，因此性能需求较高。而在后 3 种系统中，区块链作为系统的功能模块，并不存储用户数据，也不向用户提供读写接口。对于 Filecoin 和 Storj 两种云存储系统，区块链用于存储文件的分片、存储位置等元数据。以 Storj 系统为例，系统将文件分片并存储在不同节点中，用户提供的物理存储区在逻辑上被划分为桶。每个文件的逻辑地址包括各个分片的桶号与桶内 ID，区块链则用于存储逻辑 ID 和实际物理位置的映射。显然，用户读取文件系统仅需访问区块链一次，这种应用场景对区块链系统的性能要求较低。BigchainDB 系统本质上是一个分布式数据库，其运行机制与传统区块链系统不同，也不具有现有区块链系统所具有的数据全复制特性，它将所有的查询交由后端数据库执行，区块链仅作为系统的一部分用于维护数据的安全性。各系统的数据存储特点如表 1 所列。

表 1 现有区块链系统的底层存储系统总结

Table 1 Summary of underlying storage systems adopted by current blockchain systems

	系统	使用的数据存储系统	区块链中存储的数据	对关系型数据的支持	对关系型查询的支持
区块链系统	Bitcoin	BerkleyDB/LevelDB	用户数据	N	N
	Ethereum	LevelDB	用户数据	N	N
	Hyperledger	LevelDB/couchDB	用户数据	N	N/Y
基于区块链的系统	Storj	LevelDB	元数据	—	N
	Filecoin	LevelDB	元数据	—	N
	BigchainDB	rethinkDB/mongoDB	元数据	—	Y

由此可见，在应用中区块链系统的底层数据存储系统的被访问次数有限，执行功能也相对单一。而当区块链作为用户应用的数据管理平台时，底层存储系统需要面对用户的频繁访问，这对数据存储系统的功能和性能均提出了较高的要求。因此，本文将进一步讨论区块链面向用户时，查询处理模块的设计与改进。

3 区块链系统的数据查询功能

3.1 LevelDB 的查询处理机制

LevelDB 中的文件是分层有序的，且大多数情况下会有部分数据保存在内存中。当一个查询请求到达时，首先需要检查结果是否保存在内存的 Memtable 中，然后再检索磁盘。在 level 0 级存储区中，由于 sstable 之间存在 Key 重复，因此需要在每一个 sstable 文件中进行二分查找。在 level $m(m>1)$级存储区中，由于 sstable 之间有序，因此首先定位查询 Key 所在的 sstable，然后在该 sstable 内执行二分查找，并最终返回结果，查询过程如图 1(b)所示。由分析可知，假设系统 level 0 级存储区中 sstable 的上限为 k，而系统存储区总计划分为 N 个 level，则需要进行 $N+k$ 次磁盘定位和 $N+k$ 次二分查找操作，时间消耗为$(N+k)(T+\log(n')+TR)$，其中 n'为单个 sstable 文件中的数据量，n 为总数据量，T 为磁盘定位时间，TR 为一次磁盘读写时间。相比之下，以 B+树为代表的读优化索引的时间消耗为 $T+\log(n)+TR$。可以看出 LevelDB 的数据读性能相对较低。

3.2 以太坊系统的查询功能

LevelDB 提供的查询功能较为简单，系统仅提供针对 Key 的查询、更新与写入等操作。在其他应用场景中，用户需要自行在 LevelDB 上层封装查询层与索引结构。目前，以 LevelDB 为基础的区块链都没有设计相应的查询层，因此仅支持简单的查询操作。由于各种区块链系统的查询功能基本相同，本文以以太坊为例进行分析。首先介绍以太坊系统链上的数据结构。

与所有区块链结构相同，以太坊系统的链上数据包括两部分结构，即区块体和交易数据字段。区块体数据主要由系统产生和维护，用于区块链数据的保护，区块体数据结构如表 2 所列。用户数据主要存储在交易记录字段中，本文只给出与查询相关字段的相关说明，如表 3 所列。

表 2　区块体记录中的主要字段

Table 2　Main fields of body record in blockchain

子结构	英文名称	作用说明	大小
区块 ID	number	区块编号	4 Byte
区块哈希值	hash	数字签名	20 Byte
父区块哈希值	parentHash	前序区块数字签名	20 Byte
MP-Tree 校验值	transactionroot	交易的检验值	20 Byte
交易列表	transactions	区块的交易列表	变长

表 3　区块交易记录中的主要字段

Table 3　Main fields of transaction record in blockchain

子结构	英文名称	作用说明	大小
交易哈希值	hash	交易的数字签名	4 Byte
发送方	from	发送方地址	20 Byte
交易方	to	接收方地址	20 Byte
金额	value	交易金额	20 Byte
数据字段	data	用于智能合约的代码存储与参数调用	变长

以太坊在将内存中的数据持久化到磁盘时，首先将用户提交的 JSON 格式数据通过递归长度前缀编码转化成字符串(Value)，然后计算 Value 的数字签名 Key，最后将$\langle Key_i, Value_i\rangle$对存储在 LevelDB 中。执行查询时，首先从查询语句中解析用户提交的查询请求中的 key；然后调用 LevelDB 的 Get 方法获得 Va-lue 字符串；随后将 Value 解码，并使用 Json 数据封装，将结果返回客户端。具体查询执行流程如图 2 所示。

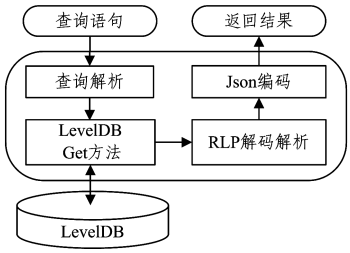

图 2　以太坊内部查询处理的执行流程

Fig. 2　Execution process of query processing in Ethereum

在以太坊系统支持的区块链上，数据查找操作包括 3 种方式：1)按照区块高度查找区块头；2)按照区块哈希值查找区块头；3)按照交易哈希值查找交易。除此之外，系统支持上述 3 种查询构成的复合查询。所有的查找操作均可归结为以 Key 值为查找谓词的查询，即：

$$\text{SearchByKey}(key)=\{Value_i \mid \langle Key_i, Value_i\rangle, Key=Key_i, i=1,\cdots,N\}$$

其中，N 为数据数量，$\langle Key_i, Value_i\rangle$为 LevelDB 中存储的键值对。在以太坊中，各个字段并不是直接串联地存储在 LevelDB 中，而是通过递归长度前缀编码(RLP)[18]进行压缩存储，且底层 LevelDB 不支持针对 Value 中任意字段的查找。由于设计之初未考虑支持分析型查询，因此没有在中间层设计索引。系统无法像传统数据库一样执行关系型查询，更无法执行 Top-K 查询、K-NN 查询等复杂分析型查询。

4　区块链系统的查询功能扩展技术

目前，关于区块链系统中查询处理模块设计的研究依然处于起步阶段，研究成果屈指可数。

Li 等提出了 etherQL 系统[19]，它是一种在区块链外层设计的查询层。该系统的主要思想是将区块链数据拷贝到外部数据库中，并借助外部数据库提供的功能接口设计查询层。其设计了区块链数据监听系统，将链上数据复制到 MongoDB 中，然后利用 MongoDB 执行分析查询操作。这种方法不需要修改原有的区块链系统，直接在区块链系统之上实现查询层即可，是一种简便可行的方案。

此外，已有研究工作对区块链数据的查询方法与数据存储系统的性能进行了分析。Dinn 等提出现有的区块链系统中执行引擎、共识协议和数据存储系统三大主要模块间存在代码耦合问题[20]，需要在未来的研究中对系统解耦并分别优化。针对数据存储系统，文献[12]提出可以使用 Ustore 等支持丰富查询语义的数据存储系统替换 LevelDB。Ustore[21]是新加坡国立大学提出的一种支持丰富查询语义的数据存储系统。相对于关系型数据库，它没有复杂的事务处理策略，设计更为精巧，且具有较高的运行效率。而相比以 LevelDB 为代表的 Key-Value 数据存储系统，它可以提供更加丰富的查询功能。蔡维德等[22]提出区块链系统的数据存储层可以采用多种存储系统，包括 Hbase，MySql 等成熟数据库系统。但是目前并没有实际的区块链应用采用关系型数据库或分布式存储系统。传统索引结构(例如 B+树和 LSM-tree)在不同数据插入顺序下会产生截然不同的结构，这给分叉语义和多版本数据的维护带来了极大的困难。针对该问题，Wang 等[23]提出了一种新的索引技术 SIRI，并依据该索引建立了 POS-tree 结构，实现了数据的分布式版本控制。这种树型结构类似于 B+树与 Merkle 树的混合结构，避免了原有结构在不同插入顺序下的一致性维护成本。

本节通过对现有区块链系统进行深入分析与研究，提出了查询功能扩展方案，并对方案进行了性能分析和比较。

4.1　基于现有系统的查询功能扩展方法

目前，多种区块链系统已经被广泛使用，对其底层数据存储系统进行彻底的更新需要面临高昂的成本和极大的阻力。因此，切实可行的解决方案是在现有的成熟系统中增加查询处理引擎。首先，查询处理模块应该尽可能减少对系统原有设计的修改；其次，系统应该具有尽可能高的查询性能。因此，我们认为在现有系统上扩充查询层有两种方法：基于外部数据库的外联数据库方法和基于内部辅助索引的内置索引方法。

1)外联数据库方法(Connecting outside database)。这种方法的思想主要来自 etherQL 系统，该系统包括 3 个主要部分：数据监听与解析模块、外部数据库和查询 API。在区块链系统运行阶段，数据监听与解析模块通过区块链提供的 API 监听区块数据并导入到外部数据库中。查询主要借助外部数据库提供的查询接口实现，查询层结构如图 3 所示。与 etherQL 方法略有不同，数据监听模块通过以太坊提供的 API 读取数据而非将代码接入系统内部。外联数据库方法的主要优势在于，将外部数据库作为媒介，以简单的实现方式换取了较高的查询效率和丰富的查询功能。

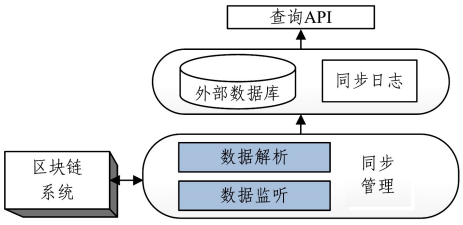

图 3 外联数据库方法结构图

Fig. 3 Architecture of connecting outside database method

2)内置索引方法(Built-in index)。外联数据库方法的思想是在系统外部设计查询层,而内置索引方法是一种内建于系统之中的查询层。这种方法的索引设计思路参考了 MyIsam 引擎的索引组织方式。MyIsam 是 MySQL 数据库中一种常用的存储引擎。这种存储引擎中的索引分为两种:主键索引与辅助索引。主键索引指向的是数据的物理地址,而辅助索引指向的是数据的主键。这种索引组织方式可以大大降低维护成本。不难看出,LSM-tree 结构即可看作一个主键索引。因此,内置索引方法以 LevelDB 作为主键索引,在数据存储模块中针对不同字段建立辅助索引,并通过索引结构设计查询层。内置索引法的结构如图 4 所示。内置索引方法的查询过程分为两个阶段:第一阶段,将查询命令发送到查询模块,通过辅助索引确定结果的 Key(主键)集合;第二阶段,使用 Key 值在主键索引(LevelDB)上查找出结果 Value,并返回给客户端。

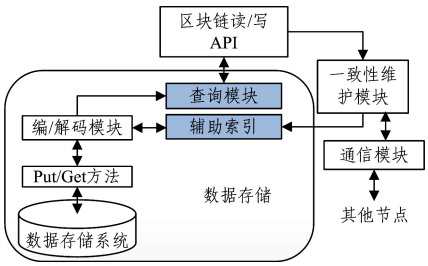

图 4 内置索引法结构图

Fig. 4 Architecture of built-in index method

4.2 两种扩展方法的比较

两种方法由于设计思路不同,因此具有不同的特性。下面从 5 个方面对两种方法进行对比。

1)查询性能。以 select 查询为例。对于外联数据库方法,查询过程在外部数据库中执行,每次查询只需要查找一次索引即可获得结果,时间消耗为 $O\log((N)+Ts(kN))$。其中 $Ts(k)$为系统读取耗时,k 为选择度。可以看出,数据存储系统的读写瓶颈并不会影响查询性能。对于内置索引方法,系统首先通过辅助索引获取到结果的 key 值,然后在主键索引(LevelDB)上执行查找返回结果。其时间消耗为 $O(\log(N)+kT(N))$,其中 $T(N)$为 LevelDB 执行一次随机查找的耗时。查询层和辅助索引只能加快第一阶段结果 key 值的查找速度,而第二阶段结果的返回速度仍会被 LevelDB 影响。

2)额外空间消耗。对于外联数据库方法,数据库需要保存区块链上完整的数据副本并为之建立相应的索引结构,这需要占据大量的磁盘空间,其空间消耗为 $O(dN+cN\log(N))$。对于内置索引方法,查询层只需要保存特定字段的索引结构,因此占用空间相对较小,其空间消耗为 $O(cN\log(N))$。理论分析尚且如此,在实际应用场景中外联数据库方法的空间浪费更加明显,在以太坊的交易记录中,用于智能合约存储与调用的 Data 字段占据的空间往往远大于其他字段空间的总和,然而实际使用中 Data 字段不可能用作查询谓词,因此也需要保存到外部数据库中,故外联数据库方法的空间浪费的缺点被进一步放大。

3)附加数据的一致性维护与容错。无论是外联数据库方法还是内置索引方法,都需要引入附加的数据存储空间,因此必须考虑这部分数据的一致性与安全性。下面分两种情况给出解决方案:系统崩溃与外部攻击。

①外联数据库的解决方案。系统崩溃恢复方法:外部数据库的数据更新是异步完成的。假如单位时间内数据的写入非常频繁,则可能造成短时间内更新进度滞后(异步更新策略可以保证副本的最终一致性),如果此时系统发生崩溃且未保存更新的位置,再次启动后区块链状态与外部数据库状态可能不一致,将造成系统难以恢复。因此在后续的改进中系统可以维护一个进度日志,以保证重启后的恢复操作。外部篡改防护策略:外联数据库方法的主要问题是防篡改能力,区块链可以保护链上数据不被篡改,但是无法保护外部数据库,如果外部数据库遭受攻击,使用者很难及时发现错误。而大规模的数据校验将耗费大量系统资源,频繁地执行会减少外部数据库可供使用的窗口时间。

②内置索引方法的解决方案。系统崩溃恢复方法:辅助索引是内建的,链上数据与索引的更新是同步的,即当一条数据已经写入索引并且写入到 LevelDB 后才允许进行下一步操作,且 LevelDB 内部自带日志系统,因此,系统崩溃重启之后,区块链状态与索引状态是相同的,一致性得以保证。外部攻击防护策略:与外联数据库方法不同,辅助索引每执行一次插入操作都需要经过一致性模块的检验,其他情况不允许对索引执行写操作,而如果对索引文件进行暴力破坏,则可以通过网络内的其他节点进行恢复。这与区块链使用的写入策略是一致的。因此,内置索引方法的抗外部攻击能力与恢复能力较强。内置索引方法第一、二阶段的平均执行时间如图 5 所示。

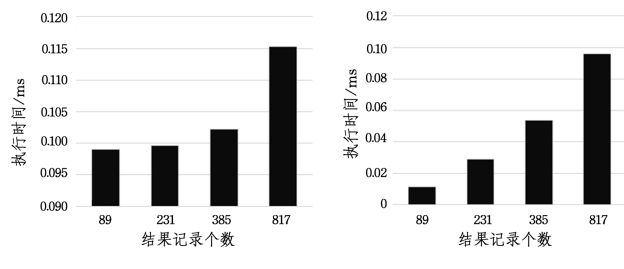

图 5 内置索引方法第一、二阶段的平均执行时间

Fig. 5 Average execution time of phase 1 and phase 2 of built-in index method

4)写性能的退化。外联数据库方法中,外部数据与区块链数据的更新是异步的,系统的写入性能并未受到影响。对于内置索引方法,需要在数据写 LevelDB 之前更新索引结构,这带来了一定的额外开销。因此,内置索引方法的写性能相对于原有系统有一定的下降。

5）可扩展性。对于外联数据库方法，查询类型的扩充可以在外部数据库中借助 API 完成，可操作性好。而内置索引方法中，查询处理模块设计在系统之中，因此扩展查询功能需要改写系统查询层代码，技术相对较为复杂。

通过以上 5 个方面的对比可知，外联数据库方法具有实现简单、可扩展性好、查询性能高的特点，但是占据的存储空间较大，外部数据不受区块链的保护。内置索引方法的实现较为复杂，查询效率难以与外联数据库方法相比，但由于其索引结构是内建的而且不需要保存庞大的副本，因此占用的额外存储空间较少，附加存储的安全性也相对较好。我们对两种方法进行定性分析，如图 6 所示，靠近外圈表示相对优秀。两种方法的优缺点各有不同，后续的优化方向也各有不同。

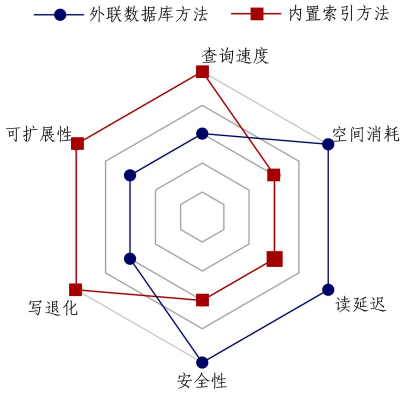

图 6　两种方法的性能对比

Fig. 6　Performance comparison of two methods

外联数据库方法的核心瓶颈在于外部数据的抗攻击能力较差，因此系统需要一种高效的大规模数据校验算法，以减少外部副本与区块链数据的不一致窗口时间，进而保证查询的正确性。内置索引方法的查询数据源仍然来自 LevelDB，因此 LevelDB 随机读操作依旧是限制查询效率的瓶颈，且受限于 LevelDB 的实现机制，难以进行深入的优化。

4.3　性能分析

针对内置索引方法，本文进行了两个阶段的对比实验。数据源包含 10 万条记录，实验以范围查询为测试查询用例，分析两个阶段的平均耗时。实验结果如图 5(a)和图 5(b)所示，对比可知阶段一耗时相对较少，阶段二耗时占据查询过程总耗时的 99%以上。而且随着结果数量增加，阶段二耗时线性增长。因此，内置索引方法更适合条件严格且返回结果数量较少的查询。

需要注意，由于区块链具有数据副本全复制的特性，本文默认主机已经保存了完整的数据副本。在重量级节点中，查询均基于单机节点执行，而不需要产生网络通信。轻量级节点需要将查询提交到附近的重量级节点中执行。这个过程会产生查询代理的可靠性问题，然而目前尚无相关工作对此展开研究。目前关于区块链分片存储的研究已经展开[7,24-26]，如何在分片区块链系统的基础上执行查询，且保证查询的正确性和效率将是未来的一个重点研究方向。

5　研究方向展望

为现有区块链系统设计查询处理模块的主要目的是，在不对系统进行大规模修改的条件下扩充查询功能，因此查询处理的性能与安全性还存在较大的局限。要设计区块链系统的查询处理模块，还需针对许多工作展开深入研究。结合已有的部分研究成果，我们认为将来区块链的查询功能应该主要由底层数据存储系统完成，而不是在数据存储系统和其他模块之间进行独立设计。查询功能在数据存储系统内部实现有以下优点：1）可以从数据存储模型开始进行有针对性的优化，使数据存储系统匹配区块链特性；2）可以从内部有效地控制查询并发，避免外部查询层中读写锁与同步机制的繁琐设计；3）避免了外部附加结构造成的数据不一致问题。目前针对单机高性能数据存储系统的研究已经非常成熟，这里仅列举部分具有代表性的成果[21,27-36]。目前数据存储系统需要解决的问题包括以下几个方面。

1）查询多样化

现有区块链底层数据存储系统支持的查询功能普遍简单，其性能不足以支持内置索引方法的查询效率，因此无法适应现有应用。未来的区块链底层数据存储系统需要具有丰富的查询功能和良好的可扩展性。

2）数据读写速度的平衡

区块链在未来可能的应用场景包括读密集型、写密集型与读/写型应用。目前系统设计偏向于写密集，然而实际应用场景正在向读密集型转化。未来的区块链应用需要平衡数据读写速度，设计合理的查询层，增强系统的可伸缩性。

3）数据类型的多样性

目前，区块链系统只考虑了区块链中区块记录与交易记录中各个字段的查询与分析。未来需要将区块链上的一些半结构化数据纳入到查询中，例如智能合约的输入参数、运行结果以及用户自定义数据等。这要求底层数据存储系统能够支持半结构化数据的保存与管理。

4）适应区块链的新技术

区块链技术将会有更多的革命性变革，比如可编辑区块链技术、区块链分片技术等。未来的查询数据存储系统需要具有良好的可扩展性和前瞻性，即数据存储系统能够适应新技术的演化，而不需要大幅修改。

底层数据存储系统的特性很大程度上决定了区块链系统的查询功能与性能的上限。因此，未来区块链数据存储模块研究工作的重点在于：1）将数据存储模块从系统中解耦，以进行有针对性的优化；2）优化底层数据存储系统的功能与性能，针对半结构化的用户数据设计高效的存储与查询策略；3）为了支持区块链分片和可编辑区块链等新技术，引入分布式系统的设计原则，在底层存储系统上层设计辅助查询模块和一致性管理模块，以有效地保证查询处理的执行效率和有效性。

结束语　区块链技术给计算机与金融行业的发展带来了新机遇，区块链系统也逐渐成为研究热点。本文结合区块链在实际应用中暴露出来的数据存储与查询处理方面的问题，讨论了未来的应用需求与现有的可行解决方案，最后对区块链查询技术的发展趋势与研究方向进行了总结与展望。随着区块链技术在实际应用领域的发展，会有很多问题值得研究人员去深入研究和探讨。

参 考 文 献

[1] HE P,YU G,ZHANG Y F,et al. Survey on blockchain technology and its application prospect [J]. Computer Science,2017,44(4):1-7. (in Chinese)
何蒲,于戈,张岩峰,等. 区块链技术与应用前瞻综述[J]. 计算机科学,2017,44(4):1-7.

[2] PASS R,SEEMAN L,SHELAT A. Analysis of the Blockchain protocol in asynchronous networks[C]// International Conference on the Theory and Applications of Cryptographic Techniques. 2017:643-673.

[3] ALI M,NELSON J,SHEA R,et al. Block stack:a global naming and storage system secured by block chains[C]// USENIX Annual Technical Conference. 2016:181-194.

[4] ATENIESE G,MAGRI B,VENTURI D,et al. Redactable blockchain-or-rewriting history in bitcoin and friends[C]// European Symposium on Security and Privacy. 2017.

[5] KOSBA A,MILLER A,SHI E,et al. Hawk:the blockchain model of cryptography and privacy-preserving smart contracts [C]// Security and Privacy. 2016:839-858.

[6] HALPIN H,PIEKARSKA M. Introduction to security and privacy on the blockchain[C]// European Symposium on Security and Privacy Workshops. 2017.

[7] LUU L,NARAYANAN V,ZHENG C,et al. A secure sharding protocol for open blockchains [C]// ACM SIGSAC Conference on Computer and Communications Security. 2016:17-30.

[8] EYAL I,GENCER A E,RENESSE R V. Bitcoin-NG:a scalable blockchain protocol[C] // Usenix Conference on Networked Systems Design and Implementation. 2016:45-59.

[9] JIN H,DAI X,XIAO J. Towards a Novel Architecture for Enabling Interoperability amongst Multiple Blockchains[C]// International Conference on Distributed Computing Systems. IEEE Computer Society,2018:1203-1211.

[10] LevelDB[OL]. http://LevelDB. org.

[11] Etherchain[EB/OL]. https://www. etherchain. org.

[12] DINH T T A,WANG J,CHEN G,et al. BLOC-KBENCH:a framework for analyzing private blockchains[C]// International Conference on Management of Data. 2017:1085-1100.

[13] O'NEIL P,CHENG E,GAWLICK D,et al. The log-structured merge-tree (LSM-tree)[J]. Acta-Informatica,1996,33(4):351-385.

[14] Wikepedia. Berkeley_DB[EB/OL]. https://en. wikipedia. org / wiki/Berkeley_DB.

[15] Hyperledger[OL]. https://www. hyperledger. org.

[16] SHAWN W,TOME B,JOSH B,et al. Storj A peer-to-peer cloud storage network[EB/OL]. https://storj. io/storj. pdf. 2016.

[17] Protocol Labs. Filecoin:A decentralized storage network [EB / OL]. http://www. filecoin. io/fi-lecoin. pdf. 2017.

[18] Wikipedia. RLP[EB/OL]. https://github. com/ethereum/wiki/wiki/RLP.

[19] LI Y,ZHENG K,YAN Y,et al. EtherQL:a query layer for blockchain system[C]// Data-base Systems for Advanced Applications. 2017:556-567.

[20] DINH T T A,LIU R,ZHANG M,et al. Untangling blockchain: a data processing view of blockchain systems[DB/OL]. https://arxiv. org/abs/1708. 05665.

[21] DINH A,WANG J,WANG S,et al. UStore:a distributed storage with rich semantics [DB/OL]. https://arxiv. org/abs/1702. 02799.

[22] CAI W D,YU L,WANG R,et al. Blockchain application development techniques[J]. Journal of Software,2017,28(6):1474-1487. (in Chinese)
蔡维德,郁莲,王荣,等. 基于区块链的应用系统开发方法研究[J]. 软件学报,2017,28(6):1474-1487.

[23] WANG S,DINH T T A,LIN Q,et al. ForkBase:An Efficient Storage Engine for Blockchain and Forkable Applications[J]. PVLDB,2018,11(10):1137-1150.

[24] KARLSSON K,JIANG W,WICKER S,et al. Vegvisir:A Partition-Tolerant Blockchain for the Internet-of-Things[C]// International Conference on Distributed Computing Systems. IEEE Computer Society,2018:1150-1158.

[25] JIA D,XIN J,WANG Z,et al. ElasticChain:Support Very Large Blockchain by Reducing Data Redundancy[C]// Asia-Pacific Web. Springer,Cham,2018:440-454.

[26] LIND,JOSHUA,NAOR O,et al. Teechain:Reducing Storage Costs on the Blockchain With Offline Payment Channels[C]// SYSTOR. 2018:125-125.

[27] WU X,XU Y,SHAO Z,et al. LSM-trie:an LSM-tree-based ultra-large key-value store for small data[C]// USENIX Annual Technical Conference. 2015.

[28] AGARWAL R,KHANDELWAL A,STOICA I. Succinct :Enabling Queries on Compressed Data[C]// USENIX Conference on Networked Systems Design and Implementation. 2015: 337 - 350.

[29] WANG P,SUN G,JIANG S,et al. An efficient design and implementation of LSM-tree based key-value store on open-channel SSD[C]// Eurosys. 2014:1-14.

[30] AHN J S,SEO C,MAYURAM R,et al. Forest DB:a fast key-value storage system for variable-length string keys[J]. IEEE Transactions on Computers,2016,65(3):902-915.

[31] LU L,PILLAI T S,ARPACI-DUSSEAU A C,et al. WiscKey:separating keys from values in SSD-conscious storage[C]// USENIX Conference on File and Storage Technologies. 2016:133-148.

[32] NAKAMOTO S. Bitcoin:a peer-to-peer electro -nic cash system [EB/OL]. https://bitcoin. or-g/bitcoin. pdf.

[33] BUTREN V. Ethereum:A next generation smart contractand decentralized application platform [EB/OL]. https://github. com/ethereum/wiki/wiki/White-paper,2013.

[34] TRENT M,RODOLPHE M,ANDREAS M,et al. BigchainDB:A Scalable Blockchain Data-base[EB/OL]. https://www. bigchaindb. com/w hitepaper/bigchaindb-whitepaper. pdf. 2016.

[35] SEARS R,RAMAKRISHNAN R. bLSM:a general purpose log structured merge tree[C]// ACM International Conference on Management of Data. 2012:217-228.

[36] LIM H,FAN B,ANDERSEN D G,et al. SILT:a memory efficient,high performance key-value store[C]// ACM Symposium on Operating Systems Principles. 2011:1-13.