软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006091]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

区块链的数据管理技术综述*

张志威[1], 王国仁[1], 徐建良[2], 杜小勇[3]

[1](北京理工大学计算机学院,北京,中国)

[2](香港浸会大学计算机系,香港,中国)

[3](中国人民大学信息学院,北京,中国)

通讯作者: 王国仁, E-mail: wanggrbit@126.com

摘 要: 最近几年,随着加密货币和去中心化应用的流行,区块链技术受到了各行业极大的关注.从数据管理的角度,区块链可以视作是在一个分布式环境下众多不可信节点共同维护且不可篡改的账本.由于节点间相互不可信,区块链通过共识协议,确保数据存储的一致性,实现去中心化的数据管理.针对区块链的安全性以及共识协议,已有诸多工作进行全面的分析.本文将从数据管理的角度,分析区块链技术与传统数据库下数据管理技术的异同.分布式数据管理的研究已经持续数十年,涵盖了数据存储模式、事务处理机制、查询执行与验证、系统可扩展性等诸多方面,并已有诸多技术广泛应用于实际的分布式数据库中.该类工作往往假定存在中心可信节点或者节点只可能发生崩溃而不存在恶意攻击.然而,在区块链环境中,系统设计需考虑不可信节点可能的攻击行为以及拜占庭容错.这给数据管理带来了新的问题与挑战.因此,本文将梳理并分析国内外有关区块链数据管理的文献,并展望未来的研究方向.

关键词: 区块链;数据管理;数据存储;事务执行;查询处理;

Survey on Data Management in Blockchain Systems

ZHANG Zhi-Wei[1], WANG Guo-Ren[1], XU Jian-Liang[2], DU Xiao-Yong[3]

[1](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

[2](Department of Computer Science, Hong Kong Baptist University, Hong Kong, China)

[3](Information School, Renmin University of China, Beijing 100081, China)

Abstract: Blockchain technologies have gained more and more attention during the last few years. In general, blockchains are distributed ledgers in which the users do not fully trust each other. Embedded with consensus protocols and security mechanism, blockchain systems achieve several properties, such as immutability, and all the users agree on all the data records and histories of transactions. From the perspective of data management, blockchain is a distributed database, in which nodes agree with the orders of executions of all the transactions. Many works have been done to survey about the security and concensus problems for blockchains. In this paper, we aim to survey and analyze the techniques about data management for the blockchain systems. In traditional databases, it assumes that the nodes in the distributed database are trusted, and only the crash failure needs to be considered. On the other hand, as the blockchains consider the malicious nodes, it needs to consider Byzantine fault tolerance. These have brought new problems and challenges to the blockchains. Since blockchains and databases have similar architecture, many works have been done to translate the techniques from distributed databases to blockchains. Considering this, in this paper, we survey the techniques for the data management in blockchains. We focus on four aspects of management, including storage, transaction management, query processing and blockchain scalability. We compare the differences and analyze the benefits of the techniques in these areas for blockchains.

Keywords: blockchain; data management; data storage; transaction execution; query processing;

最近几年,由于加密货币以及去中心化应用的兴起,区块链技术在工业界与学术界得到了极大的关注,并产生了巨大的影响.加密货币中的比特币[1]、以太坊[2]等已被允许在新加坡、加拿大等国家进行支付.从数据处理的角度,区块链是一种由网络中一组节点维护、只可添加的链式数据结构.在区块链系统中,数据与事务以区块为最小单位进行存储.区块与区块之间以链表的方式进行连接,且新的区块只能在链表末尾添加.因此,区块链中每个节点均按照区块被添加的顺序来存储区块内的数据内容.在这种情况下,所有的区块形成了一个存在全局顺序的链表.区块链因此也常被称作账本.与此同时,区块链系统可在不可信的网络中支持数据的一致性以及不可篡改等特点.为了在不可信的网络中保证数据不可篡改,区块链将每个区块及前一区块的哈希值存储在区块中,因此网络中的节点可通过哈希值,验证自身数据以及前一区块的数据是否与相应的哈希值一致.同时,考虑到不可信网络中恶意节点的恶意攻击行为,区块链系统要求可以支持拜占庭容错,即当网络中存在少量恶意节点时,依然可以确保网络中数据的一致性.由于以上特性,整个区块链网络构成了一个去中心化且不可篡改的一致数据存储系统.

* 基金项目: 国家自然科学基金(61602395, 61732014);香港研究资助局项目(12200819);广东省科技专项资金项目 SDZX2019042

Foundation item: National Natural Science Foundation of China (61602395, 61732014); Hong Kong General Research Fund(12200819); Guangdong Key Science and Technology Project (SDZX2019042)

收稿时间: 2019-11-25; 修改时间: 2020-03-08, 2020-04-29; 采用时间: 2020-05-25; jos 在线出版时间: 2020-06-08

区块链系统的去中心化特点使其与传统的分布式数据库在技术,应用等方面有很大不同.传统的分布式数据库往往假定存在中心节点,且节点不存在恶意行为,因此大部分分布式数据库只需要考虑崩溃性故障.而区块链中节点之间是不可信的,即节点可能恶意地发送消息或执行计算,因而区块链需考虑拜占庭容错.诸多企业,例如 IBM[3]、Oracle[4]、SAP[5]、华为[6]等均建立了自身的区块链系统.随着区块链技术的推广以及对智能合约的支持,区块链技术也被进一步应用于多个领域,包括物联网[7, 117]、医疗健康[8-9]、专利保护[10]、政府监管[11]、资产管理[12]等.与此同时,越来越多的工作对区块链各个方面进行了优化,包括系统模型[13-14]、共识算法[15-16]、数据安全[17-20]、数据存储[21]以及性能评价基准[22]等等.

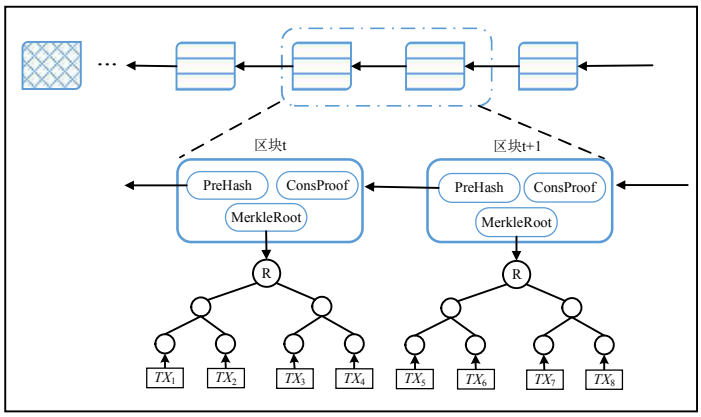

Fig. 1 The structure of the blockchain.

图 1 区块链中的数据结构

从数据管理的角度,区块链是一个在分布式环境下存储大量存在全序关系的数据记录系统.数据以及对数据的操作均存储在区块内并以区块的粒度进行管理.一个典型的区块结构如图 1 所示,包括前一区块哈希值(PreBkHash)、共识验证字段(ConsProof)以及区块内事务的哈希值(MerkleRoot).其中节点可利用共识验证字段验证其存储的区块是否满足共识条件并确保网络中少量节点的恶意行为不会影响区块链的一致性,从而使其支持拜占庭容错.在分布式数据库系统中,由于其假定节点不存在恶意攻击行为,因此分布式数据库中节点共识只需考虑节点崩溃的情况.典型的共识算法有 Paxos[23]、Raft[24]等.这类共识算法已经被应用于全球化的分布式数据库中,并已经取得了很好得性能.而在区块链中,为了支持拜占庭容错,区块链中典型的共识机制包括工作量证明 PoW[25]、实用拜占庭容错 PBFT[26]等.关于区块结构以及共识算法将在“区块链的核心构成”部分介绍.

虽然与分布式数据库相比,区块链可以支持在不信任网络环境中的去中心化数据管理,但是区块链系统在数据管理方面仍然面临着诸多问题.其核心问题包括数据存储、事务处理性能、查询处理优化等方面.例如,现阶段区块链的每秒可执行事务数(TPS)远小于传统的分布式数据库,也很难满足实际公司级别的交易量需求.最广泛使用的电子货币比特币支持平均每秒 7 笔交易,而 visa 等信用卡公司每秒需执行 4000 笔左右的交易[27-28].具体而言,区块链系统中的数据管理与现有分布式数据库有如下方面的不同:

- 数据存储方面:在区块链系统中,数据以区块作为基本存储单位.为了方便数据的存取,以太坊[2]等利用 LevelDB 这一基于 Key-Value 结构的数据库存取数据,而部分区块链则选择利用文件系统或关系型数据库进行存储.与传统的分布式数据库或其他分布式存储系统相比,由于区块链网络中的节点均存储数据的备份,导致数据的存储在区块链框架下产生新的挑战.首先,直接将所有数据存储于区块链上将导致区块链上的数据规模极大.在一般情况下,系统中的节点均需要存储所有历史和新添加的数据,因而导致整个系统的存储开销极大.这将使得部分节点可能由于存储空间的限制,不能加入到区块链的网络中.其次,由于区块链中存储了大量的历史数据,在这种情况下,一个数据在区块链中可以存在多个历史版本.现有的诸多操作需要基于数据的不同版本进行,因此数据存储需支持多版本的存储、查询等操作,且可以在各个版本上进行独立的数据添加更改等操作.
- 事务执行方面:在区块链中,智能合约的一次执行相当于数据库中的一个事务.在传统的数据库中,事务处理需满足 ACID 特性,包括原子性(atomicity)、一致性(consistency)、隔离性(isolation)与持久性(durability).在区块链系统中,其事务的处理同样需满足以上特性.为了提高系统的吞吐量,部分区块链的事务处理同样会采用并发机制.但是,区块链系统的并发控制与分布式数据库的并发控制相比有诸多不同.首先,区块链系统中的事务的提交(Commit)是以区块为单位,而不是以每个事务为单位.一个区块内将包含多个事务.这将导致针对事务的并发控制需要考虑区块的提交顺序.当一个事务与其他事务并发执行却没有在同一区块中提交将不会明显提高系统的执行效率.其次,部分区块链中的事务处理流程与数据库中的事务不同.例如 Hyperledger Fabric 中,事务的处理包含模拟、验证等过程.当其共识协议达成之后,每个节点才会执行区块内的事务.第三,分布式数据库往往基于 master-slave[32-33]或者

master-master 模式[34-35].其中 master 节点是可信任节点.由于区块链运行在不可信环境,事务的处理过程需要考虑区块链系统所采用的共识协议影响.以上不同使得分布式数据库的并发控制无法直接应用于区块链系统.

- 查询处理方面:由于区块链应用于不可信的环境,其查询处理的过程可归类为一般查询处理和可信查询处理.一般查询主要针对的是溯源查询等.针对可信查询问题,传统的针对查询结果的验证技术需要数据拥有者产生一个利用私钥生成的签名.后续会利用该签名进行验证.而在区块链系统中,不存在绝对的数据拥有者.网络中的多个节点(例如工作量证明共识协议下的挖矿者)均有机会添加区块到区块链中.这些有机会添加区块到区块链的节点不一定是数据本身的拥有者,进而不可能拥有相应的私钥以及签名.其次,传统的数据库签名方法针对不同的查询产生不同签名,并且均存储在数据库中.而在区块链系统中,由于区块链系统不可篡改的特点,签名一旦写入区块中就不可以更改.因而,当应对不同的查询时,写入的签名需要满足不同查询的验证条件.最后,基于传统数据库生成的签名往往基于的是静态数据库.区块链是一个动态,且对数据格式、大小无限制的系统.因而,可验证的签名需满足可验证动态添加的数据的要求.
- 可扩展性方面:为了提高分布式系统的可扩展性,传统的数据库可采用分片机制以提高性能.分片技术会将网络中的节点分割为多个独立的片.每个片内包含多个节点.片与片的数据一般存在一定的共有数据,从而可以保证当部分节点失效时,依然可以恢复全部数据.而在区块链系统中,由于节点的不可信,需要利用共识机制确保节点执行的事务等的一致性并能抵御恶意节点的恶意攻击.因此,区块链的分片技术要确保某一分片内的恶意节点不能控制该分片,使其恶意攻击行为成功.其次,为了将分片机制应用到区块链系统中,要确保分片环境下的共识机制的开销远小于分片所带来的性能的提升.最后,一般的分布式数据库存在 master 节点,来分配及协调事务的处理.而在区块链系统中,直接利用 master 节点来分配事务需要考虑到 master 节点可能是恶意节点的情况.

本文将从以上几个不同的数据管理方面问题,分析区块链与现有技术的异同.我们将基于现有的数据管理技术,尤其是分布式数据库管理技术,来梳理并分析现有的区块链环境下数据管理问题.现有针对区块链的工作从安全与可信性[109, 115]、系统架构[110]、智能合约技术[112, 114]、访问控制[113]等方面进行了综述.同时,[110]主要从区块链与数据库外联结合的角度,综述了在区块链中基于数据库的存储与查询的研究工作.与之前工作不同,本文将从数据管理的角度,聚焦区块链中的数据存储、事务处理、查询处理以及系统性能的方面的研究工作.

1 区块链的核心构成

在一个区块链网络中,包含互相不信任的多个节点.节点间无法确认某个节点是否为恶意节点.区块链网络假定某些节点可能产生恶意攻击,但是网络中大多数节点是诚实节点.整个区块链系统的节点一般可分为全节点、矿工节点和轻节点.全节点同步区块链上的所有数据,包括各个区块的头部哈希值、事务列表等.轻节点一般是用户的客户端.轻节点不存储区块链上的全部数据而往往只存储区块头部的哈希值.轻节点有时会提交针对区块链上数据的查询等,并利用返回的结果以及自身存储的哈希值对结果进行验证.矿工节点是一种特殊的全节点.该节点不但存储区块链的全部数据,同时参与区块链的共识过程.在一些区块链系统中,矿工节点与全节点是相同的节点.在这一部分,我们将介绍区块链的核心构成,包括区块结构与事务、共识机制、UTXO 与 Account 模型、智能合约以及区块链的类型.

1.1 区块结构与事务

区块链网络节点存储整个区块链系统的数据以及事务的日志.数据以区块的方式进行存储并链接.节点中一个区块的数据如图 1 所示.后一个区块均包含前一个区块的哈希值(PreBkHash 表示).除此之外,区块内还包含共识验证字段(ConsProof)、默克尔哈希树根的哈希值(MerkleRoot)以及事务日志所构成的默克尔哈希树.默克尔哈希树是一种二叉树[36],其叶子节点存储事务,而非叶子节点则存储其子节点内容的哈希值.默克尔哈希树的一个特点是,对于数据的任何变动,会影响从叶子节点到根节点路径上所有节点的哈希值.因此,所有针对数据的修改都会从叶子节点向上传递到根节点.因此利用该性质可验证数据是否被篡改.由于区块链以链式方式存储数据并执行事务,且区块链只允许在现有区块链的尾部添加新的区块,该性质使得区块链可以记录所有对数据进行更改的操作,因而其也被称为分布式日志或分布式账本.区块链中的一个事务对应于一次智能合约的完整执行.区块链的事务也同样要求满足数据库中事务的原子性、一致性、隔离性以及持久性.区块中的共识验证字段在可以在部分共识机制中,验证节点广播的消息是否符合系统设定的条件.其具体细节将在“共识机制”部分介绍.同时,区块链要求区块只能添加在区块链的尾部且不能删除.虽然有部分工作讨论修改区块链数据的方法[37],但是该类工作并不修改区块链存储的数据.其方法通过设计视图隐藏需要修改的数据,因而并不改变区块链对数据的存储和管理.

1.2 共识机制

由于区块链网络中节点不可信并且其要求所有节点均存储数据的相同状态,当对其中一个节点的数据准备进行更新时,所有

节点需要对更新操作达成共识.共识协议往往基于两种思路,分别为基于计算的共识协议和基于通讯的协议.现有的共识机制大部分采用以上两种之一或者二者之间做一定的平衡[38].

基于计算的共识协议代表是比特币所使用的工作量证明（proof-of-work,PoW）[25].其原理为,网络中的节点通过工作量证明,确定下一个有权利将生成的区块添加到链中的节点.参与工作量证明的节点称为挖矿节点.例如,当一个区块的结构如图 1 所示,包括前一区块的哈希值,共识的验证字段以及区块内事务的哈希值时,挖矿节点利用这些哈希值,计算出该区块的共识的验证字段,使得验证字段满足以下条件：

$$\text{hash}(\text{PrevBKhash}|\text{ConsProof}|\text{MerkleRoot}) \leq Z$$

方程 1：工作量证明方程

其中Z这一参数控制工作量的大小.最先计算出相应验证字段的挖矿节点将其结果全网广播.其他的挖矿节点验证其收到的验证字段是否满足方程 1 条件.若结果满足,则将区块添加到已有区块链的尾部,而最先计算出结果的节点则获得相应的奖励.工作量证明的共识协议可以很好地应用于公有链网络中,并且可以抵御女巫攻击[39].当恶意的算力达到全网算力的 10%时,在 6 个区块后确认的交易可以保证超过 99.9%的真实性[40].同时,在工作量证明方程的结果广播的过程中,PoW 共识可利用中继节点,通过多轮传递方式来减少工作量证明共识机制的冗余网络开销[41].

另一类型的共识协议则是基于通讯的共识协议.基于通讯的协议将投票权分配给网络中的节点,并利用多轮通讯,达成共识.该协议的代表为实用拜占庭容错协议（PBFT）[26].PBFT 共识协议需经过三个过程,分别为 pre-prepare 阶段、prepare 阶段、以及 commit 阶段.当用户提交请求时,一个主节点会在 pre-prepare 阶段广播其请求.在 prepare 阶段,所有的节点收到请求后判断是否执行该操作.如同意执行,则将自己的签名添加在消息中,并广播给其他节点.若不同意执行则不发送消息.当节点接收到其他节点发送的执行操作的消息达到一定数量时,则说明执行操作达成了共识.在 commit 阶段,节点执行相应的请求,并将执行结果回报给主节点.由于所有的节点都将全网发送并接受来自所有其他节点的消息,因而在网络中有 N 个节点的情况下,其网络开销复杂度为 $O(N^2)$.

以上两种共识分别基于计算和通讯.同时,有其他共识协议改进 PoW 共识协议中的条件方程,或在两种共识协议中做一定的取舍.例如,Elastico[42]、RapidChain[43]、Byzcoin[44]、Tendermint[45]等均利用采样的方式,选取领导者节点并在领导者节点中,利用 PBFT、工作量证明或其他基于投票的算法达成共识,从而减少其共识阶段的开销.需要注意的是,基于计算的工作量证明共识并不能确定保证网络中的事务均合法.当恶意节点控制全网 33%的算力时,则恶意节点已有可能控制整个网络[46, 47].与之不同,PBFT 共识则是确定性的.但是由于其网络开销为 $O(N^2)$,当增加网络节点个数时,网络产生的额外开销会超过增加算力所节省的时间,使得系统的性能随着算力增加而降低.因此直接使用 PBFT 共识将会降低整个网络的可扩展性[48],并且成为部分区块链系统的性能瓶颈[22].关于其他共识机制的综述可参考文献[49,50].

1.3 UTXO与Account模型

在区块链系统中,典型的数据模式是比特币所基于的 Unspent Transactions Outputs（UTXO）结构.以比特币为例,在 UTXO 模型中,不存在一个账户记录一个用户所持有的所有比特币.每个交易均是通过已有的 UTXO 生成新的 UTXO.换言之,交易只是代表 UTXO 集合的变更,而一个用户所拥有的全部比特币是其所拥有的 UTXO 总和.例如,假定用户 A 分别在事务 1 和事务 2 中转账 2 个比特币到 B 用户.此时 B 用户有两个 UTXO,$UTXO_1$ 与 $UTXO_2$,且分别包含 2 个比特币.当用户 B 计划生成事务 3,并在事务 3 中转账 3 个比特币到用户 C 的账户时,事务 3 的输入需要包含的是 B 用户的 UTXO.本例中,事务 3 的输入可以为 $UTXO_1$ 以及 $UTXO_2$.之后在事务执行过程中,事务将输入的 UTXO 标记为已花费的状态,并生成一个新的包含 3 个比特币 UTXO 并将其转移到 C 用户.同时,事务生成一个 UTXO 包含剩下的一个 1 个比特币并将其转移到 B 用户.通过 UTXO 这一模型,所有交易均在已有的 UTXO 之上进行,大多数交易只需要判定是否存在双花（一个 UTXO 被多次交易）即可.因而交易是否被执行可以很快地验证并且具有很高的隐私性.但是由于其不存在账户的概念且交易需要基于 UTXO,其编程复杂度较高.

另一种数据模型是 Ethereum 所采用的 Account 模型.该模型下一个账户包含用户所有可用的电子货币.当一个交易被提交时,相应的事务首先判断账户内的电子货币是否足以完成该交易.如果可以,则该交易被执行.否则,交易被取消.与 UTXO 相比,Account 模型更容易理解,并节省了大量的存储空间.而 UTXO 则可以更好的保护用户隐私.

1.4 智能合约

智能合约最早于 1994 年由 Nick Szabo 提出[51].其设想为在一个计算机系统中,当一些事务被执行时,可以激发相应的代码自动执行,并产生相应的输出合约.但是由于很难确保智能合约的代码等不被篡改,因而其在实际的应用受到了很大的限制.随着区块链技术的发展,利用区块链所支持的数据不可篡改的特点,区块链上的智能合约得到了越来越多的重视.智能合约可以看作是一段区块链所有节点都认可的代码.部署在区块链上的智能合约需采用该区块链系统可执行的代码,并同样存储于区块中.当智能合

约的条件被满足时,合约自动被激活并执行相应的代码.例如,一个执行转账功能的智能合约会首先判断其激活条件是否被满足.若被满足,则进一步验证相应的事务是否可执行,包括事务是否被篡改、支出账户的金额是否足够等.在验证合法之后,合约将自动执行相应的事务.

1.5 区块链类型

根据节点加入或退出区块链系统的方式,区块链系统可分为公有链、联盟链、私有链等三类.

公有链系统也称为非许可链,允许任何节点参与区块链数据维护和读取,同时节点也可以随时加入或离开该网络.因此公有链系统是一个完全的去中心化的系统.典型的公有链系统包含比特币[1]、以太坊[52]等.在公有链中,其共识机制一般是基于计算的工作量证明共识.理论上,恶意节点需占有网络中 51%的算力才可以控制该区块链.

联盟链也称许可链,是一种新加入的节点需要获得许可的区块链.联盟链限制可参与的成员节点,且节点的读写权限以及所负责的计算由联盟链的设计规则来确定.典型的联盟链包括 Hyperledger Fabric[31]等.整个联盟链由所有节点共同维护,但是与公有链不同,联盟链可以假定网络中部分节点是可信的.节点的加入以及共识过程一般通过可信的节点控制.联盟链一般不采用工作量证明的共识机制,而是多采用 PBFT 等基于通讯的共识算法.

私有链则仅限信任的个体使用,且不完全能够解决信任问题.网络节点彼此之间需要透明,但不对外公开.

2 区块链的数据存储

在区块链的各种应用中,数据的存储是底层设计中非常重要的层次.由于数据库的广泛应用及其丰富的数据管理工具,很多工作研究利用数据库的存储来实现区块链的数据存储管理.针对区块链数据的存储模式主要集中在两个方向.其中一个方向为基于键值对的存储模式.该模式下存储的任一数据记录包含一个可以确定该记录的主键,并将其余部分视为该主键所对应的数值进行存储.该模式数据结构简单,可以在处理大规模数据时获得很好的性能.另一个方向是采用关系型数据的存储模式.该模式将数据以关系模型的方式建模并存储.在该模式下,可以保证数据满足诸多限定条件.例如,数据库的一致性、原子性等.因此,部分公司已经推出了基于数据库的区块链系统,包括 Amazon QLDB[1]以及 ORACLE blockchain table[2]等.该类区块链在数据库的基础上,通过设计共识机制,确保各个节点数据的一致性.

由于区块链上数据存储的成本远高于一般数据库,部分工作采用重新编码、链上-链下结合等方式来减少区块链数据存储的开销.因此,我们也将从键值对模型和关系数据模型两个方向对其进行综述,并介绍针对区块链数据存储的优化技术.表 1 分别从性能、可拓展性以及技术复杂度方面,比较现有关于数据存储的工作.

表 1 区块链存储技术的比较

Table 1 Comparision of the approaches for data storage in blockchains

区块链名称	存储方式	存储模型	性能	可拓展性	优化技术复杂度
Hyperledger Fabric[31]	链上	键值对	高	强	低
BlockchainDB[55, 56]	链上&链下	关系型	中等	弱	低
Ethereum[2]	链上	键值对	低	弱	低
[57]	链上	键值对	中等	强	高
[58-61]	链上&链下	键值对	高	强	高
Ekiden[62]	链上&链下	-	中等	强	高

2.1 基于键值对的数据存储模式

在区块链的应用中,许多系统采用基于键值对的文件系统存储其数据状态.由于基于键值对的数据库往往具有比较高效的查询处理性能,Hyperledger Fabric[31]、 Ethereum[52]均采用键值对数据库存储其数据.比较常见的区块链键值对存储系统包括 LevelDB[53]、 RocksDB[54]等.

在典型的区块链系统中,各个节点均要存储区块链内数据的备份.在这种情况下,数据在链上的存储代价极高.同时,考虑到部分共识协议需要大量的网络开销,在数据规模极大的情况下,直接将所有数据存储于区块链中将产生巨大的存储开销和网络通讯代价.因此,部分工作旨在以较小的代价提高数据存储规模,包括分布式存储、链下存储等技术.

[57] 研究通过分布式编码的方式提高区块链存储规模.具体而言,在[57]中,区块链的节点分为多个区域,并且保证每个区域内

1 https://aws.amazon.com/cn/qldb/

2 https://blogs.oracle.com/blockchain/

的所有节点数据可以通过整合,生成整个区块链数据.因而,在这种情况下,一个区域的单一节点不需要存储整个区块链内数据副本,而只需要存储部分数据,即可达到数据一致性的要求.在编码的过程中,首先假定要将数据分割为 n 个不同的块,并将其分配到一个区域的 m 个节点中.利用分布式存储的编码技术,例如 MDS 编码,可以生成相应的数据分配方案,使得在 m 个节点中,获取任意 k 个节点的数据,即可获得所有原始数据.因此整个系统的数据可以在存在不超过 k 个恶意节点的情况下,依然保证一致性.与此同时,为了保护数据隐私,数据利用 Shamir's secret sharing 技术加密并将密钥分配给一个区域内的各个节点.Shamir's secret sharing 技术可以保证各个节点独自的密钥均不相同且不与原始密钥相同.当需要获得完整数据时,系统可通过各个节点的密钥计算出原始密钥,再对数据进行解密.由于该方法减少了每一个节点的存储开销,因而其具有很好的拓展性.但是由于需要对数据进行编码操作,因而在数据动态输入的过程时,需要额外的编码计算开销.

当利用区块链存储相应的数据时,如果数据规模极大,直接在链上存储所有的原始数据会导致存储开销极大,且降低系统的性能.为了减少相应的数据存储开销,[58, 59] 研究基于区块链的链上链下混合存储的方式.在链上链下混合存储的模型中,数据均以键值对<key, value>的格式进行存储.原始数据往往直接储存在相应的云服务器中.云服务器中的数据不在区块链中,而是相当于在链下.在区块链中存储相应数据的哈希值,即<key, hash(value)>.这样链上的数据不包含原始数据,而只有 key 以及数据的哈希值.当用户通过云服务获得数据时,可利用区块链上的哈希值对数据进行验证,从而可以保证数据的完整性与一致性.同时,由于在区块链上存储数据以及部署智能合约均需要存储开销以及相应的货币开销,[60, 61] 提出了多种基于链下存储的优化技术.其中,在计算方面,可以将状态检查等计算采用链下验证的方式,而不需要采用链上的智能合约,从而进一步降低存储开销.Ekiden[62]则提出为了保证智能合约的隐私性,智能合约也可以在需要的情况下采用链下存储.为了保证安全性,链下的执行环境需为可信的执行环境（例如采用 Intel SGX 处理器）.同时,区块链存储相应的智能合约状态的哈希值.在这样的结构下,网络同时存在计算节点与共识节点.计算节点负责智能合约的状态的改变以及计算等,而共识节点只负责针对智能合约的状态哈希形成共识.

由于区块链存储全部的历史数据,部分区块链操作需获取不同历史版本的数据.Forkbase[21]设计了支持数据版本查询的数据存储系统.针对版本控制问题,Forkbase 将数据拆分为块,并在块的级别减少重复的数据冗余存储.具体而言,Forkbase 利用 POS-Tree（Pattern-Oriented-Split Tree）这一索引来快速地确定重复数据.类似于 B+Tree,在 POS-Tree 中每个叶子节点包含相应的数据,而非叶子节点保存其子树的键值.非叶子一般存在多个不同的分支.与此同时,POS-Tree 利用默克尔树的特性,对每个节点计算其相应的哈希签名.内部节点的签名由其子节点的签名决定.因此,当一个分支的根节点哈希值没有变化时,则可以确定该节点的所有子节点所包含的内容均没有更新版本.相应的,当两个 Pos-Tree 中节点的数据不同时,必然导致其祖先节点的签名不同.因而在查找不同的数据时,只需从签名不同的根节点开始,搜索其子树中带有不同签名的节点即可.在[21]中,Forkbase 被应用于 Hyperledger Fabric,以取代 Hyperledger Fabric 的存储层.

2.2 基于关系型数据的存储模式

在关系型数据中,数据往往以表的形式进行存储.存储于同一表的数据具有相同的模式,而不同表之间的数据根据主外键等产生关联关系.关系型数据一直以来都被广泛应用于企业的管理等诸多方面.[55-56] 提出了 BlockchainDB 这一将数据库的关系模型概念与区块链结合的系统.由于企业间的交易往往希望将部分关系型数据在二者之间公开并获得共识,BlockchainDB 主要针对共享表类型数据,并在多个用户均需要对共享表进行操作的情况下,支持对表进行数据验证.该系统采用联盟链的设计方式并假定节点不可信.区块链作为数据的存储层,保证数据在不可信环境下的一致性.在一个 BlockchainDB 系统内部,节点间存在多个分块.每一块独立地维护一个区块链并存储共享表的部分数据.每一块独立的区块链保证了分块内数据的一致性.

在每一块的区块链存储层之上,BlockchainDB 构建数据库层.数据库层负责的任务包括：（1）数据库层记录表的任一条记录存储在哪一些分块中;（2）提供用户可直接调用的 API;（3）保证用户提交的事务处理一致性.当需要进行读写操作时,用户需提交数据所在分片的 ID.之后 BlockchainDB 会在相应的分片中执行读写操作.由于采用分块的方式管理数据,当用户获得数据时,需验证所得到的数据是否被恶意篡改及其完整性.作者在文中提出了 Online 和 Offline 两种验证方法.在 Online 验证方法中,用户在提交读请求并获得相应的数据之后,广播其所读数据并要求所有存储该数据的节点进行验证.如果大部分节点返回的消息表示所需要验证的请求以及数据符合本地的数据,则返回认可读取到的数据的消息.当大多数的节点均认可所读数据时,则认为数据已被验证.否则,则判定读取的数据被恶意篡改.在 Offline 验证方法中,采用延迟验证方式,即当用户提交验证请求时且验证请求达到系统设定的参数 e 时,BlockchainDB 开始进行验证.

3 区块链事务处理

与数据库中对事务处理的要求类似,区块链中的事务同样要求保证其原子性（atomicity）、一致性（consistency）、隔离性（isolation）与持久性（durability）.原子性要求事务的所有操作,或者全部完成,或者全部不完成,而不会存在中间状态.在区块链中,当一个智能合约对多个区块链进行操作时,同样要求其所有操作全部完成,或全部不完成.隔离性则是针对数据资源的并发访

问,规定了各个事务之间相互影响的程度.在区块链中,为了提高吞吐量,区块链中的并发控制也得到极大了关注.在本部分,我们将从原子性、隔离性、一致性三个角度,综述现有工作中区块链的事务处理技术.表 2 总结了部分工作针对事务的三个特性所采用的主要技术方式及其针对的事务处理的阶段与对象.

Table 2 Comparison of the approaches for transaction execution

表 2 针对事务处理的技术比较

事务特性	针对对象	针对的阶段	技术方式	文献
原子性	跨链事务	-	特殊节点处理	[63,74]
	智能合约	部署阶段	特殊节点处理	[75]
隔离性	一般事务	排序阶段	读写冲突图	[64]
	一般事务	排序阶段&执行阶段	读写冲突图	[65]
	智能合约	验证阶段	锁机制	[67]
	智能合约	验证阶段	软件事务内存 STM	[68]
一致性	一般事务	-	快照隔离	[70]

3.1 确保原子性的处理技术

[63] 研究多链之间事务的处理过程,以保证其原子性.当一个事务需要在不同的链上进行读写操作时,一个保证原子性的直接做法是将多个链合并成一个链,进而使得所有的事务处理只需在一个链上进行.这种方式在不需要考虑数据隐私的情况下是可行的.但是实际中,用户或组织之间进行交易时,不希望自身内部的交易也被公开.换言之,当事务涉及多个链时,需要考虑每个链上单独的数据隐私问题,同时也要确保跨链操作的可信性.因而[63]提出事务应存储在所有与其相关的链上.同时,跨链的事务应该保证原子性,即只能在所有相关的链上均执行操作,或者均不执行（即 abort）.

为了保证事务的原子性,[63]提出了两类方法.第一种方法需要区块链网络中存在额外的负责排序的节点.首先,所有的事务均广播其签名,从而确保发送消息的真实性.之后,所有的区块链针对其自身所需要处理的事务,形成自身的事务处理顺序.在形成顺序之后,区块链的节点将需要参与跨链操作的事务以及该事务在本地区块链事务顺序中的前一个事务的哈希值,合并为一个消息进行广播.对于负责排序的节点,其获得的消息包括跨链事务,以及跨链事务在各个区块链中前一个执行事务的哈希.排序节点根据获得的消息,确定跨链事务是否执行以及执行顺序.

第二类方法没有相应的排序节点,但是各个区块链有部分节点参与跨链事务的排序.具体而言,与第一种方法类似,首先各个区块链生成独自的事务执行顺序.在形成顺序之后,区块链的节点同样将需要参与跨链操作的事务以及该事务在本地区块链事务顺序中的前一个事务的哈希值,合并为一个消息进行广播.与第一种方法不同,该广播发送到所有区块链的各个参与排序的节点中.参与排序的节点利用 PBFT 或 Paxo 等基于投票的共识协议,确定跨链事务产生的顺序,并将其广播到所有链的节点.各个区块链依据产生的顺序执行事务.

[63]利用额外的节点或已有区块链网络中的部分节点,负责事务中所有操作的执行顺序问题,以保证事务的原子性.与之不同,[74]考虑利用锁机制实现跨链事务的原子性.当需要跨链事务时,首先区块链 A 的用户利用智能合约对其资产进行加锁操作,使其不能被其他事务使用.同时,利用哈希函数$h = H(s)$产生签名并将签名与资产写入到智能合约中.该智能合约中需添加条件为,如果另一方可以提供 s',使得$h = H(s')$,则向另一方转账 A 中被锁定的资产.另一方可通过查看智能合约,确认合约的条件以及相应的 h 值.之后在自身的区块链部署智能合约.合约的框架为,如果 A 提供 S,使得$h = H(S)$,则向 A 转账.[74] 利用以上的基于哈希函数锁机制,实现了跨链交易的原子性.

[74] 的方法可以保证跨链事务的原子性,但是其要求所有的参与者需先选举一个领导者,并由领导者串行地部署相应的智能合约,而串行的方式导致部署智能合约的开销很大.[75] 针对[74]的问题,研究如何在多条区块链并发地部署智能合约,同时保证跨链事务的原子性.其核心思想在于,通过智能合约,确保当事务的原子性没有被满足时,相应的事务被终止并且回滚所有已经执行的操作.例如 A 持有比特币与 B 持有的以太币进行兑换,只有当 A,B 均获得对方的货币时候,整个事务才可以结束.当 A 转账给 B,但是 B 没有转回相应的金额时,则该事务在被终止的同时,需确保 A 拿回自己的那一部分金额.在[75]中,论文作者假定每一组交换类型的事务中有一个发起者,并且有一组节点负责各个区块链间的共识.该组节点被称为观察网络.事务的发起者负责部署智能合约到该组节点中.首先,多组交换的事务可以建模为一个图.图中的节点表示一个账户,边表示账户间的交易.一个合法的交换事务需要满足的条件为：该事务所对应的图,在删掉表示发起者的节点之后,余下的图不可以存在环.发起者将事务所对应的图结构签名并广播.其余参与交易的账户验证该图结构,并部署相应的智能合约在自身的区块链中.同时,发起者部署一个智能合约到观察网络中.该智能合约的触发条件为当所有相应的账户均执行对应的智能合约后,确认交易.否则,则将所有智能合约锁住的资产回退到相应的账户中.在[75]中,观察者网络既可以是可信的节点,也可以是非可信的环境.

3.2 确保隔离性的处理技术

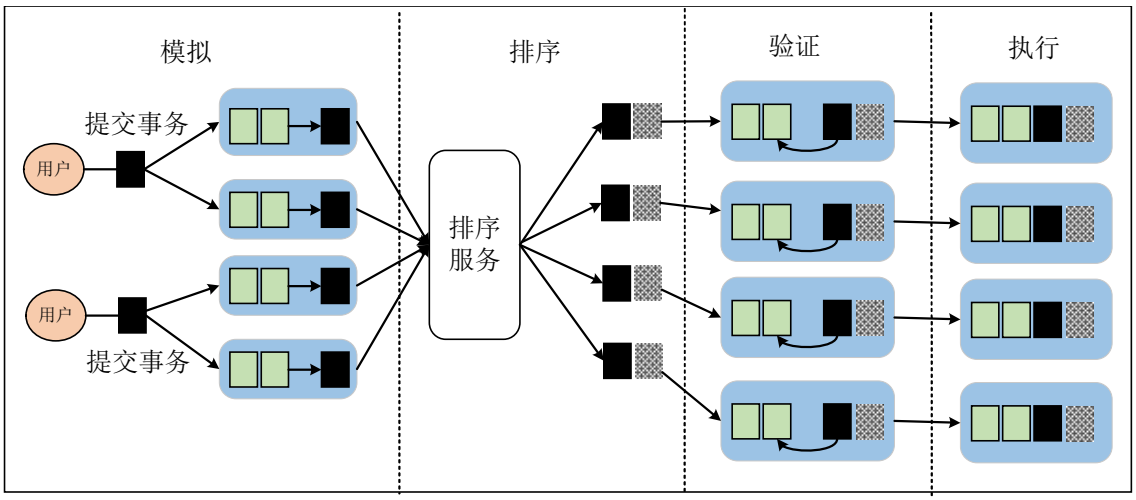

Fig. 2 Transaction Processing in Hyperledger Fabric

图 2 Hyperledger Fabric 中的事务处理流程

为了提高区块链系统的事务处理效率,分布式系统中的部分并发控制技术已经被应用到区块链系统中.本小节主要综述针对事务隔离性的技术.

在数据库中,针对事务的并发处理典型的方式采用先排序后执行的策略.在区块链系统中,很多系统依然沿用这种策略[1, 52].首先,在排序过程中,所有的节点需要达成一个针对所有事务顺序的全局共识.之后在执行阶段,所有节点在本地按照已经达成共识的顺序执行相应的事务.与数据库中的先排序后执行的策略不同,在 Hyperledger Fabric[31]区块链系统中,其工作流程则是采用先模拟后排序的思路,如图 2 所示.整个流程包含了模拟阶段、排序阶段、验证阶段、执行阶段四部分.在模拟阶段,客户端将一系列的事务处理请求提交给部分节点.收到请求的节点将在本地的数据中模拟事务的执行.节点会产生一个签名并将签名以及事务所需进行的读写数据集合进行广播.在排序阶段,排序服务节点接受所有提交的事务并确定事务执行顺序.默认情况下,事务会按照提交时间的顺序进行排序.在排序结束之后,这些事务被组织成一个区块并分发给网络中的节点.在验证阶段,节点按照顺序确定事务是否可执行.在该阶段需验证两部分内容.首先,节点验证事务的签名是否满足共识条件.如果不满足,证明存在节点篡改了客户提交的事务.此时该事务被标记为不可行.第二个方面,节点验证提交的事务顺序是否存在读写冲突.如果某一事务与排序在其前面的事务存在读写冲突,则该事务被标记为不可行.在执行阶段,所有可行与不可行的事物均被写入区块中.同时,可行的事务被执行.常用的先排序后执行的策略在排序之后,区块链中的各个节点只能顺序地执行相应的事务.而 Hyperledger Fabric 则可以利用模拟阶段的结果,并发地执行事务并保证数据的一致性,从而提高系统性能.针对 Hyperledger Fabric 的各个阶段,如表 2 所示,许多工作针对不同阶段提出相应的优化技术,以提高系统的并发度.

在事务的并发控制过程中,如果事务间存在读写冲突,那么只有部分事务可以成功执行.其他的事务将被中止.在[64]中,通过实验发现,即使 Hyperledger Fabric 可以提高事务的并发度,直接利用读写冲突处理区块链的事务依然导致大量的事务被终止.[64]研究通过改变事务的执行顺序,减少事务中止的个数,从而提高并发度.在 Hyperledger Fabric 的“模拟-排序-验证-执行”过程中,可以从两个方面提高事务的并发度.首先,一个区块中的事务默认按照提交时间的顺序进行处理.然而,可以通过改变区块内事务的顺序,使得原本被终止的事务在新的顺序下没有冲突,从而可以顺利执行.其次,由于事务的处理过程需要四个阶段,而现有工作只能在验证阶段判断事务是否被执行或终止.如果可以在该阶段之前判断事务不可执行,则可以提前将事务标记为终止状态,从而节省后续阶段的开销.

基于以上两点,针对区块内事务的顺序问题,[64]提出了利用冲突图的方法产生执行顺序的算法.具体而言,在冲突图中,图中的一个节点表示一个事务.当两个事务T_i, T_j存在读写冲突时,相应的事务的点之间存在一条有向边$T_i \leftarrow T_j$,表示T_i应在T_j之前执行.而一个区块中,代表所有可执行的事务的点以及之间的边所构成的图一定是冲突图中的一个无环子图.因而作者提出了启发式的算法,在冲突图中,通过查找其中最大的无环图,进而产生相应的事务执行顺序的算法.

同时,针对区块间的事务读写冲突,作者提出分别在模拟阶段与排序阶段判断事务是否可行的机制.如果不可行,则事务不需进入到下一个阶段.在模拟阶段,系统的节点需对数据进行读操作.节点的数据都添加了相应的版本号.当一个事务的多个读数据的版本号不一致时,则证明部分读入的数据已经被其他事务修改,则该事务被标记为不可行.在排序阶段,当两个事务都需读入相同数据且后一事务读入的版本号与前一事务不相同时,则后一事务被标记为不可行.其原因在于,Fabric 的事务执行不是以单一事

务为单位,而是以区块内所有事务为单位.因而区块内事务读入的数据应具有相同的版本号.

ParBlockchain[65]以及[66]研究了在 Hyperledger Fabric 中事务的并行问题,并针对执行阶段进行优化.在 ParBlockchain 的执行阶段,与[64]类似,部分节点根据事务的读写冲突,建立关系依赖图.图中的节点表示事务,并用有向边表示事务的冲突.该图在建立后被传送到负责排序阶段的节点中.排序节点根据关系依赖图确认事务的执行顺序.在执行层面,当多个事务属于不同的应用,或者不同事务之间不存在读写冲突时,则这些事务可以并行执行.ParBlockchain 将执行事务的节点按照不同的应用进行分类.一个节点只负责处理一部分应用的事务.当需并行执行不同应用的事务时,处理不同应用的节点可并行地执行对应应用的事务.在执行完成之后,各个节点广播其执行结果,即可保证所有节点数据的一致性.

以上工作主要针对的是排序与执行过程.针对验证阶段,[67, 68] 研究针对智能合约验证过程的并发控制.智能合约在公有链上的执行一般分为两个阶段.在第一个阶段,挖矿节点负责串行的执行智能合约代码并将新生成的事务、智能合约的新状态、前一个区块的哈希等均存储在一个新的区块中.之后该区块会通过挖矿节点间的共识,加入到区块链中.在第二个阶段,所有网络中的节点会验证该区块内的事务与结果是否与本地的数据一致.只有当对应的数据均一致时,该区块才会被节点所接受.因此,在第二个阶段,智能合约所产生的事务会串行的被节点重新执行.[67,68]通过不同的机制,实现该阶段智能合约的并发验证.在[67]中,作者利用加锁的方式实现并发控制.对于挖矿节点,其在执行智能合约代码的同时,生成 happens-before 图.该图中每个节点代表一个智能合约.节点与节点的边记录智能合约的执行需遵循的先后顺序.该顺序可根据智能合约间的读写冲突确定.挖矿节点在生成 happens-before 图之后将其写在区块内.区块链网络的节点需要验证区块内容时,可根据 happens-before 图,确认可以并发执行的智能合约.与[67]中加锁机制不同,[68]则利用软件事务内存（STM）机制来处理智能合约的并发执行.[68]将一个智能合约视作一个原子事务,原子事务中包含的操作只可能全部执行或者全部取消.在验证过程中,所有智能合约均会按照时间戳的顺序并发地执行.同时,节点会检测智能合约之间是否存在读写冲突.当智能合约间存在冲突时,则终止其中一个合约并在之后重新提交该智能合约的事务.[69] 则研究事务处理的顺序如何满足拜占庭容错.

3.3 确保一致性的处理技术

针对事务的一致性,[70] 提出利用数据库的一致性来处理区块链中并发事务的一致性问题.现有数据库可以支持执行多种事务与查询,因而在数据库上建立区块链的一个核心问题是如何确保所有数据库节点执行事务的顺序完全一致,从而可以保证节点数据的一致性.在[70]中,为了保证区块链的安全性,作者假定在联盟链的环境中,每一个节点均需要对其提交的每一个事务进行签名.其余节点可根据签名验证事务的真实性.为了提高吞吐量,在确定事务处理顺序时,作者优化了快照隔离的并发控制方法（serializable snapshot isolation）[71],使其从针对事务的并发拓展到针对区块的并发,并提出两种针对一般事务的并发控制策略.

其一为传统的先排序后执行并发控制.在该方式框架下,系统中的部分节点负责对提交的事务进行验证并排序.由于在区块链中,一个区块内的事务需全部执行之后方能完成.当存在写操作时,若采用传统数据库中的快照隔离方法,在多个事务均对一个数据进行写操作时,一个事务需等待之前所有的事务写操作均完成之后方可开始执行.这将导致系统性能大幅下降.因而,作者提出,在事务执行阶段,不考虑事务所处的区块,而是直接对事务进行排序.与此同时,对数据的写操作会产生多个副本.当事务完成时,按照事务在区块链中的顺序确认所产生的多个副本中的一个为合法的副本.

第二种方式为先执行后排序的并发控制方式.在该种方式下,同样利用快照隔离来进行事务的并发处理.但是该方式需要处理三个问题,首先,先执行再排序要避免幻读的可能,即读入还未写入的数据;其次,要避免脏读的可能,即读入已被更新的数据;最后网络中不同的节点由于延迟等原因,数据可能未达成一致,进而可能使得事务所需访问的数据不存在.为了处理这些问题,作者提出了基于区块高度的快照隔离.具体而言,每一个数据库节点都按照区块链的内容来存储数据.每一个节点记录相应的已写入的区块链的个数.区块链的个数可区分现有区块链的快照.当一个节点 A 的区块的个数多于另一节点 B 时,则 A 节点的数据已更新,而 B 的数据还未更新.作者设计了基于区块链高度的并发快照隔离方法,确保所有的节点均可在事务完成后,保证数据的一致性.

[55, 56] 提出了 BlockchainDB,一个将数据库与区块链相结合的系统,并设计了多种一致性水平,以提高事务写的效率.在这种情况下,当用户提交查询时,数据库层的 TransactionManager 负责分发提交的查询并确保不同的一致性.具体而言,BlockchainDB 有两种一致性水平可供选择.第一种为串行一致性.在串行一致性中,所有的写操作按照提交的顺序插入队列中.当多个写操作同时对一个数据进行写操作时,只有队列中的第一个写操作可以被执行.其他的写操作均处于等待状态,直到写操作完成.第二种为最终一致性.在最终一致性中,写操作依然按照顺序插入到队列中,但是所有的读操作可以马上执行.在这种情况下,脏读的情况可能会发生,即数据在被读入之后,被其他的写操作覆盖,因而之前所读的数据与存储的数据不一致.当用户提交查询时,由分片管理器确认查询所需的数据处于区块链中的位置.之后,该位置的数据会被并行的读写.由于 BlockchainDB 支持不同的一致性,查询的效率在不同的一致性下将会不同.例如,在串行一致性下,BlockchainDB 的查询可能需要等待队列中写操作的完成之后才可进行.

4 区块链查询处理

在区块链中,所有数据均以链表的形式存储.与一般数据库的查询不同,区块链存储的数据包含在所有事务当中.区块链查询分为一般查询与可信查询.表 3 从对关系型查询语句的支持、可信性查询以及链上链下数据查询等多方面比较了最新的研究工作.

Table 3 Comparison of the Query Processing in Blockchains

表 3 区块链查询处理比较

区块链	关系型语义	支持可信性查询	支持链上链下数据查询
Chainsql[79]	强	弱	✗
BigchainDB[80]	弱	弱	✗
SEBDB[81]	强	强	✓
Bitcoin[1] & Ethereum[2]	弱	弱	✗
vCahin[84]	弱	强	✓

4.1 一般查询处理

LineageChain[82]研究区块链环境下数据溯源查询的效率问题,并支持需要数据溯源的智能合约.LineageChain 针对每个数据,赋予其初始版本号.当一个事务对数据进行读写操作时,事务的输出可能是新的数据或者更新现有数据的版本.这钟情况下,如果一个区块内还有不同事务时,则可以视作一个区块的事务对前一区块数据库状态进行的修改.因此,数据版本号与区块同步递增.LineageChain 将这种递增关系建模成为一个无环图.当需要溯源查询时,利用无环图中点的拓扑序进行搜索.

与[82]需额外构建图结构不同,[83] 研究利用智能合约确定数据的关系,并对数据进行溯源查询.当一个数据的创建者创建一个文档时,创建者可在区块链中部署一个智能合约.该合约的功能包括更改数据拥有者、添加可访问用户、进行溯源查询等.同时,该合约要求所有对指定文件的更改需经过投票过程.参与投票的用户由数据拥有者确定.对数据的任一次更改需至少半数以上的用户投票同意才可进行.在这种环境下,恶意的用户若对数据进行恶意篡改,需控制半数以上参与投票的用户.同时,由于所有的修改均利用部署的智能合约,因而对数据的溯源查询转化为对智能合约调用的查询.

为了使得区块链查询处理语句与数据库的 SQL 查询语句类似,[81]提出了 SEBDB 并设计了与数据库 SQL 语句类似的查询语句,使之可以完成在区块链中对关系型表的建表、插入、选择等操作.在 SEBDB 中,数据分别采用链上链下存储方式.链下存储的数据由一个本地关系型数据库进行管理.针对链上与链下的数据,SEBDB 设计了包括连接操作等语句.当只需要链下数据进行连接操作时,SEBDB 可以直接利用本地数据库的 Join 语句来进行.当需要链上链下数据同时进行连接时,SEBDB 首先扫描链上数据并利用 hash-join 的方法,将其与链下数据进行连接.为了提高查询效率,SEBDB 设计了区块层级的 B+-Tree.SEBDB 利用区块 ID、事务 ID 以及时间戳作为每一个 B+-Tree 里面数据的排序键值.当给定区块 ID、事务 ID 或者时间戳时,可利用区块级的 B+Tree,快速找到存储相应数据的区块.同时,SEBDB 建立表与区块的索引,即每个表的索引指向所有包含对该表进行操作的事务的区块.利用以上索引,使得对链上数据的访问不需扫描所有区块.[116]在 SEBDB 基础上,通过建立数据的视图,进一步提高其查询效率.

4.2 可信查询处理

考虑到区块链应用在不可信的环境下,当用户提交查询并获得查询结果时,往往需验证其查询结果的可信性[95-98].[84]提出了一个区块链系统 vChain,实现了高效的可验证查询处理算法.该系统假定用户节点并不一定存储整个区块链的全部数据,而是只存储所有区块中的哈希值.由于一个区块的哈希值所占的存储空间远远小于区块内存储事务所花费的空间,因而该假设即使在轻量级的硬件环境下也可实现.为了验证查询结果,在每个区块中添加一个额外的命名为 AttDigest 的字段.具体而言,AttDigest 由多集合加密累加器（Cryptographic Multiset Accumulator）所生成.所生成的 AttDigest 具有如下性质：给定同一公钥,当需验证两个集合的交集是否为空时,可以通过各自的 AttDigest 直接验证.利用该性质,当查询条件可以表达为集合交集的形式时,可利用返回结果的 AttDigest 与查询条件中集合的 AttDigest 进行比较,来确定返回的结果与查询的条件是否存在交集.如果发现二者不存在交集的关系,则验证查询结果和查询条件不匹配.

[84]进一步将该方法推广到范围查询.范围查询中,用户会对值域为数字的属性给定一个区间.查询结果返回在该区间中满足条件的数据.当验证范围查询的查询结果时,无法直接将 AttDigest 应用到范围查询中.其原因在于 AttDigest 只适用于判定集合的关系.因此,作者将范围查询转化为集合查询.具体而言,首先 vChain 将所有数字以二进制形式进行标识,并将其视作字符串.利用二进制表示,vChain 构建了前缀树来表示所有数字.每个叶子节点是一个具体的数字,而非叶子节点对应一个数字的范围.当一个查询中包含范围条件时,该范围对应前缀树的一部分连续的叶子节点.因此可将其转化为基于前缀的集合表示形式.例如,当查询范围为[0,6],且该属性数据的范围为[0,7]时,其对应的表达形式可为 0*∩10*∩110.符号*表示二进制的 0 或 1.基于此集合表示,则可以将 AttDigest 的签名同样应用于范围查询.

同时,为了提高验证效率,vChain 提出了区块内以及区块间的索引.区块内的索引为一种类似于默克尔树的结构.树中的每一个节点包含三个域,分别为左子树哈希值、右子树的哈希值和集合属性所包含的元素.利用该树结构,查询处理可以看作对树的搜索.当搜索到树中节点时,如果该节点的集合的 AttDigest 不满足查询条件,则不需要继续搜索该节点的子节点.同样的思想也被用来建立区块间的索引.vChain 将区块按照指数递增的方式来分段.例如,每段区块域包含的区块的个数分别为 2、4、8 等.每一个区块域均计算相应的 AttDigest.当一个区块域所包含的所有集合元素所产生的 Attigest 均不满足查询条件时,则整个区块域可以不被搜索.利用区块内以及区块间的索引,vChain 提高了查询验证的效率.

同样针对范围查询,[85]研究在以太坊框架下,查询结果验证的开销问题.开销是指在以太坊框架下,执行操作所产生的 gas 开销.在以太坊的框架下,当用户需要存储以及执行智能合约时,需要支付相应的 gas.gas 可由以太币兑换.在智能合约中,不同的操作需要支付不同数量的 gas.例如,从区块中读数据需要 200gas,更新操作需要 5000gas,而写操作需要 20000gas.在这种情况下,当用户执行查询验证时,一个考量是如何用尽量少的 gas 来达到验证的目的.基于此,作者提出了GEM^2-Tree 这一数据结构,来达到减少 gas 开销的目的.一般验证查询结果需要区块链以及用户端均产生相应的验证签名,并在查询返回结果之后,通过签名进行查询结果验证.针对区块链这一读写操作开销差别极大的情况,作者提出利用代价低的读操作代替代价高的写操作的思想,并提出了 SMB 树.SMB 树是一种压缩的默克尔树.它并不显示地存储默克尔树的非根节点,而是只在区块中存储其根节点的哈希值.整个智能合约的索引包含一个完整的默克尔树以及多个 SMB 树.新加入的数据总是先加入到 SMB 树中.由于 SMB 树一般规模较小,所以针对插入操作,其 gas 开销较小.由于区块链中包含一个默克尔树和多个 SMB 树,当执行范围查询时,服务器需要搜索整个默克尔树,并在相应数据上执行查询处理过程.在得到结果的同时,服务器也需要将包含结果的默克尔树或者 SMB 树的验证信息全部返回.客户端获得所有验证信息后,需要利用相应的验证重构默克尔树或 SMB 树并进行查询验证.

Veritas[99]针对数据库共享表的情况,设计支持验证的区块链.Veritas 将数据存储在数据库中,并且允许共享表可被网络中的节点访问.在数据库之外,数据签名以及共识投票则存储在区块链中.为了保证共享表在各个数据库的一致性,Veritas 利用可信的节点来对事务进行排序.与一般的区块链系统不同,Veritas 的各个节点只在自身的数据库中执行针对共享表的事务.所有节点在一段时间内广播其所有新生成的日志以及执行事务所进行的读写数据.所有节点对于广播的日志以及数据,利用投票方式来确保一致性.

5 区块链可扩展性

分布式系统的可扩展性主要考虑两个方面的因素：每秒处理事务数（TPS）和网络节点数.而区块链在这两方面均受到一定的制约[22].其原因在与为了满足拜占庭容错,节点间需要满足共识协议条件.无论是基于计算的共识协议还是基于通讯的共识协议,都会产生极大的计算开销和网络开销.另一方面,传统的中心化系统可利用平衡不同节点的工作量等方式提高系统的吞吐量.但是在区块链中,节点均存储区块链中的数据且均需执行相应的事务.

[90]讨论了提高区块链吞吐量的诸多挑战.部分工作,例如 BigchainDB[80]等,针对比特币所支持的 UTXO 这类交易数据模式进行优化,以提高可拓展性.而大部分提高区块链吞吐量的工作针对的是 Account 模型下的数据模式.由于区块链系统中,共识的达成是基于区块的粒度,所以一种简单的提高吞吐量的方法是通过调整系统的各个参数,例如区块的大小,使得系统的 TPS 得到提升.该种方法在公有链环境下对性能有一定的提升. 但是 StreamChain[91]指出,改变区块的大小可以提高基于工作量证明共识协议下的区块链系统吞吐量,而在联盟链中,多采用的是基于通讯的共识.在这种情况下,增加区块的容量反而会使系统性能降低.Iota[92]、Hashgraph[93]提出了将区块链的链式结构拓展到无环图的情况以提高区块链的共识效率.但是,由于无环图同样可以产生一个全局的区块顺序并且可以根据该序列转化为链式结构,因而本章节主要针对链式结构下提高区块链可扩展的技术,尤其是分片以及链下事务处理技术.

5.1 基于分片的区块链系统

为提高吞吐量,将数据库中分片技术引入到区块链系统中是研究的一个热点问题[94].表 4 总结了部分区块链可扩展性的特点.

Table 4 Comparision of the Scalability of Blockchains

表 4 区块链可扩展性比较

区块链	委员会节点加入条件	委员会节点替换策略	通讯复杂度	吞吐量与节点数关系
Elastico[42]	PoW	全部替换	$O(n^2)$	正比
Omniledger[87]	Permissioned	部分替换	O(n)	正比
RapidChain[43]	Permissioned	部分替换	O(n)	正比
ChainSpace[88]	-	-	$O(n^2)$	正比
Monoxide[89]	PoW	-	-	正比
Hyperledger Fabric[31]	Permissioned	-	$O(n^2)$	反比

Elastico[42]、RapidChain[43]、Omniledger[87]等系统利用分片技术大幅度提升了系统的吞吐量.在一般情况下,系统中所有节点划分为多个片且每个片包含多个节点.这样,每个片都具有一定的算力可以支持区块链的共识协议.由于采用了分片技术,且区块链系统运行在非可信环境中,片的划分需要保证存在恶意节点前提下,依然可以安全进行事务处理.因此,现有的基于分片的区块链工作大多需要在每个事务处理过程中,进行片的随机再构建等操作[42, 22, 87],以保证恶意节点无法控制一个片.同时,为了降低共识阶段的开销,大多数系统会选取一部分节点组成委员会,且只有委员会节点参与共识阶段.为了保证安全性,部分区块链会在一段时间后,对委员会节点进行重新选择或替换.

Elastico[42]最早提出在公有链系统中采用分片模型.在 Elastico 中,整个网络被分为多个片.Elastico 首先会要求所有节点计算基于工作量证明的验证字段（公式 1）.最先得到该函数哈希值的 C 个节点称为字典节点.字典节点负责记录整个网络所有节点所对应的片.网络中的节点由其 PoW 提交的结果决定该节点将被分配到哪个片.因而节点的分配具有随机性,并且恶意节点无法确定其是否分配到同一个片.同时,为了防止女巫攻击,每个新加入 Elastico 公有链的新节点都必须先执行 PoW,网络中的现有节点验证新节点的 PoW 并授权其加入网络.

当一笔交易提交到网络时,Elastico 会根据事务发送者的地址,随机分配到不同的片中执行.各个委员会内对事务利用 PBFT 等机制达成共识,并最终将达成共识的事务提交到最终委员会.其中,最终委员会的节点也是根据委员会节点 ID 进行随机选择.最终委员会在节点中运行 PBFT 共识协议从而确保整个网络的一致性.由于分片具有随机性,当每个分片的节点数量不低于 600 个时,其中三分之一的节点是恶意的概率为百万分之一.因而系统的安全性可以得到保证.每执行一定数量的事务,Elastico 中所有的节点重新计算 PoW 以便重新分配节点所属的片.为了减少片间的通讯,Elastico 的消息通过 C 个字典节点进行传递,使得其最优的通讯复杂度为$O(Cn)$.

Elastico 在处理跨片事务的过程中,如果事务所需访问的数据在一个片被加锁,但在另一个片被拒绝,将会导致被加锁的数据始终处于锁的状态.为了解决该问题,Omniledger[87]在进行跨片事务验证时,采取两步验证来避免出现由于部分非法输入而导致的事务中断.首先,在第一阶段,Omniledger 锁定所有输入分片以及输出分片.所有输入分片检查自身所在的片内输入是否合法.如果合法,则生成一个包含本交易的区块签名的接收证明.类似的,如果非法,则生成拒绝证明.当所有的输入都生成相应的证明,则可判定该事务可执行或取消.在第二阶段,如果所有的输入都返回接受证明,那么事务被执行.同时,终端创建并分发一个解锁请求.每个参与该事务的节点验证并更新账本状态.如果存在一个输入所在的片返回拒绝证明,则事务取消.同时,终端生成并分发一个解锁取消请求.输入所在的片接受该请求并将已加锁的数据解锁.

RapidChain[43]同样在 Pow 共识协议下利用分片提高区块链可扩展性.与 Elastico 不同,RapidChain 不需要所有节点都存储区块链的全部数据.在这种情况下,对于一个事务,例如输入输出分别为 *In, Out* 的事务 *tx(In,Out)*,有两种情况产生.第一种情况是事务的输入与输出均在同一片内.由于数据都存储于同一片中,这种情况下,直接利用片中的节点间的共识机制即可处理该事务（RapidChain 利用近似的 PBFT 共识协议处理片中的事务）.第二种情况为输入 *In* 与输出 *Out* 的数据处于不同片中.在这种情况下,不失一般性,假定输入包含 p 组数据 $In=\{I_1, I_2 \dots I_p\}$, 输出包含一组数据 *Out*.RapidChain 在处理该事务时,*Out* 所在的片将生成 p 个新的事务$tx_i(I_i, O_i)$.每个事务输入为I_i,输出为O_i,并且满足$|I_i| = |O_i|$.同时生成一个事务$tx_{p+1}(O, O_{ut})$.其中输入 O 包含 $O_1, O_2 \dots O_p$, *Out* 为原事务的输出.在进行事务验证时,输出所在的片首先要求I_i所在的片验证事务$tx_i(I_i, O_i)$是否可执行.如可执行,I_i 所在的片将事务tx_i写入其账本,并发送O_i至 *Out* 所在的片.最终$tx_{p+1}(O, Out)$ 可利用片内的共识机制进行验证.整个过程通过将事务分割,实现了片间的共识.

Chainspace[88]则旨在解决分片情况下数据的一致性问题.当数据在多个片均存在相同的备份,且不同片通过不同的智能合约,均对该数据进行了修改,则可能导致不同片中数据不一致的情况出现.为了解决这个问题,作者提出了事务痕迹的概念.事务痕迹包含这个事务所需要访问的所有数据,以及产生这些数据所调用的智能合约.基于事务痕迹,作者提出了 S-BAC 共识算法.具体而言,当客户端提交事务时,Chainspace 根据事务痕迹,在所有可能产生数据不一致的分片中分别执行共识算法并且判断在一个分片内,该事务是否和其他事务产生冲突.如果没有冲突,则该事务在该分片内可执行.否则,返回取消命令.一个事务只有在它的痕迹在所有分片均可执行时,方可执行.当有任一分片返回取消命令时,则该事务会被终止.利用 S-BAC 共识,Chainspace 解决了不同片之间的数据一致性问题,但是代价是一个事务需执行多轮共识.

[89]提出了 Monoxide 区块链系统,该系统采用分片架构来对公有链系统进行优化.Monoxide 可以有效地解决分片结构中算力分散的问题.由于系统中存在多个片,且每个片独立地进行共识计算.当攻击者可以聚集算力攻击特定片时,极有可能攻击者具有该片超过 51%的算力,从而使得共识协议失效.为了处理这种攻击,作者提出了“连弩挖矿”这一协议.在该种协议下,Monoxide 允许一次成功的解决哈希问题可以确认多个片中的块.连弩挖矿允许矿工同时参与多个编号连续的片.一般的工作证明方程如方程 1 所示,其中默克尔哈希树根包含区块内事务的哈希值.而在连弩挖矿中,方程的默克尔哈希树根替换为由多个区块块头部分的哈希值组成的新哈希值.这样,在确认块时,哈希函数将覆盖多个块的块头,同时这些块头将共用一个验证字段.这个结构允许连弩挖

矿包含算力难度不同的共识组.一旦一个节点计算出验证字段,则该节点将所有其涵盖的区块索引以及相应的默克尔树的根节点的哈希值均记录到区块链中.其他节点也可根据此记录,验证其数据的真实性.最后,当区块链出现分叉时,为了保证正确性,所有区块链分支上的事务的后续原子操作均不会被执行,直到一个分叉被舍弃.

在[86]中,作者研究在 PBFT 共识协议下,通过分片的方式提高区块链系统的吞吐量.为了提高系统的可扩展性同时保证安全性,该文献将可信执行环境（Trusted Execution Environment）融入到区块链节点中.在可信执行环境中,数据的可认证性以及完整性均受保护,不被恶意程序所篡改.在共识协议达成阶段,可利用可信执行环境的节点计算结果,减少共识达成所需要的开销.在事务处理方面,作者将分布式数据库中二阶段确认与二阶段锁引入到区块链中.具体而言,当节点需要处理跨片的多个事务时,首先向事务输入所在的片发送确认请求,确认事务处理输入是否合法.在收到所有确认之后,节点进行事务的执行并将执行的结果（Commit 或者 abort）回传给所在的片.在事务执行阶段,利用二阶段锁机制,将多个事务并发处理,进而提高整个系统的吞吐量.

针对共识算法,Algorand[72]提出了基于委员会节点的共识算法,以提高区块链网络的吞吐量.在 Algorand 中,共识算法会通过可验证随机函数,随机地选择一组节点作为委员会.委员会节点会根据其所持有的区块链网络中的货币份额来分配权重,并利用权重证明来实现共识.由于委员会的选择采用随机函数生成,因而攻击者并不能确定哪一部分节点将会属于委员会.同时,Algorand 会定期替换属于委员会的节点,从而保证网络的安全性.

5.2 区块链的链下事务处理

为了提高系统的吞吐量,部分事务可从链上处理迁移到链下,即部分交易不在区块链所记录且不需要节点间达成共识.但是,由于存在不需要达成共识的事务存在,链下事务处理往往解决节点之间的可信问题.现有的工作多应用在链下的转账交易等（例如比特币）.通过建立链下的信任网络[100-102],区块链允许在链下执行特定用户之间的事务,并最终合并成一个事务存储在链上.通过这样的方式,大量的信任节点之间的交易不需要通过区块链的共识验证,从而提高整个网络的吞吐量.

[102, 103]分别基于比特币和以太坊,提出了支持比特币的 Lighting 网络和支持以太币的 Raiden 链下支付网络,来作为比特币和以太坊交易的补充.在链下支付网络中,当用户与用户确认彼此可信时,可以在他们之间建立支付通道.支付通道内的交易所产生的事务可以不需要经过链上的共识确认,而可以直接确认.在链下的支付网络中,参与的用户需提前存入一定量的比特币或以太币.当一个支付产生时,付款方需将存入的一部分金额转入收款方.因而,链下支付需保证支付的金额小于用户在网络中所剩的金额.与此同时,链下交易网络满足传递性.当用户 A 与 B、B 与 C 之间分别存在支付通道,即使 A 与 C 之间没有建立支付通道,A 依然可以通过路径 A,B,C,将一定金额通过链下交易网络,转入 C 的账户中.因此,在链下支付网络中,部分工作旨在减少支付事务的时间开销.[104]通过设计多个查找支付路线的算法,降低查找支付路线的平均时间开销.

而与 Lightning、Raiden 等网络直接对账户内的金额进行交易不同,[105]提出了 Sprites,这一利用智能合约确保事务执行的网络.通过智能合约,当多个事务存在线性关系时,Sprites 可以部分地并行执行多个事务.从而提高链下交易的执行效率.

链下网络的另一个问题是再平衡问题.在链下支付网络中,用户需先存于一定的金额于支付通道中,并且该金额只能用于该支付通道中的两个用户之间的支付.假定用户 A、B、C 三个用户两两之间均有支付通道.在某一时刻,假定三者的金额状态为 AB 通道之间 A 有 200 元,AC 通道之间 C 有 200 元.则当 A 想转账 50 元于 C 时,由于 AC 之间的支付通道中 A 的所剩金额为 0,因而,A 不能通过 AC 之间的支付通道转账于 C.一个可行的通路是 A 通过 AB 以及 BC 之间的通道转账于 C.但是实际上,用户 A 的账户金额含有 AB 之间的 200 元.金额上 A 足够支付 AC 间的转账金额.因此,支付网络中支付通道的不平衡导致支付需要访问更多的节点,才能完成支付.

为了解决再平衡的问题,[106]提出了 Revive 系统.在[106]中,作者构建了支付网络图.该图中的一个节点为一个用户.节点与节点的边为建立的支付通道.利用该图结构,一个支付网络的支付能力为图中所有支付通道可支付的金额总和.在这种情况下,最大化支付网络的支付能力可以建模为线性规划问题.但是,直接求解线性规划问题是 NP 难的问题.因此,在 Revive 中,一个支付网络的再平衡过程可以首先找到图中一个环形的结构,并在环形结构中的所有节点转移相同的金额.由于是环形结构,该过程不会减少任何用户的账户金额.基于此,作者设计了基于环形结构的再平衡算法,并提出相应的启发式优化算法.

6 区块链数据管理的未来研究趋势

区块链作为近几年被广泛讨论的技术,在数据管理方面尚有许多值得深入探索的问题.在本文的最后,我们提出一些值得进一步挖掘的针对数据管理的研究问题,希望对本领域其他研究者有所启发.

- 区块链内数据的隐私保护技术

在一个典型的区块链中,所有的数据以及事务需要获得所有节点的共识,因此所有的数据均可被网络中的节点访问.在这种情况下,需要研究数据隐私保护问题.文献[81]初步尝试了将部分数据存储于链上,而将敏感数据存储于链下的方式.在这种情况下,需要人为地将数据分割为敏感数据与非敏感数据.对需要保护的敏感数据,无法在区块链的环境下对其进行管理.因此,解决该类隐

私保护问题,从数据管理的角度,其难点在于如何对区块链中的数据进行访问控制.已有的针对数据库的访问控制方式对数据或者表设定访问权限.直接将其应用于区块链中会导致每个区块的哈希值无法与所获得的数据对应,因而用户无法验证该链中的数据是否被篡改.

- 链间事务处理优化技术

现有的多数工作考虑单一区块链并对单一区块链下的事务处理等问题提出了高效的算法.但是当一个事务需要访问多个区块链时,会产生诸多问题,包括如何保证数据的一致性、高效的并发控制算法等.文献[63]考虑了多链情况下的视图以及验证问题.现有的工作多考虑如何保证跨链事务一致性的问题.与单链相比,较少工作考虑跨链事务处理的效率.在跨链事务的执行过程中,可能会对一条链的数据进行加锁等操作.这种操作将会影响与该链相关的事务执行. 因此,仅仅考虑单链的事务或者仅考虑跨链的事务都可能降低整个系统的吞吐量.对链间事务处理效率问题的研究将着眼于,如果设计针对多链环境下的事务处理优化,以达到在保证一致性的同时,提高吞吐量的目标.

- 面向智能合约的事务处理优化技术

区块链所部署的智能合约可以对数据进行添加、修改等操作.现有的工作往往将智能合约视为一个事务处理.但是,由于智能合约执行的复杂度并不一致,且一个智能合约可以调用其他智能合约,仅将一个智能合约视为一个事务并不能完全衡量系统的性能优劣.因此,需要一个更适用于区块链的衡量标准,以比较不同区块链系统的性能.该标准的难点在于,如何设计基本的语义以及如何将智能合约与智能合约的嵌套关系同时考虑在性能衡量的指标中.

7 总结

我们从区块链数据存储、区块链事务处理、区块链查询处理、区块链可扩展性等几个方面,对区块链环境下的数据管理相关研究工作进行了较为系统的综述,分析了各种区块链数据管理方法的优点与不足,最后介绍了区块链的未来研究趋势.从数据管理的角度,区块链的现有系统与方法仍然有非常多的工作需要进行.并且在一个不可信场景下构造一个可信计算环境仍然具有非常巨大的应用前景,因此,通过本文的综述与介绍也希望有更多的研究者,尤其是数据库研究者更多地开展区块链数据管理研究工作.

References:

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf.

[2] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07).

[3] IBM Blockchain, “Enterprise blockchain solutions and services,” 2018. [Online]. Available: https://www.ibm.com/blockchain.

[4] Oracle, “Transforming the enterprise with oracle blockchain platform,” 2018. [Online]. Available: https://www.oracle.com/cloud/blockchain/.

[5] SAP, “Blockchain applications and services,” 2018. [Online]. Available: https://www.sap.com/products/leonardo/blockchain.html.

[6] Huawei, “Huawei blockchain whitepaper, toward a trusted digital world,” 2018. [Online]. Available: https://static.huaweicloud.com/upload/iles/pdf/20180416/2018041614245061761.pdf.

[7] C. Berger, B. Penzenstadler, and O. Drögehorn, “On using blockchains for safety-critical systems,” in Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems, ser. SEsCPS ’18. New York, NY, USA: ACM, 2018, pp. 30–36.

[8] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, “Medrec: Using blockchain for medical data access and permission management,” in 2nd International Conference on Open and Big Data, OBD 2016, Vienna, Austria, August 22-24, 2016, pp. 25–30.

[9] S. Wu and J. Du, “Electronic medical record security sharing model based on blockchain,” in Proceedings of the 3rd International Conference on Cryptography, Security and Privacy, ser. ICCSP’19. New York, NY, USA: ACM, 2019, pp. 13–17.

[10] Intellectual property blockchain platform, 2018. [Online]. Available: https://www.bernstein.io/.

[11] S. Underwood, “Blockchain beyond bitcoin,” Commun. ACM, vol. 59, no. 11, pp. 15–17, Oct. 2016.

[12] V. Zakhary, M. J. Amiri, S. Maiyya, D. Agrawal, and A. El Abbadi, “Towards global asset management in blockchain systems,” CoRR, vol. abs/1905.09359, 2019.

[13] G. Ateniese, A. Faonio, B. Magri, and B. de Medeiros, “Certified bitcoins,” in Applied Cryptography and Network Security, 2014, pp. 80–96.

[14] J. Garay, A. Kiayias, and N. Leonardos, “The bitcoin backbone protocol: Analysis and applications,” in Advances in Cryptology EUROCRYPT, 2015, pp. 281–310.

[15] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-NG: A scalable blockchain protocol,” in *USENIX NSDI*, 2016, pp. 45–59.

[16] G. Pirlea and I. Sergey, “Mechanising blockchain consensus,” in ACM SIGPLAN Int’l Conf. Certified Programs and Proofs. ACM, 2018, pp. 78–90.

[17] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. van Moorsel, “Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing,” in ACM CCS. ACM, 2017, pp. 211–227.

[18] J. Camenisch, M. Drijvers, and M. Dubovitskaya, “Practical unsecure delegatable credentials with attributes and their application to blockchain,” in ACM CCS. ACM, 2017, pp. 683–699.

[19] R. Zhang, R. Xue, and L. Liu, “Security and privacy on blockchain,” CoRR, vol. abs/1903.07602, 2019.

[20] L. Kiffer, R. Rajaraman, and a. shelat, "A better method to analyze blockchain consistency," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 729–744.

[21] S. Wang, T. T. A. Dinh, Q. Lin, Z. Xie, M. Zhang, Q. Cai, G. Chen, B. C. Ooi, and P. Ruan, "Forkbase: An efficient storage engine for blockchain and forkable applications," Proc. VLDB Endow., vol. 11, no. 10, pp. 1137–1150, Jun. 2018.

[22] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017, pp. 1085–1100.

[23] L. Lamport, "Paxos made simple," ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001), pp. 51–58, December 2001.

[24] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference, ser. USENIX ATC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 305–320.

[25] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security, ser. CMS'99. Deventer, the Netherlands, Kluwer, B.V., 1999, pp. 258–272.

[26] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in Proceedings of the Third Symposium on Operating Systems Design and Implementation, ser. OSDI'99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.

[27] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," IEEE Trans. Knowl. Data Eng., vol. 30, no. 7, pp. 1366–1385, 2018.

[28] S. Maiyya, V. Zakhary, M. J. Amiri, D. Agrawal, and A. El Abbadi, "Database and distributed computing foundations of blockchains," in Proceedings of the 2019 International Conference on Management of Data, ser. SIGMOD '19. New York, NY, USA: ACM, 2019, pp. 2036–2041.

[29] F. Baig and F. Wang, "Blockchain enabled distributed data management - A vision," in 35th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2019, Macao, China, April 8-12, 2019, pp. 28–30.

[30] C. Mohan, "State of public and private blockchains: Myths and reality," in Proceedings of the 2019 International Conference on Management of Data, ser. SIGMOD'19. New York, NY, USA: ACM, 2019, pp. 404–411.

[31] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in Proceedings of the Thirteenth EuroSys Conference, ser. EuroSys '18. New York, NY, USA: ACM, 2018, pp. 30:1–30:15.

[32] P. A. Alsberg and J. D. Day, "A principle for resilient sharing of distributed resources," in Proceedings of the 2nd International Conference on Software Engineering, ser. ICSE '76. Los Alamitos, CA, USA: IEEE Computer Society Press, 1976, pp. 562–570.

[33] M. R. Stonebraker, "Distributed systems, vol. ii: Distributed database systems," W. W. Chu, Ed. Norwood, MA, USA: Artech House, Inc., 1986, ch. Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES, pp. 193–199.

[34] E. Cecchet, G. Candea, and A. Ailamaki, "Middleware-based database replication: The gaps between theory and practice," in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD'08. New York, NY, USA: ACM, 2008, pp. 739–752.

[35] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso, "Understanding replication in databases and distributed systems," in Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000), ser. ICDCS'00. Washington, DC, USA: IEEE Computer Society, 2000.

[36] R. C. Merkle, "A certified digital signature," in Proceedings on Advances in Cryptology, ser. CRYPTO '89. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 218–238.

[37] I. Puddu, A. Dmitrienko, and S. Capkun, "How to forget without hard forks," *IACR Cryptology ePrint Archive,* vol. 2017, p. 106, 2017.

[38] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *Open Problems in Network Security*, J. Camenisch and D. Kesdoğan, Eds. Cham: Springer International Publishing, 2016, pp. 112–125.

[39] J. R. Douceur, "The sybil attack," in Revised Papers from the First International Workshop on Peer-to-Peer Systems, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 251–260.

[40] M. Rosenfeld, "Analysis of hashrate-based double spending," CoRR, vol. abs/1402.2009, 2014.

[41] G. Naumenko, G. Maxwell, P. Wuille, S. Fedorova, and I. Beschastnikh, "Bandwidth efficient transaction relay for bitcoin," CoRR, vol. abs/1905.10518, 2019.

[42] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 17–30.

[43] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 931–948.

[44] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in Proceedings of the 25th USENIX Conference on Security Symposium, ser. SEC'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 279–296.

[45] J. Kwon, "Tendermint: Consensus without mining," 2014.

[46] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," Commun. ACM, vol. 61, no. 7, pp. 95–102, Jun. 2018.

[47] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on bitcoin’s peer-to-peer network,” in Proceedings of the 24th USENIX Conference on Security Symposium, ser. SEC’15. Berkeley, CA, USA: USENIX Association, 2015, pp. 129–144.

[48] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making byzantine fault tolerant systems tolerate byzantine faults,” in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI’09. Berkeley, CA, USA: USENIX Association, 2009, pp. 153–168.

[49] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, “Consensus in the age of blockchains,” CoRR, vol. abs/1711.03936, 2017.

[50] C. Cachin and M. Vukolic, “Blockchain consensus protocols in the wild,” CoRR, vol. abs/1707.01873, 2017.

[51] N. Szabo, “Formalizing and securing relationships on public networks,” First Monday, vol.2, no. 9, 1997.

[52] V. Buterin, “Ethereum: A next-generation smart contract and decentralized application platform,” 2019, accessed: 2018-08-05.

[53] “Leveldb” https://github.com/google/leveldb/, accessed July 20th, 2019.

[54] “Rocksdb” https://rocksdb.org/, accessed July 20th, 2019.

[55] M. El-Hindi, M. Heyden, C. Binnig, R. Ramamurthy, A. Arasu, and D. Kossmann, “Blockchaindb - towards a shared database on blockchains,” in Proceedings of the 2019 International Conference on Management of Data, ser. SIGMOD ’19. New York, NY, USA: ACM, 2019, pp. 1905–1908.

[56] M. El-Hindi, B. Carsten, A. Arasu, D. Kossmann, and R. Ramamurthy, “Blockchaindb - a shared database on blockchains,” vol. 12, no. 11. VLDB Endowment, Jun. 2019, pp. 1597–1609.

[57] R. K. Raman and L. R. Varshney, “Dynamic distributed storage for scaling blockchains,” CoRR, vol. abs/1711.07617, 2017.

[58] G. Ayoade, V. Karande, L. Khan, and K. W. Hamlen, “Decentralized iot data management using blockchain and trusted execution environment,” in 2018 IEEE International Conference on Information Reuse and Integration, IRI 2018, Salt Lake City, UT, USA, July 6-9, 2018, 2018, pp. 15–22.

[59] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, “Blockchain based data integrity service framework for iot data,” in 2017 IEEE International Conference on Web Services, ICWS 2017, Honolulu, HI, USA, June 25-30, 2017, 2017, pp. 468–475.

[60] J. Eberhardt and S. Tai, “On or off the blockchain? insights on off-chaining computation and data,” in Service-Oriented and Cloud Computing - 6th IFIP WG 2.14 European Conference, ESOCC 2017, Oslo, Norway, September 27-29, 2017, Proceedings, 2017, pp. 3–15

[61] J. Eberhardt and J. Heiss, “Off-chaining models and approaches to off-chain computations,” in Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, ser. SERIAL’18. New York, NY, USA: ACM, 2018, pp. 7–12.

[62] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. M. Johnson, A. Juels, A. Miller, and D. Song, “Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution,” CoRR, vol. abs/1804.05141, 2018.

[63] M. Amiri Javad, D. Agrawal, and A. E. Abbadi, “Caper: A cross-application permissioned blockchain,” vol. 12, no. 11. VLDB Endowment, Jun. 2019, pp. 1385–1397.

[64] A. Sharma, F. M. Schuhknecht, D. Agrawal, and J. Dittrich, “Blurring the lines between blockchains and database systems: The case of hyperledger fabric,” in Proceedings of the 2019 International Conference on Management of Data, ser. SIGMOD ’19. New York, NY, USA: ACM, 2019, pp. 105–122.

[65] M. J. Amiri, D. Agrawal, and A. El Abbadi, “Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems,” CoRR, vol. abs/1902.01457, 2019.

[66] H. Javaid, C. Hu, and G. J. Brebner, “Optimizing validation phase of hyperledger fabric,” CoRR, vol. abs/1907.08367, 2019.

[67] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, “Adding concurrency to smart contracts,” in Proceedings of the ACM Symposium on Principles of Distributed Computing, ser. PODC ’17. New York, NY, USA: ACM, 2017, pp. 303–312.

[68] P. S. Anjana, S. Kumari, S. Peri, S. Rathor, and A. Somani, “An efficient framework for optimistic concurrent execution of smart contracts,” in 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2019, Pavia, Italy, February 13-15, 2019, 2019, pp. 83–92.

[69] A. Bessani, J. a. Sousa, and M. Vukoli´c, “A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform,” in Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, ser. SERIAL ’17. New York, NY, USA: ACM, 2017, pp. 6:1–6:2.

[70] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, and P. Jayachandran, “Blockchain meets database: Design and implementation of a blockchain relational database,” CoRR, vol. abs/1903.01919, 2019.

[71] M. J. Cahill, U.R¨ohm, and A.D. Fekete, “Serializable isolation for snapshot databases,” in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’08. New York, NY, USA: ACM, 2008, pp. 729–738.

[72] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in Proceedings of the 26th Symposium on Operating Systems Principles, ser. SOSP ’17. New York, NY, USA: ACM, 2017, pp. 51–68.

[73] G. Weikum and G. Vossen, “Transactional Information Systems: Theory, Algorithms, and the Prac- tice of Concurrency Control and Recovery”. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[74] M. Herlihy, “Atomic cross-chain swaps,” in Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, ser. PODC ’18. New York, NY, USA: ACM, 2018, pp. 245–254.

[75] V. Zakhary, D. Agrawal, and A. El Abbadi, “Atomic commitment across blockchains,” CoRR, vol. abs/1905.02847, 2019.

[76] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, “Untrusted business process monitoring and execution using blockchain,” in Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings, 2016, pp. 329–347.

[77] E. Androulaki, C. Cachin, A. D. Caro, and E. Kokoris-Kogias, “Channels: Horizontal scaling and confidentiality on permissioned blockchains,” in Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I, 2018, pp. 111–131. [Online]. Available: https://doi.org/10.1007/978-3-319- 99073-6 6

[78] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Tim´on, and P. Wuille, “Enabling blockchain innovations with pegged sidechains,” 2014.

[79] V. Buterin, “Chainsql,” 2019, accessed: 2018-08-05. [Online]. Available: http://www.chainsql.net/

[80] “Bigchaindb: The blockchain database,” 2019, accessed: 2019-07-27.

[81] Y. Zhu, Z. Zhang, C. Jin, A. Zhou, and Y. Yan, “SEBDB: semantics empowered blockchain database,” in 35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019, 2019, pp. 1820–1831.

[82] P. Ruan, G. Chen, T. T. A. Dinh, Q. Lin, B. C. Ooi, and M. Zhang, “Fine-grained, secure and efficient data provenance on blockchain systems,” Proc. VLDB Endow., vol. 12, no. 9, pp. 975–988, May 2019.

[83] A. Ramachandran and M. Kantarcioglu, “Using blockchain and smart contracts for secure data provenance management,” CoRR, vol. abs/1709.10000, 2017.

[84] C. Xu, C. Zhang, and J. Xu, “vchain: Enabling verifiable boolean range queries over blockchain databases,” in Proceedings of the 2019 International Conference on Management of Data, ser. SIGMOD ’19. New York, NY, USA: ACM, 2019, pp. 141–158.

[85] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, “GEM2-Tree: A gas-efficient structure for authen- ticated range queries in blockchain,” in Proceedings of the 35th IEEE International Conference on Data Engineering, Macau SAR, China, Apr. 2019, pp. 842–853.

[86] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, “Towards scaling blockchain systems via sharding,” in Proceedings of the 2019 International Conference on Management of Data, ser. SIGMOD ’19. New York, NY, USA: ACM, 2019, pp. 123–140. [Online].

[87] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “Omniledger: A secure, scale-out, decentralized ledger via sharding,” 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598, 2018.

[88] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, “Chainspace: A sharded smart contracts platform,” in 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018, 2018.

[89] J. Wang and H. Wang, “Monoxide: Scale out blockchain with asynchronous consensus zones,” in Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI’19. Berkeley, CA, USA: USENIX Association, 2019, pp. 95–112.

[90] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, “On scaling decentralized blockchains (A position paper),” in Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers, 2016, pp. 106–125.

[91] Z. Istv´an, A. Sorniotti, and M. Vukolic, “Streamchain: Do blockchains need blocks?” CoRR, vol. abs/1808.08406, 2018.

[92] S. Popov, “The tangle (2016),” URL http://www.iotatoken.com, 2018.

[93] L. Baird, “The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance,” 2016.

[94] Z. Ren, K. Cong, J. Pouwelse, and Z. Erkin, “Implicit consensus: Blockchain with unbounded throughput,” CoRR, vol. abs/1705.11046, 2017.

[95] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, “Dynamic authenticated index structures for outsourced databases,” in Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’06. New York, NY, USA: ACM, 2006, pp. 121–132.

[96] H. Pang and K. L. Tan, “Authenticating query results in edge computing,” in Proceedings of the 20th International Conference on Data Engineering, ser. ICDE’04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 560-571.

[97] Q. Chen, H. Hu, and J. Xu, “Authenticated online data integration services,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’15. New York, NY, USA: ACM, 2015, pp. 167–181.

[98] Bitcoin, “Running a full node,” 2018, [Online]. Available: https://bitcoin.org/en/full-node.

[99] J. Gehrke, L. Allen, P. Antonopoulos, A. Arasu, J. Hammer, J. Hunter, R. Kaushik, D. Kossmann, R. Ramamurthy, S. T. V. Setty, J. Szymaszek, A. van Renen, J. Lee, and R. Venkatesan, “Veritas: Shared verifiable databases and tables in the cloud,” in CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings, 2019.

[100] C. Decker and R. Wattenhofer, “A fast and scalable payment network with bitcoin duplex micropayment channels,” in Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings, 2015, pp. 3–18.

[101] M. Green and I. Miers, “Bolt: Anonymous payment channels for decentralized currencies,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS ’17. New York, NY, USA: ACM, 2017, pp. 473–489.

[102] J. Poon and T. Dryja, “The bitcoin lightning network,” 2019, [Online]. Available: https://lightning.network/.

[103] “Raiden network,” 2019, [Online]. Available: https://raiden.network/.

[104] P. Prihodko, S. Zhigulin, M. Sahno, A. Ostrovskiy, and O. Osuntokun, “Flare : An approach to routing in lightning network white paper,” 2016.

[105] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, “Sprites: Payment channels that go faster than lightning,” CoRR, vol. abs/1702.05812, 2017

[106] R. Khalil and A. Gervais, “Revive: Rebalancing off-blockchain payment networks,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and CommunicationsSecurity, ser. CCS ’17. New York, NY, USA: ACM, 2017, pp. 439–453.

[107] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “Fastfabric: Scaling hyperledger fabric to 20, 000 transactions per second,” CoRR, vol. abs/1901.00910, 2019.

附中文参考文献:

[108] 贾大宇, 信俊昌,王之琼,郭薇,王国仁.存储容量可拓展的区块链系统的高效查询模型.软件学报,2019.

[109] 刘敖迪,杜学绘,王娜,李少卓.区块链技术及其在信息安全领域的研究进展.软件学报,2018.

[110] 邵奇峰,金澈清,张召,钱卫宁,周傲英.区块链技术:架构及进展.计算机学报,2017.

[111] 王千阁,何蒲,聂铁铮,申德荣,于戈.区块链系统的数据存储与查询技术综述.计算机科学,2018.

[112] 范吉立,李晓华,聂铁铮,于戈.区块链系统中智能合约技术综述.计算机科学,2018
[113] 刘敖迪,杜学绘,王娜,李少卓.基于区块链的大数据访问控制机制.软件学报,2019.
[114] 贺海武,延安,陈泽华.基于区块链的智能合约技术与应用综述.计算机研究与发展,2018.
[115] 钱卫宁,邵奇峰,朱燕超,金澈清,周傲英.区块链与可信数据管理：问题与方法.软件学报,2018
[116] 蔡磊,朱燕超,郭庆兴,张召,金澈清.面向区块链的高效物化视图维护与可信查询,NDBC,2019.
[117] 刘耀宗,刘云恒.基于区块链的 RFID 大数据安全溯源模型,计算机科学,2018.