软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(10):4607−4635 [doi: 10.13328/j.cnki.jos.006677]

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

分布式账本系统性能优化技术综述*

石 晶[1,3], 张 奥[2,3], 白晓颖[3], 蔡华谦[1], 刘譞哲[1]

[1](北京大学 信息科学技术学院, 北京 100871)

[2](清华大学 计算机科学与技术系, 北京 100084)

[3](北京大数据先进技术研究院, 北京 100091)

通信作者: 白晓颖, E-mail: baixy@aibd.ac.cn; 刘譞哲, Email: xzl@pku.edu.cn

摘 要: 分布式账本作为分布式数据管理的体系架构, 通常在多节点之间通过共识机制来共同维护数据记录, 可将数据所有权、传播过程、交易链条等相关信息完整全面地记录在分布的账本中, 并在数据产生、流动的整个生命周期中, 保证数据的不可篡改、不可抵赖, 为确权、维权、审计提供背书. 区块链是一种典型实现. 随着数字货币、数据资产交易等数字经济新应用的发展, 分布式账本技术得到了越来越广泛的关注, 但系统性能是其大规模落地应用的一个主要瓶颈, 账本性能优化成为产业界和学术界一个研究热点. 从账本体系结构、数据结构、共识机制和消息通讯 4 个方面, 系统地调研分析了分布式账本性能优化的主要方法、关键技术和代表性的解决方案.

关键词: 分布式账本; 区块链; 性能优化

中图法分类号: TP311

中文引用格式: 石晶, 张奥, 白晓颖, 蔡华谦, 刘譞哲. 分布式账本系统性能优化技术综述. 软件学报, 2023, 34(10): 4607–4635. http://www.jos.org.cn/1000-9825/6677.htm

英文引用格式: Shi J, Zhang A, Bai XY, Cai HQ, Liu XZ. Survey on Performance Optimization Technologies of Distributed Ledger System. Ruan Jian Xue Bao/Journal of Software, 2023, 34(10): 4607–4635 (in Chinese). http://www.jos.org.cn/1000-9825/6677.htm

Survey on Performance Optimization Technologies of Distributed Ledger System

SHI Jing[1,3], ZHANG Ao[2,3], BAI Xiao-Ying[3], CAI Hua-Qian[1], LIU Xuan-Zhe[1]

[1](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[3](Advanced Institute of Big Data, Beijing 100091, China)

Abstract: Distributed ledger (DL), as a distributed data management architecture, maintains data records (the ledgers) across distributed nodes based on consensus mechanisms and protocols. It can comprehensively record all information of data ownership, transmission, and trading chains in distributed ledgers. Additionally, data will be not tampered and denied throughout the life cycle of data production and transactions, providing an endorsement for data rights confirmation, protection, and audit. Blockchain is a typical implementation of DL systems. With the emerging digital economy applications including digital currency and data asset trading, DL technologies receive increasingly widespread attention. However, system performance is one of the key technical bottlenecks for large-scale application of DL systems, and ledger performance optimization has become a focus of the academia and industry. The study investigates the methods, technologies, and typical solutions of DL performance optimization from four perspectives of system architecture, ledger data structure, consensus mechanism, and message communication.

Key words: distributed ledger; blockchain; performance optimization

---

* 基金项目: 北京市科技计划 (Z201100007720010)
收稿时间: 2021-09-10; 修改时间: 2021-10-24, 2021-12-21, 2022-01-27; 采用时间: 2022-03-14; jos 在线出版时间: 2022-05-24
CNKI 网络首发时间: 2023-04-06

数字化记账始于 20 世纪 60 年代. 传统的数字化账本系统是通过中心化数据管理系统来记录和维护账户数据. 性能高, 但是需要高度依赖银行等中心机构的可信度来保证数据的正确性和安全性. 一旦中心机构出现伪造数据等信任问题, 则会对系统造成无法挽回的重大损失. 分布式账本系统是基于网络的去中心化数据存储管理系统, 不依赖于中心化机构的可信度, 在对等节点之间维护多个账本副本, 节点之间通过分布式共识机制来共同记录、更新和验证数据, 使得账本的数据记录不易伪造、难以篡改.

区块链是分布式账本的一种典型实现. 2008 年, 署名中本聪的作者发表了“比特币: 点对点的电子现金系统”[1]一文, 提出了区块链的分布式账本结构, 采用多节点共识和密码学机制, 保证了交易记录的持久性、安全性和不可篡改性. 2014 年, Buterin 等人提出以太坊[2], 以太坊将智能合约应用于分布式账本平台之上, 采用账本记录合约数据, 通过 Solidity 语言来灵活构建合约, 实现不同领域的去中心化应用. 2015 年, 在 Linux 基金会的支持下, Fabric 开源项目被提出[3]. 该项目在原有区块链技术的基础上提供了完备的权限和安全控制设计, 实现了企业级的许可分布式账本, 使得若干组织或社区可以共同维护账本数据.

分布式账本作为分布式数据管理的体系架构, 可将数据所有权、传播过程、交易链条等相关信息完整全面地记录在分布的账本中, 并在数据产生、流动的整个生命周期中, 保证数据的不可篡改、不可抵赖, 为确权、维权、审计提供背书. 目前, 随着数字货币、数据资产交易等数字经济新应用的发展, 分布式账本技术得到了越来越广泛的关注, 其应用不仅局限于金融领域, 而是可以延伸到物联网、关键数据存证、智能工业生产、供应链管理以及数据资产的确权等更加广泛的领域[4,5].

然而, 至今为止, 分布式账本的系统性能仍是其大规模应用的一个主要技术瓶颈. 以典型的区块链系统为例, 可以看到其性能指标与实际应用需求尚存在较大差距. 系统每秒交易处理数及交易确认时间是两个主要性能度量指标[6−9].

(1) 交易处理数 (transactions per second, *TPS*), 即每秒交易的数量. 假设系统区块容量为 *blockSize*, 单笔交易容量为 *txSize*. 若在 $[t_i,t_j)$ 时间段中, 系统处理的有效区块数为 $blockNum_{ij}$, 如公式 (1), 公式 (2) 所示, 由此可得在 $[t_i,t_j)$ 时间中的交易处理数量 $txNum_{ij}$, 以及系统的交易处理数量 *TPS*.

$$txNum_{ij}=blockNum_{ij}\times\frac{blockSize}{txSize} \tag{1}$$

$$TPS=\frac{txNum_{ij}}{t_j-t_i} \tag{2}$$

比特币的平均区块容量为 1 MB[1], 平均单笔交易容量约为 250 B, 则每个区块可容纳 4 194 笔交易. 比特币平均每 10 分钟生成一个区块, 由公式 (2) 可得, 其每秒交易量仅为 7 笔.

(2) 交易确认时间 (transaction confirmation time, *TxCT*), 即从交易发起到交易被确认的时间. 在区块链系统中, 交易确认时间包括等待上链和最终确认的时间. 在高并发情况下, 账本容易出现分叉, 而链式的账本结构仅将一条分叉链作为有效主链. 为了处理分叉, 账本系统根据后续账本的最长子链、最重子树等方式选择主链. 系统往往需要等待后序多个区块后才得以被确认上链, 使得交易确认时间较长, 可达到平均出块时间的数倍.

假设交易于 $t_{i\text{Input}}$ 时间发起, 系统于 $t_{i\text{Confirmed}}$ 时间完成交易确认, 可得单笔交易确认时间. 由于账本分叉, 系统往往会等待后续 n 个区块后, 才可以确认该笔交易有效上链. 若区块的出块时间为 $avgBlockTime$, 则系统的交易确认时间如公式 (3) 所示.

$$TxCT=t_{i\text{Confirmed}}-t_{i\text{Input}}=avgBlockTime\times n \tag{3}$$

在比特币中, 根据概率和历史数据, 链上交易需要等待 6 个区块确认. 其出块时间为 10 min, 这意味着每笔交易的最终确认时间为 60 min.

传统的金融交易系统性能远高于比特币、以太坊等区块链系统. 例如跨境金融交易系统 Paypal 每秒可以处理 500 笔, VISA 每秒可以处理 4 000 笔[10], 并且能够实现单笔交易秒级确认. 因此, 分布式账本系统尚难以支撑大规模的海量的分布式数据管理和交易.

针对这个问题, 分布式账本性能优化成为一个重要的研究方向, 已有多个研究工作分别从共识算法、扩展性、性能与安全性的权衡等方面, 分析了相关研究现状.

(1) 在共识算法方面, Lepore 等人[11]对 PoW, PoS, Pure PoS 这 3 种主流共识协议进行性能分析, 在共识类别、延迟、吞吐量等指标对比了上述 3 种主链共识协议, 研究发现 Pure PoS 共识的吞吐量和扩展性较好且交易延迟时间较短, 基于许可链的 PoS 共识相比非许可链的 PoS、PoW 共识, 具有较好的吞吐量和扩展性. 张彭奕等人[12]分析了性能、资源、能耗对区块链共识效率的影响, 并定义了共识算法效能的概念, 说明了当共识算法的资源花费小、性能较好、能耗较低时, 该共识算法效能越高. 该工作从公有链与联盟链两方面对其共识的效能优化方法进行总结, 并指出多链、跨链与 BaaS 场景下资源共享问题中的效能优化需要结合影响共识算法效能的 3 个要素, 实现高效的资源共享.

(2) 在可扩展性方面, Zhou 等人[13]将解决系统可扩展性的方法分为 3 类, 分别是区块链底层 (layer 0)、链上 (layer 1)、链下 (layer 2), 重点从链上和链下进行讨论. 在链上优化中, 该工作从块数据、共识、分片、DAG 的角度总结了性能和存储可扩展性优化的项目和方法; 在链下优化中, 将优化方法分为支付通道、侧链、链下计算和跨链 4 类, 分别进行讨论其机制和相关项目. 研究指出底层消息传播的协议的优化、链上区块的压缩、分片交易划分和跨片的完善、和基于主侧链的链下扩展方案会进一步提升系统性能. 毛志来等人[14]将区块链系统抽象为交易、区块、共识 3 个方面, 分析了网络规模、区块大小等影响区块链性能扩展的主要因素, 总结了提升区块大小、提高出块频率、隔离见证、块压缩等经典的性能扩展机制, 并探讨了分片、跨链等新型的性能扩展机制, 最后指出了扩展区块链系统应结合应用场景, 将区块链的性能扩展机制与其带来的运维成本权衡, 从而选择符合实际应用需求的扩展方案.

(3) 在系统性能与安全性的权衡方面, Zheng 等人[7]调研了区块链的性能和安全问题, 结合链类型、拜占庭容错、链上链下数据存储以及交易存储等技术, 讨论性能和安全问题背后的架构选择问题, 比较了 Ethereum、Parity、HyperLedger 等项目的吞吐量、延迟、容错节点数等指标, 还讨论了区块链在金融、物联网等领域产业应用所面临的性能和安全性挑战. 研究指出并行化是区块链性能优化的重要研究方向, 并指出设计性能基准测试和评估工具的重要性.

以比特币系统为代表的分布式账本存在性能瓶颈问题. 近年来, 不同的性能优化技术被陆续提出, 如图 1 所示, 如侧链将主链扩展, 实现了多链的互联互通; 分片技术以节点为单位进行分片划分, 极大提升了系统的并行处理能力; 见证人共识减少系统中的共识参与者, 提升了系统的交易处理能力. 本文采用软件系统的视角, 分析了分布式账本性能问题, 分别从账本系统架构设计、账本数据结构设计、账本共识机制以及账本消息通讯 4 个方面, 系统地调研和分析了影响分布式账本性能的主要设计因素. 图 2 展示了本文中分布式账本性能优化方法的分类结构.

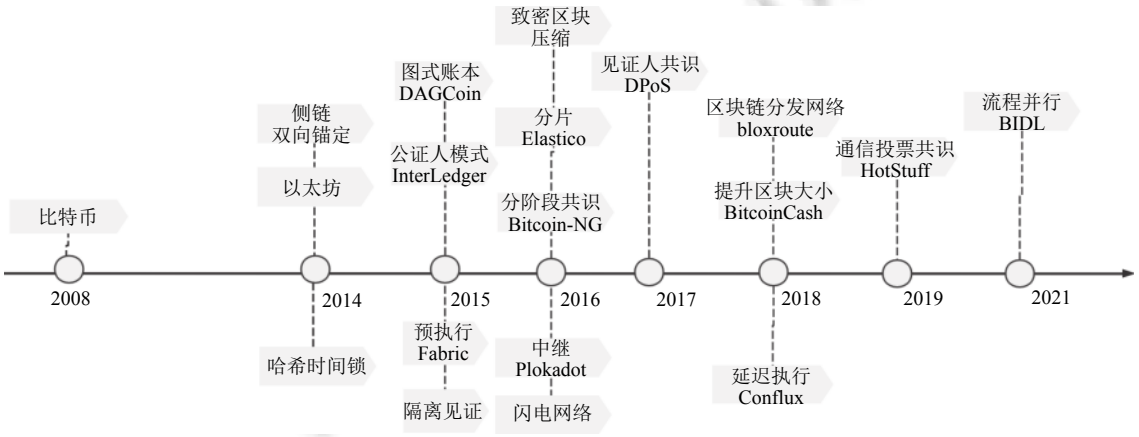

图 1 分布式账本性能优化技术发展

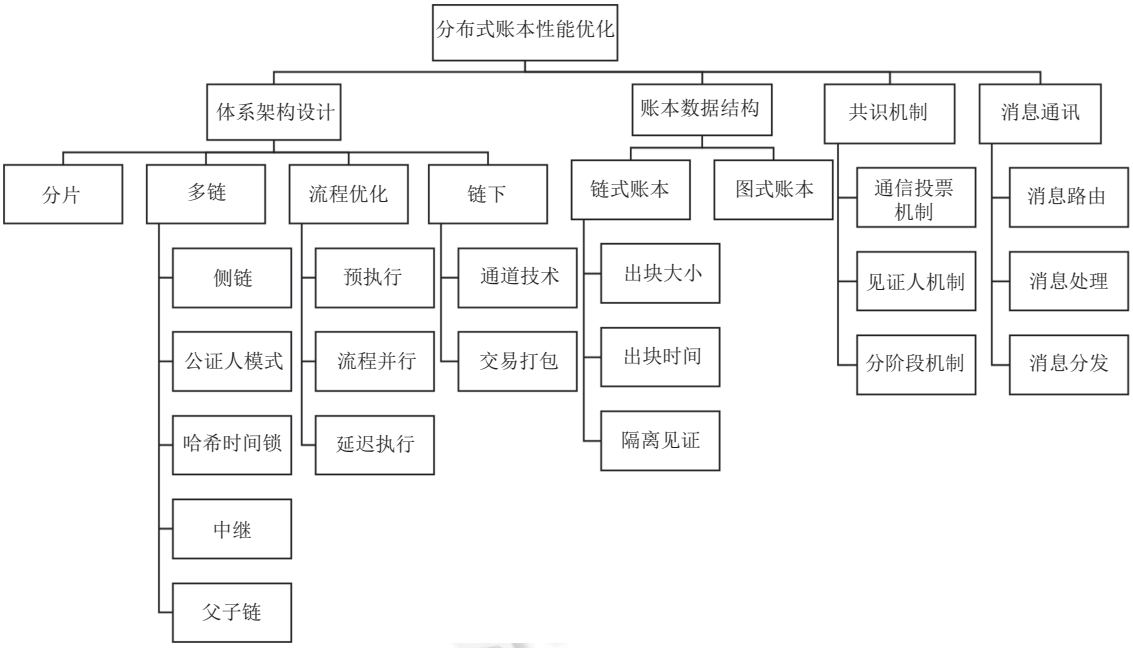

图 2 分布式账本性能优化方法分类

1 体系架构设计

本节总结了分布式账本架构设计中的典型性能优化策略. 其中, 分片将链上若干节点划分为委员会, 提高系统的并行度; 多链采用连接技术, 在链上实现了多条同构、异构链的连接, 提升系统的可扩展性; 通道技术和交易打包将负载转移至链下进行计算, 从而减轻链上负载.

1.1 分 片

分片是指将分布式账本系统中的节点按照一定机制划分为若干集合, 通常一个集合被称为一个委员会. 系统以委员会为单位来对交易进行处理.

分片的思想可以追溯到分布式数据库, 其将数据划分成若干数据子集合, 并将数据子集合分发到网络中不同的数据库节点. 如图 3 所示, 2016 年, Elastico 采用将分片技术应用到区块链数据的存储管理[15]. 先后出现了基于对象的分片 (Chainspace[16])、全分片 (RapidChain[17])、基于账户地址的分片 (Monoxide[18]) 等技术. 不同于传统的分布式数据库, 在分片账本中各委员会可以验证、执行其对应的交易, 必要时, 进行跨片通信, 保证系统统一的全局状态, 共同维护账本数据结构.

基于分片的分布式账本系统通常在 3 个层面采用分片技术: 网络分片、交易分片和状态分片[19]. 网络分片是将参与 P2P 网络的节点划分为若干委员会, 并将其作为交易处理的最小单位. 交易分片是将交易映射到对应委员会中, 并以委员会为单位执行交易. 状态分片是将全局状态进行划分成若干子状态, 每个子状态由其对应的分片来维护.

分片对于系统性能提升主要体现在两个方面: 一是多委员会并行处理交易, 提升了系统处理效率; 二是分片后, 仅在片内委员会节点之间共识, 参与共识节点数量减少, 通信、计算等共识资源开销减少. 但分片机制会带来额外管理开销, 如委员会划分、片间通信、分片重构等, 对系统性能产生影响. 如图 4 所示, 分片系统主要包括节点分片、交易分发、片内共识、跨片处理以及分片重组 5 个阶段.

(1) 节点分片: 系统对节点身份认证, 建立分组. 每个分片的成员节点组成分片委员会, 即委员会机制. 每次节点的分片结果会持续一个固定时间, 即一个 epoch.

(2) 交易分发: 系统按照交易到委员会的映射规则来分发交易.

(3) 片内共识: 委员会内部对交易形成片内共识. 分片需要每隔一段时间, 即 epoch, 进行分片重组. 为了保证分片的快速重组, 分片内部通常采用传统分布式共识协议, 如 PBFT, ByzCoinX[20]等, 快速确认交易, 而较少采用效率较低的 PoW 等协议.

(4) 跨片处理: 对于涉及多个分片的交易, 系统引入跨片处理机制, 保证跨片交易的一致性.

(5) 分片重组: 每个 epoch 结束, 系统节点需要进行分片重组, 避免恶意节点涌入某一分片, 以保证安全性.

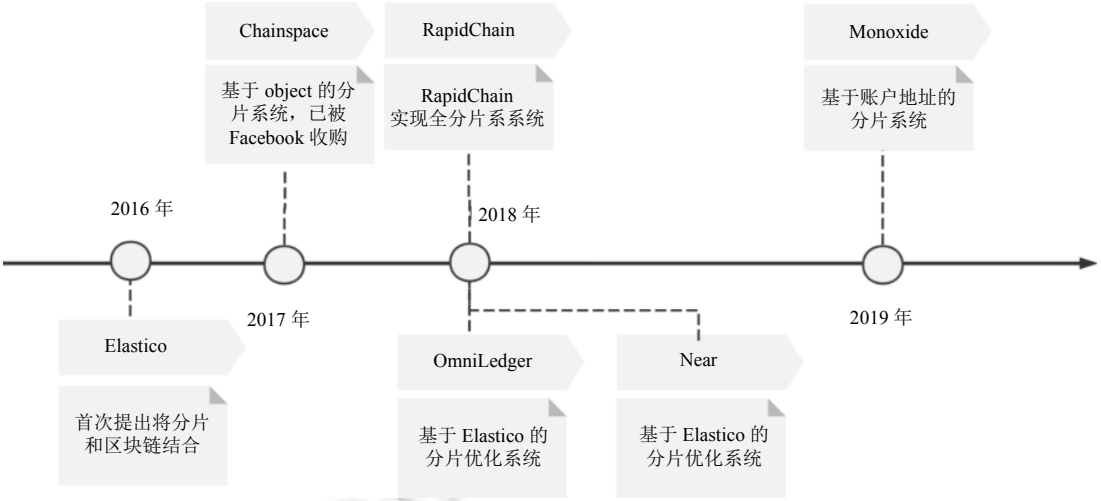

图 3 分片技术发展

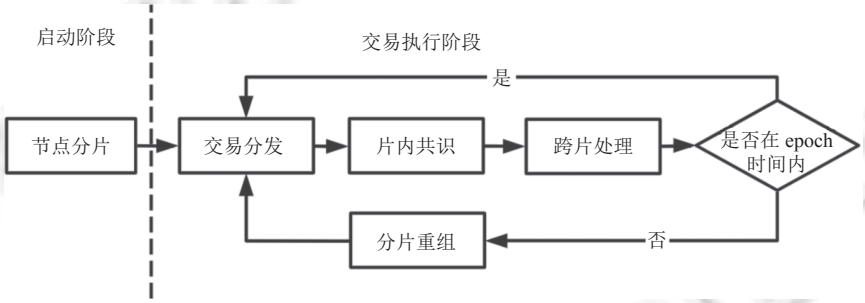

图 4 分片流程图

1.1.1 委员会机制

每个分片对应一个委员会, 负责处理片内交易. 系统中多个委员会可以并行处理交易, 提升账本系统吞吐量. 委员会的产生机制, 即节点分片机制, 是性能优化的一个关键环节. 目前主要有 hash 位和随机函数两种方式.

(1) 基于 hash 位的委员会机制

每个节点通过工作量证明, 形成各自的 hash 串. 系统截取 hash 串中固定位置的 s 位 hash bit, 具有相同的 s 位 hash bit 的节点成为一个逻辑上的委员会. 同一个委员会的节点需要相互通信, 确认身份后, 由此产生的 2^s 个委员会正式生效[15].

委员会节点之间通过广播机制进行身份确认, 其通信的复杂度是 $O(n^2)$, 网络开销较大. Elastico[15]采用了基于目录节点的通讯方式来减少通信开销, 即令前若干位计算出 PoW 的节点作为目录节点, 拥有全局视角, 负责广播委员会的成员信息, 将通信复杂度降低至 $O(n)$.

(2) 基于随机函数的委员会机制

OmniLedger[20]的研究者提出基于可验证随机函数的委员会机制. 系统通过随机函数生成随机种子, 产生节点

的随机排列, 并划分节点至若干个桶中, 同一桶中的节点构成委员会. 基于随机函数委员会机制的处理过程如下:

① 系统选举产生领导节点.

② 领导节点根据随机函数, 生成随机种子, 并广播至全网节点.

③ 其他节点根据随机种子, 生成 $\pi e(1\ldots n)$, 对应全网节点的排列, 并将其划分至 m 个桶, 同一桶中的节点构成委员会.

系统往往会采用身份链的方式来维护全局的节点身份信息, 保证全网节点身份信息的一致性, 不需要委员会相互通信, 节点在生成随机排列, 将其划分到桶中后, 可以得到全局分片划分结果, 从而减少了委员会节点相互身份确认的通信开销. 基于随机函数的委员会机制依赖于领导节点. 分布式环境下领导节点的选举会产生计算和通信资源, 特别是在节点数量较多的情况下, 会影响委员会机制的效率.

上述两种委员会机制都需要设定其划分委员会的数量 n. 当 n 设置过于小, 委员会数量较少. 当整个网络节点数量激增, 每个委员会内的节点数量会过多, 导致每个委员会的交易处理效率较低; 反之, 当 n 设置过大, 委员会数量较多, 节点数量不变的情况下, 片内节点数量较少, 诚实节点的数量被分散, 安全性较差.

1.1.2 交易映射机制

分布式账本有未花费交易 (unspent transaction output, UTXO) 和账户的两种交易模型. 在分片系统的交易分发过程中, 不同的交易模型会采用不同的映射机制来将交易映射至分片. 映射机制会影响分片处理的性能, 使得系统中存在片内和跨片两类交易.

(1) UTXO 模型的交易映射机制

UTXO 最早应用于比特币中, 将交易看作是一定数量的币在不同私钥持有人之间进行权限转移的操作. 这使得在总币值不变的情况下, 交易过程改变了资产的所有权. UTXO 可以表示系统的当前状态, 每一次交易都会引起 UTXO 变化, 即系统状态变化.

基于 UTXO 的分片是一种基于状态分片的思想, 其将 UTXO 划分至分片内, 每个分片代表着系统的一个子状态. 存在依赖的分片之间相互协作, 实现子状态的转换; 而相互独立的状态分片之间可以并行处理. 目前, OmniLedger, Elastico 等项目主要采用了基于 UTXO hash 的交易划分, 将 UTXO 和交易映射到分片. 一笔交易常会涉及多个 UTXO, 系统中一笔交易涉及的多个 UTXO 会映射在多个不同的分片中, 这类交易的执行会涉及大量跨片处理, 降低了系统的交易处理性能.

(2) 账户模型的交易映射机制

类似于传统中心化账本的账户模型, 以太坊[2]、蚂蚁链[21]、迅雷链[22]、Near[23]等系统采用的是账户余额模型. 在该模型中, 账户地址多采用 hash 串的形式. 分片系统可以将用户地址空间截取固定 k 位作为分片依据, 来映射到 2^k 个分片中. 交易可以根据账户地址来映射到分片, 当资产在不同的分片之间转移, 该过程也会产生较多的跨片处理, 带来一定的通信和时间开销.

Monoxide[18]是 Wang 等人提出的基于账户模型的分片账本, 系统会将用户地址空间的前 k 位来映射到对应的分片 (即 Zone) 中, 再进行交易处理.

(3) 对象模型的交易映射机制

Chainspace[16]设计了分片场景下的智能合约, 其合约可定义多个对象 (object), 即合约中有状态的原子结构. 当合约中交易的输入 object 为激活状态时, 该交易为可执行的有效交易. Chainspace 分片负责管理和维护 object 状态, 保证可执行交易的所有输入 object 都为激活状态. 当有效交易被提交执行, 输入 object 状态转移至非激活态.

基于对象的合约模型使用比较灵活. 采用分片管理对象会提升了其系统的交易并发处理能力. 基于对象的合约模型使得合约账本不仅局限于传统支付应用, 而是可以扩展到更广泛的应用场景. 但是, 该机制的交易会涉及多个对象, 使得该交易进行跨片处理的概率较大, 影响性能提升. 特别是在合约逻辑复杂情况下, 涉及的对象越多, 其跨片的概率会越大.

1.1.3 跨片交易处理

交易映射机制具有随机性, 一笔交易的发送方和接收方往往会被映射到多个不同分片. 在这类交易验证、执

行时, 系统需要提供跨片机制来保证这类交易的原子性, 即其涉及的多个分片统一执行或放弃本次交易, 保证跨片交易执行的一致性.

在交易仅涉及双方转账, 即仅关联到两个分片的情况下, 跨片需保证交易执行在两个分片的一致性. Monoxide 分片采用了基于中转交易的 Efficient Cross-Zone Atomicity 协议[18], 其将区块划分为链块和交易块两部分, 链块记录了区块工作量证明信息, 而交易块记录了两种交易, 即片内交易和跨片交易. 在跨片交易中, 转账账户所在的分片会构建中继交易, 并将跨片交易信息直接发送给接收片. 接收片会先缓存收到的跨片信息, 在构建新块时, 会从收到的跨片交易中选择交易, 验证并加入区块中. 由于接收片需要异步确认和验证中转交易, Efficient Cross-Zone Atomicity 协议只能保证交易的最终原子性.

当交易涉及多个输入分片时, 系统往往采用两阶段协议 (2PC) 来保证交易执行的原子性. 两阶段协议是指系统中存在一个协调者, 负责转账信息收集和决策, 协调者会先请求其他节点询问是否就绪, 当且仅当所有节点就绪, 协调者发送消息同步所有节点并执行交易; 否则, 发送取消交易的消息至所有节点. 如图 5 所示, 在分布式账本中, 主要有 3 种方法实现两阶段跨片处理: 客户端协调、输入片交易锁定、输出片协调.

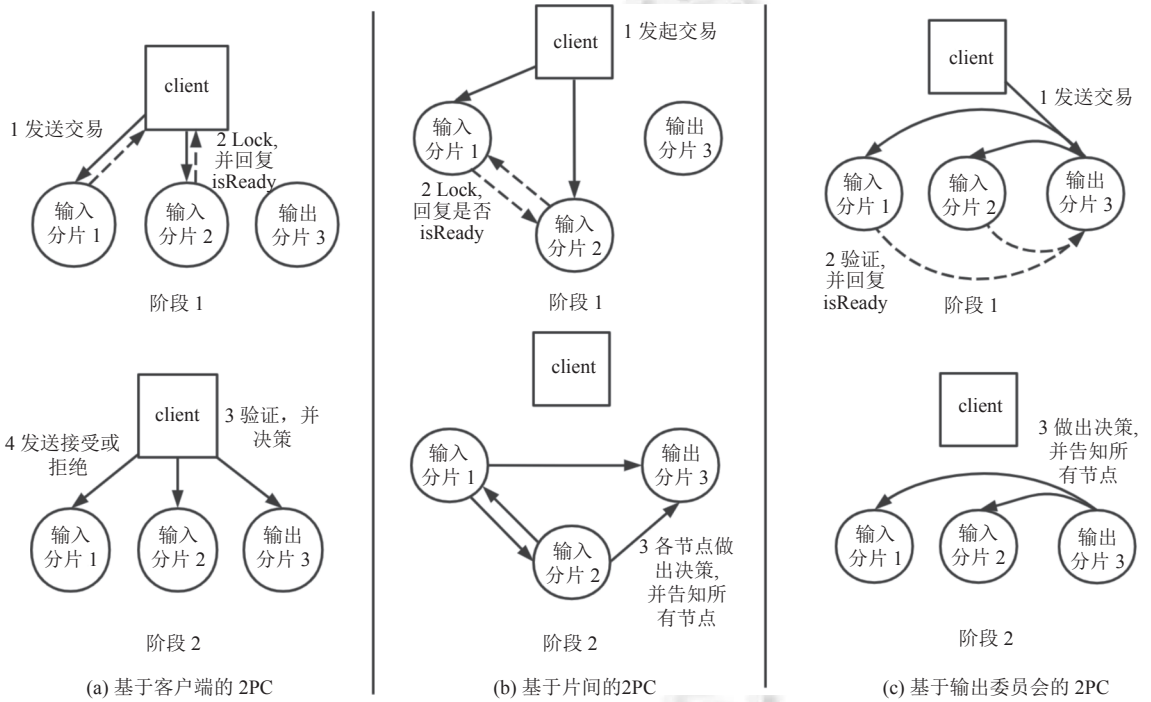

图 5　3 种两阶段跨片处理

(1) 客户端协调. 在阶段 1, 客户端将交易信息发送至输入片, 输入片检查交易是否可以执行, 并返回客户端“接受”或是“拒绝”交易. 在阶段 2, 客户端做出决策, 当所有输入片接受交易的情况下, 同步所有输入、输出片, 执行用户交易; 否则, 通知所有输入片取消交易. 客户端协调跨片的通信复杂度为 O(n), 效率较高. 目前, OmniLedger[20]采用了基于客户端的 Atomix 协议, 即通过客户端收集各分片的接收/拒绝证明来做出决策, 保证交易的一致性.

(2) 输入片交易锁定. 用户将交易发送至多个输入片后, 各输入片锁定交易, 并确认交易是否可行, 再通过广播机制相互同步, 并确认结果. 各输入片在收到确认后, 做出决策并再次广播决策至交易相关的分片 (包括输出片). 该机制增加了输入片之间通信量, 会影响系统效率. 目前, Chainspace 采用了 S-BAC 协议, 结合片内 PBFT 协议和跨片广播机制, 发送并确认决策信息, 从而实现跨片一致性[16].

(3) 输出片协调. 用户将交易请求发送至输出片, 输出片将交易请求分别发送给各个输入片, 当且仅当收到所

有输入片确认就绪的信息后, 同步执行交易. 这种方式可大量减少通信开销. RapidChain[17]分片采用了基于输出委员会复制交易的方式, 实现跨片共识.

1.1.4 分片重组

分片减少了参与共识的节点数量, 但也降低了单一分片的安全性. 恶意节点可以通过不断涌入单一分片, 使得恶意节点的比例超过该分片的安全阈值, 因此分片系统引入了重组机制, 即系统会在每间隔一定时间 (即 epoch) 对系统节点进行重新分片.

分片的重组方法主要分为两类: 整体重组和部分重组. 整体重组是指对所有节点重新进行委员会划分, 即按照上述委员会划分机制, 对系统所有节点进行重新分片. 这造成系统中所有节点都需要重新启动, 进行存储迁移等操作, 造成大量的性能损耗. 部分重组是指在每个 epoch 中, 系统会选取部分节点重新进行分片. RapidChain 基于改进的布谷鸟规则 (the cuckoo rule)[24], 实现部分重组. 布谷鸟规则是采用 hash 函数来将节点映射到 [0, 1) 区间的一个随机数, 再将 [0, 1) 区间划分为 k 个子域, 对应于 k 个委员会. 当有新节点加入某一子域时, 为保证子域之间节点数量的平衡, 新节点位置周围的临近节点会被移除到其他子域. 这种机制有效限制了受影响的子节点数量, 其仅需要部分节点重新启动, 并进行存储迁移, 节省了系统重组带来的性能开销. 此外, RapidChain 分片采用有限布谷鸟原则, 即把委员会分成两类, 分别是活跃委员会和不活跃委员会, 在节点加入网络时, 系统会把该委员会中的固定数量节点随机加入不同的不活跃委员会, 委员会的规模大小保持平衡, 保证了分片的稳定性.

OmniLedger 设计了一种灰度化处理方法, 即分片中分出部分节点用于下一个 epoch, 进行重组优化. 在 epoch 的切换过程中, 系统会从当前 epoch 的每个分片中通过随机方式转移出部分节点, 用于下一个 epoch 中新分片节点, 进行分片启动. 剩余节点会继续进行交易处理, 这使得将当前 epoch 的交易处理和下一个 epoch 的分片重组并行, 节省了 epoch 切换的时间开销, 提升系统性能. 在 OmniLedger 中, 移出节点的个数需要限制小于片内节点数量的 1/3, 从而使得移除节点后的片内剩余节点可以保证 BFT 共识的安全性.

分片重组会存在存储迁移的问题, 对此, OmniLedger 提出了一种“状态区块”的设计, 状态区块会记录当前分片的状态信息. 在节点转移前, 片内的领导节点会将当前分片的 UTXO 生成一个有序 Merkle tree, 将 Merkle root 加入区块头中, 并进行片内共识, 生成状态区块. 该状态区块可以成为下一个 epoch 的创世区块, 减少了节点因重组产生的存储和启动的开销.

分片通过将节点进行委员会划分, 并将交易分发至各委员会, 各委员会并发处理交易, 提升了系统的并发度. 表 1 对比了分片项目的委员会, 交易, 跨片, 重组等设计.

表 1 分片项目对比表

方法	Elastico	Chainspace	OmniLedger	RapidChain	Monoxide
委员会机制	Hash 位	—	随机函数	—	自由委员会
交易数据模型	UTXO	UTXO	UTXO	UTXO	账户模型
交易映射	—	基于object的UTXO	UTXO hash	UTXO ID	用户地址空间
跨片协议	—	S-BAC协议	基于客户端的Atomix协议	基于输出委员会的一致性协议	基于中转交易的Efficient Cross-Zone Atomicity协议
跨片通信复杂度	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n)$	$O(1)$
片内共识	PBFT	BFT	ByzCoinX	BFT-based consensus	PoW
恶意节点容忍比例	1/3	1/3	1/4	1/3	1/2
分片重组	整体重组	—	整体重组	Bounded Cuckoo Rule 部分重组	无重组

但在分布式环境下, 分片增加了委员会机制, 片间通信等环节, 这些环节在一定程度上减弱了原有的性能提升, 导致各个分片的性能不同.

Elastico 在 1 600 个节点的环境下, 每秒可以出 16 个 1 MB 大小的块[15]. Chainspace 使用 4 核 16 GB 的 Amazon EC2 容器, 在 15 个分片, 每个分片有 2 个节点的环境下, 每秒可以处理 350 笔交易, 每个节点每秒平均可处理 11 笔

交易[16]. OmniLedger 使用了 1 800 台不同配置的机器, 在恶意节点比例为 1%, 每个分片有 4 个节点的环境下, 其每秒可以处理超过 10^5 笔交易[20]. RapidChain 使用了 4 000 个节点, 块大小为 2 048 KB 时, 每秒可以处理超过 7 000 笔交易[17]. Monoxide 使用了 1 200 台 8 核 32 GB 的虚拟机, 在 2 048 个分片环境下, 系统每秒可以处理 11 694.89 笔交易, 每个节点每秒平均可处理 9 笔交易[18].

1.2 多 链

随着分布式账本技术日益广泛地应用在各个领域, 单链难以承担高负载和多领域的业务需求. 一方面, 其需要采用分而治之的设计思想, 按照不同的业务需求、数据类型、交易特定等设计实现不同的账本系统, 从而满足特定领域的性能、可靠性和安全性等需求, 提高系统的可扩展性; 另一方面, 在整个体系生态中, 由于技术体系的多样性, 分布式账本系统日益呈现出数据、算法、协议的异构性, 其需要建立有效的机制, 支持异构系统之间互操作. 多链技术应运而生, 其通过侧链、公证人机制、哈希锁、中继等技术, 实现了可集成、可扩展的分布式账本系统, 扩展了传统单链的处理能力, 可支持更多场景的应用, 例如在金融领域, 其可以应用于资产抵押, 借贷等场景, 支持资产和数据的流动性. 多链平台的代表性应用项目有微众 WeCross[25] , 蚂蚁链跨链平台[21]等.

多链技术是通过不同的连接机制实现多条同构/异构区块链的互操作性, 使得交易可以并发在多条链处理, 提升了系统的交易处理能力. 但不同的连接技术对多链聚合、跨链通信的性能影响不同. 本节将针对多链连接技术来分析其性能设计的影响.

多链技术可以分为两类, 一类是同构链的连接, 即对已有链进行扩展, 如父子链. 而另一类是异构链的连接, 其需要解决的是数据可信跨链问题, 根据信任对象的不同, 可以划分侧链、公证人、哈希时间锁和中继 4 种. 其中侧链的信任基础是一定长度的区块链难以被分叉攻击; 公证人机制信任基础是公证人节点集合; 哈希时间锁的信任基础是哈希函数的单向性; 中继的信任基础是依赖于中继节点或接收链本身. 表 2 从结构、验证、实现等方面, 对比了上述 5 种多链技术.

表 2 多链技术对比表

对比项	侧链	公证人	中继	哈希时间锁	父子链
结构	双链锚定	星型结构	可扩展的星型结构	双链锁定	树状结构
是否需要中间结构	否	是	是	否	否
验证方	链本身	公证人节点	中继节点或链本身	链Hash验证	父链
可扩展性	两链直接进行锚定	较差, 由于难以保证公证人系统的扩展性	较好, 由于中继系统可扩展	两链直接加锁	父链可以有若干子链
典型项目	Liquid Network、BTC Relay	InterLedger	Polkadot、Cosmos	闪电网络	Plasma

1.2.1 侧 链

侧链技术是指在原有账本链 (即主链) 外, 创建或者加入新的账本链 (即侧链), 该链通过连接技术来连接到主链, 并且可以有相对独立的架构设计. 主链可以转移定量资产至侧链, 在侧链进行交易, 将部分交易转移到侧链, 从而降低主链的负载, 提升主链的性能. 2014 年 Back 等人首先提出了基于 SPV (简单支付验证证明) 双向锚定的侧链连接方案[26]. 如图 6 所示, 其具体步骤如下.

(1) 主链接收到资产转移交易请求后, 将资产锁定到特定地址.

(2) 待确认交易后, 主链创建 SPV 证明发送到侧链.

(3) 侧链验证通过 SPV, 并释放等价的数字资产, 进行交易处理.

(4) 如果侧链的剩余资产需要主链处理, 侧链依旧可以将本链资产锁定, 通过 SPV 证明反向传回, 令主链释放等价资产.

双向锚定技术不存在类似交易所的第三方干预, 仅需要接收链验证 SPV 证明即可, 主侧链之间的通信比较高效. 2018 年, Liquid Network[27]采用双向锚定技术, 将比特币 (BTC) 和 Liquid 比特币 (LBTC) 锚定连接, 实现了世界上第 1 个基于比特币的侧链. RootStock 通过基于智能合约的双向锚定技术来建立在比特币上的侧链, 实现比特

币 (BTC) 和 RSK Smart Bitcoin (RBTC) 之间的货币转移, 为比特币扩展了功能, 也提升了一定性能[28,29].

但是, 如果主侧链频繁进行资产转移, 双向锚定需频繁地锁定、验证、消息传输, 耗费大量的计算和通讯资源. 此外, 双向锚定仅限于 2 条链之间的资产转移, 对多链之间复杂的资产转移需经过多次双向锚定.

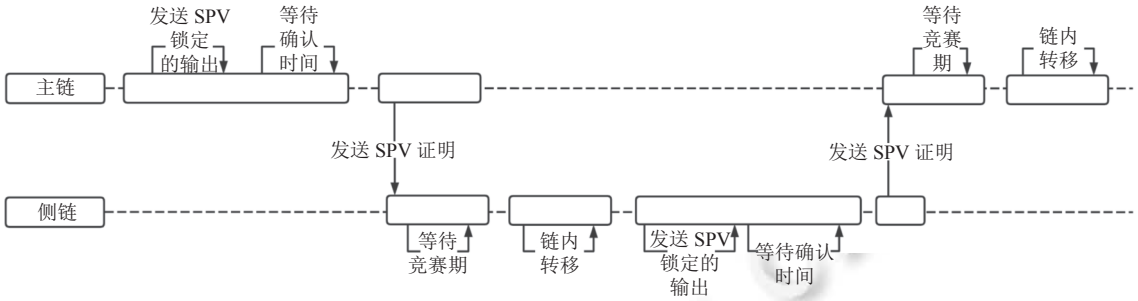

图 6 基于 SPV 的双向锚定[26]

1.2.2 公证人模式

公证人模式是在多链之间, 采用可信的第三方作为公证人, 负责验证跨链消息的可靠性, 并签名传输, 来实现多链之间的消息交互.

公证人节点对跨链事件进行主动或者被动监听, 对事件进行可信验证, 并做出事件响应. 图 7 表示了公证人模式的基本过程. 根据第三方公证的结构, 其可以分为单节点和多节点两种公证人模式. 多节点的公证人模式比单节点公证人机制更安全, 它是由一个节点集群构成第三方公证来进行验证和签名, 节点集群可以通过多重签名, 门限签名, 分布式签名等密码学方法, 在交易验证时共同进行签名和转发, 保证第三方公证实体的可信度.

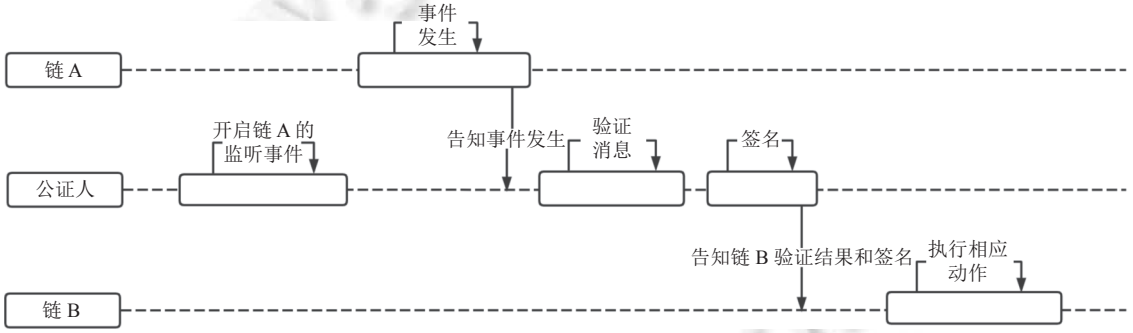

图 7 公证人模式

公证人模式实现了多链连接, 提高了多链消息的可信度. 但是中间节点使得两条链无法直接通信. 其次, 在跨链频繁的情况下, 公证人节点的崩溃或者低性能会造成多链的性能瓶颈. 特别是, 在单节点作为公证人时, 其有较低的性能, 节点容易被攻击, 安全性较差. 多节点的公证人模式采用了密码学机制, 提升了安全性, 但是多节点处理会带来消息同步, 密钥分发, 收集等阶段, 造成计算和通信资源消耗, 影响性能提升. Ripple 公司于 2015 年提出了“InterLedger”协议, 采用了公证人模式, 并创建了两种公证模式, Atomic 模式和 Universal 模式[30]. Atomic 模式是通过选定一组公证人节点来进行账户转账, 其允许多个连接器串行连接, 组成支付通道, 实现多账本之间的消息传输. 而 Universal 模式是通过激励机制, 来实现各方之间的安全支付.

1.2.3 哈希时间锁

哈希时间锁 (hash time lock contract, HTLC) 是一种基于跨链的原子转移, 最早由 TierNolan 提出[31]. HTLC 采用智能合约技术, 通过哈希锁和时间锁来锁定跨链账户, 实现跨链账户之间的货币交换和信息交互.

假设账户 A 已有的链 a 的资产, 账户 B 已有链 b 的资产, 账户 A 希望置换账户 B 的资产. 如图 8 所示, 哈希

时间锁的基本过程是账户 A 用随机数 h 生成哈希值 H, 即 hash(h), 发送给账户 B. 接下来, 账户 A 加哈希时间锁, 即用哈希值 H 和时间值 T, 锁定链 a 上的转账资金. 其中哈希锁保证当收到原始值 h 后, 释放锁; 时间锁保证当设置时间到期后, 释放锁. 账户 B 收到哈希值后, 同样用哈希值 H 和时间值 T2 加锁. 当账户 A 将原始值 h 发送到账户 B 上, 合约执行解锁, 使得 A 获取 B 的资产. 账户 B 收到 h 后, 同样通过 h 解锁并获取账户 A 的资产. 如果原始值 h 在设定时间内没有被发送, 所有冻结的资产会在时间锁到期后释放退还.

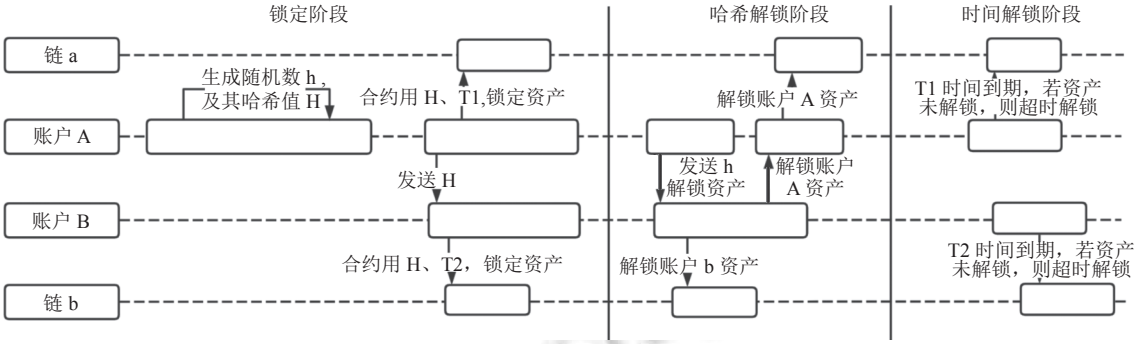

图 8 哈希时间锁[31]

哈希时间锁通过智能合约, 实现了多链之间的互联互通, 转移原有账本链的负载, 提升原有账本链的性能. 但是, 其仅涉及基于账户之间的资产转换, 只会影响特定账户的负载, 对性能的提升具有一定局限性.

1.2.4 中 继

链之间的消息传递可以通过中继, 即基于统一消息传递协议的第三方可扩展组件, 进行消息的跨链传输, 将多条异构链连接起来. 由于中继负责消息收集和转发, 多条异构链可以采用同一个中继结构连接, 形成星型拓扑结构.

基于中继的多链集成可分为两类. 一类是有消息验证的中继, 即中继在传输消息的过程中, 维护一个多链的全局状态, 传输的消息可以被记录和验证. Polkadot[32]通过构造一条第三方的公有链, 即中继链, 并通过跨链消息传递协议和转接桥, 将多条异构链 (即平行链) 接入并连接. Polkadot 提供了验证人节点, 负责平行链数据的验证.

另一类是无消息验证的中继, 即中继不负责交易的验证, 仅提供消息的传输, 目标链收到中继数据后, 会通过 SPV 证明、节点签名数量等信息来自行验证. Cosmos [33]是将枢纽 (hub) 作为中继, 采用跨链交互协议 (inter-blockchain communication protocol), 实现各 Zone (即平行链) 的跨链通信. 所有接入 Cosmos 的链需要具备交易验证的能力.

中继的多链集成提升了传统单链/侧链的扩展性, 转移负载至其他多条链上. 其次, 相比传统的公证人机制, 中继设计侧重于跨链消息传输, 扩展性较强, 支持多链的集成. 而且, 中继会缓解一定网络压力, 避免链链直接通信造成的系统瘫痪. 但可消息验证的中继通过中继链来维护多链的全局状态, 会给多链通信带来存储和性能开销.

1.2.5 父子链

父子链是一种已有链的扩展方式, 其中每条子链负责特定计算任务, 并且定期将状态 Merkle 树根值的结果证明同步至其父链. 多层父链和子链的同步关系, 构成了如图 9 所示的树状多链拓扑结构. 树结构中根节点为主链, 其分支节点分别执行不同的计算任务.

交易只需要提交到树最末端的叶子节点, 即子链. 子链周期性提交 Merkle 树根值的结果证明, 交易会间接地流向主链. 父节点将计算任务分散并委托给各个子节点, 子节点完成计算后, 再提交至父节点, 由父节点进行验证和整合.

分散计算任务使得根节点仅需处理少量子链请求, 减少了父节点的负载, 扩展了系统的交易处理能力, 提升了性能. 但是, 如果父子链的树结构越来越深, 交易会越频繁地被验证. 这既造成了计算开销, 也会增加交易处理时间和延迟时间. 因此, 树结构应尽可能扁平化设计. Plasma 是由 Poon 等人提出的以太坊 Layer2 扩容方案[34], 其通过一系列以太坊的智能合约来实现其父子链树形结构.

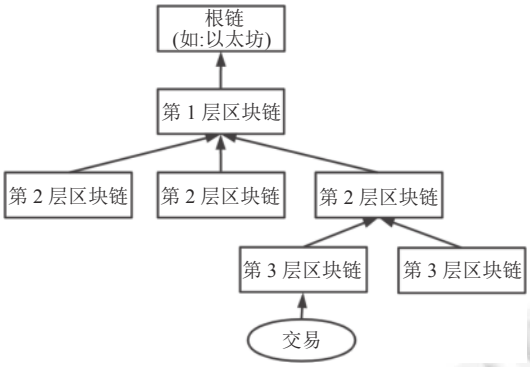

图 9 父子链[34]

1.3 流程优化

传统分布式账本的交易处理过程采用 Order-Execute 流程, 即先对交易定序、打包和共识, 再广播同步到各个节点, 最后, 各节点验证区块, 并以上述确定的顺序执行交易.

流程优化是通过将交易处理的流程进行阶段性划分, 即划分为若干子流程, 再调整各个子流程, 实现性能优化. 主要的流程优化方法包括预执行、流程并行和延迟执行 3 类. 图 10 对比了传统共识流程以及 3 种优化的流程.

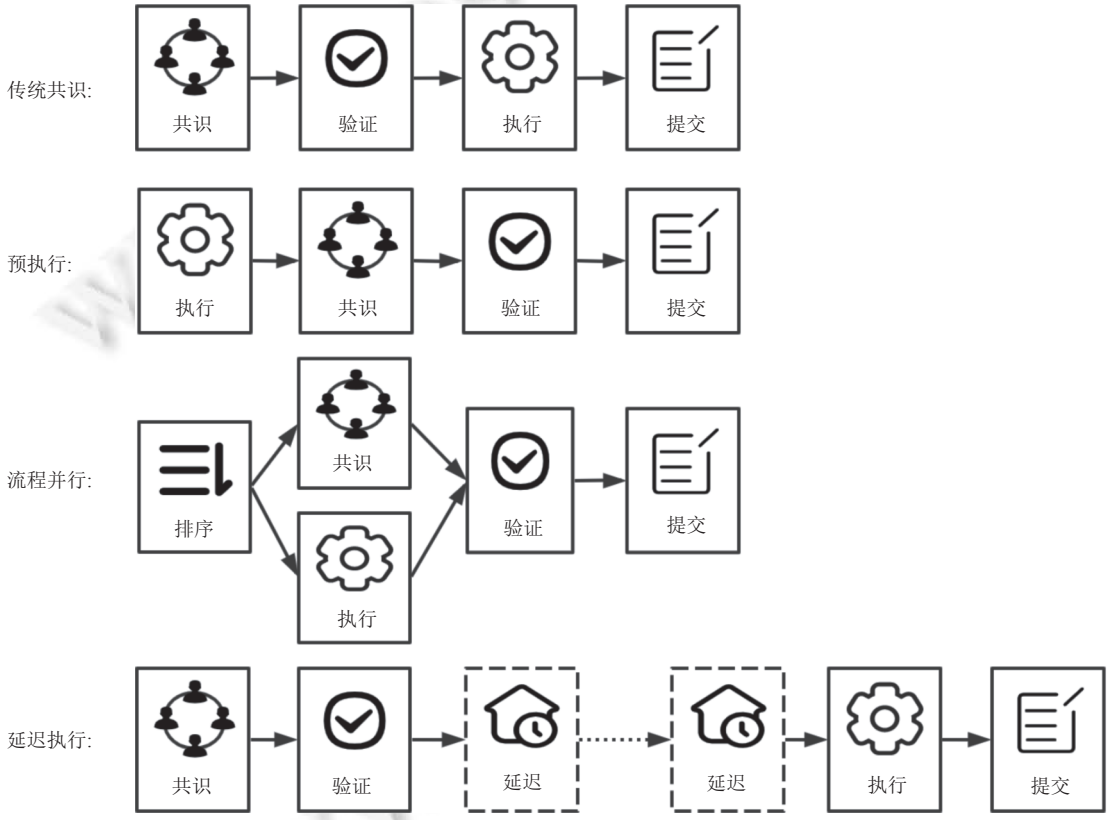

图 10 流程优化

1.3.1 预执行

预执行对交易处理过程的顺序做出调整, 采用 Execute-order-validate 流程. 其由一部分节点先执行交易, 得到

交易结果; 再由另一部分节点对执行后的交易进行验证, 共识和广播验证.

预执行中其节点根据阶段承担了不同的角色, 执行不同的任务. 假设节点集合为 S, 令节点子集合 S_e 负责交易执行, 节点子集 S_o 负责交易排序. 其中,

$$S = S_e \cup S_o \tag{4}$$

$$S_e \cap S_o = \varnothing \tag{5}$$

预执行为节点赋予不同的角色和任务. 系统可以根据其机器性能配置来为节点分配任务, 充分利用节点的优势资源, 提升系统整体性能. 此外, 交易执行和排序验证阶段为一个 epoch, 节点的角色分离可以实现 epoch 之间的异步处理, 即当 S_o 集合节点负责 epoch n 的交易执行时, S_e 集合节点负责 epoch n+1 的交易执行, epoch 之间不需要同步. 还有, 为了提升交易执行阶段的效率, S_e 集合也可以切分为若干子集合, 在高并发场景下, S_e 子集可并行完成交易执行, 提升执行阶段的性能.

HyperLedger Fabric[3]是 Linux 基金会提供的开源许可链框架, 其采用了基于客户端的预执行的模块化架构设计, 即客户端提交交易至执行节点, 收集来自执行节点的背书和交易结果, 并将其提交至排序节点. Fabric 在 peer 节点数量大于 1 000 时, 其每秒吞吐量能够大于 3 500 笔交易[3]. OmniLedger 的片内采用 Trust-but-Verify 架构[20], 即令一部分节点为乐观验证者, 做交易预处理; 另一部分节点为核心验证者, 负责交易排序, 乐观验证者会将执行后的结果, 直接发送至核心验证者, 由核心验证者共识出块.

1.3.2 流程并行

流程并行是将交易处理的串行工作流转换为并行工作流, 从而提升系统并发度, 优化性能. Qi 等人[35]提出 BIDL, 一个针对数据中心网络优化的许可链框架, 其通过将交易的共识和执行并性化处理, 提升框架的处理性能.

并行共识和执行子过程会提升系统的交易处理效率, 但是其会带来数据不一致的问题, 造成提交节点无法立即提交有效交易. BIDL 引入了排序节点, 对交易编号排序, 减少交易冲突, 提高系统性能. 但恶意的排序节点会给共识和执行节点发送不一致的交易, 导致系统低性能. BIDL 采用基于拒绝列表的视图更改协议, 通过检测系统的吞吐量来触发视图更改.

1.3.3 延迟执行

在传统区块链系统中, 交易在打包成块, 全网广播后, 会立即被网络节点执行. 账本的结构会随着分叉的出现而改变, 从而改变交易的顺序, 造成交易的多次执行, 消耗节点的计算资源, 影响系统性能. DAG 账本结构允许分叉, 其账本结构更容易改变. 如在 Conflux 中, 其新加入账本的分叉交易会带来中枢链发生变化, 更容易导致交易被反复执行多次.

因此, Conflux[36]提出延迟若干区块再执行的策略来消除不必要的执行过程, 进一步优化性能. Conflux[36]在 400 个 Amazon EC2 节点上采用以太坊上真实交易和随机生成交易作为工作负载, 经测试可得其每秒交易吞吐量可以到 3 480 笔, 其交易确认时间小于 1 min[37].

1.4 链下设计

链下设计是将交易的复杂处理移放至链下, 令链上的资源用于验证, 减轻链上计算压力, 提升链上交易处理的性能. 链下技术可以根据链下处理和同步方式, 分为两类, 分别是通道技术和交易打包. 表 3 从其交易处理类型、数据组织、链上数据等方面, 对比了通道技术和交易打包两种链下技术.

表 3 链下技术对比表

对比项	通道技术	交易打包
处理交易类型	小额交易	普通交易
交易方数量	(单通道)双方	多方
链下数据组织	通道	Merkle tree
同步链上时间	哈希锁: 双方协定同步 时间锁: 设定时间同步	— (累计同步)
同步数据内容	双方签名和交易结果	Rollup结果、proof证明、交易信息
典型项目	闪电网络、雷电网络	Zk Rollup、Optimistic Rollup

1.4.1 通道技术

通道技术将频繁上链的小额交易转移至链下处理, 减少系统上链频率, 提升链上性能. 通道是一种链下交易处理架构. 交易双方可以在链下通道锁定资产, 并完成多笔小额交易. 当达到合约设定的上链条件后, 链下累积的状态会被同步到链上, 不需要同步交易数据, 直接在链上实现清算. 图 11 展示了通道技术的具体过程, 其中用空白方框表示区块.

通道同步累计交易的最终状态, 使得链上不需要执行多笔交易, 仅需要对最终结果验证, 共识, 大大节省了链上计算、通信资源, 提高了链上交易处理的性能. 但是, 对于单笔交易而言, 其需要经过链下处理, 达到条件才能链上同步, 这会增加单笔交易的确认时长.

2016 年, Poon 等人提出了基于比特币的状态通道设计, 即闪电网络, 采用哈希时间锁 (HTLC), 可撤销的顺序成熟度合同 (RSMC) 机制, 确定是否上链, 保证上链数据的正确性[38]. 随后, 基于以太坊的通道设计, 即雷电网络[39], 被提出, 其主要是基于以太坊的 ERC20 令牌传输, 采用以太坊图灵完备的智能合约开发, 可以实现比闪电网络更加复杂的智能条件, 如组合锁, 保证链下交易的安全性, 但同时也会牺牲一定的性能.

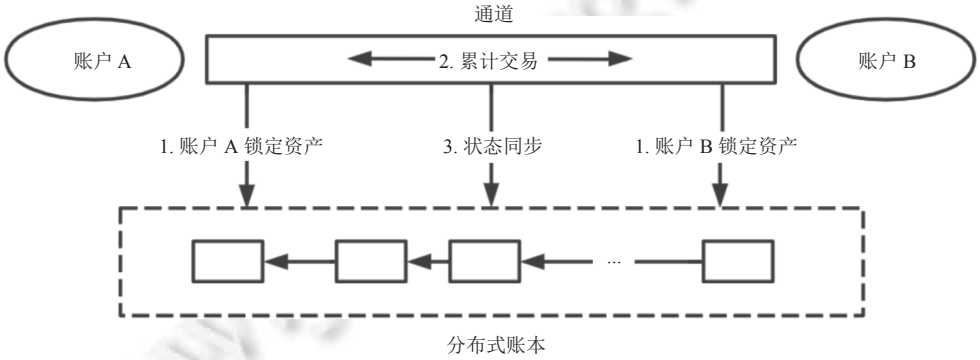

图 11 通道技术

1.4.2 交易打包

交易打包, 即 Rollup, 是指系统在链下会维护一棵 Merkle 交易状态树, 作为链下交易的数据组织, 其根值 root 是由叶子节点自底向上经过哈希计算得到的哈希串, 其间接包含了新的交易信息. 链下会周期性地向链上提交根值. 不同于通道技术, Rollup 会同步交易数据至链上, 仅将数据的计算移到链下, 实现数据可用性. 交易打包采用密码学, 博弈等方式来保证提交数据的正确性.

交易打包在链下采用了 Merkle 树结构来对交易集合进行压缩, 减小了链上的交易频率, 节省了链上的计算和通信开销, 提升了系统的性能. 但是对于单笔交易而言, 其需要经过在链下处理, 等待上链, 待链上共识交易数据, 这增加了单笔交易的确认时长.

(1) 基于零知识证明的交易打包

链下交易的执行者会在执行完成后, 生成相应的零知识证明, 向链上提交原状态树根值, 新状态树根值, 交易信息, 零知识证明等信息. 链上合约会根据证明来验证数据的正确性. Whitehat 等人提出 Zk Rollup[40], 即基于 ZK-SNARK 的零知识证明解决方法, 为执行者信息提供有效性证明. Validium[41]是类似于 Zk Rollup 的架构, 采用零知识证明来验证数据的有效性. 但是其交易数据存储在链下, 验证者不需要提交交易数据.

零知识证明数据很小, 网络传输数据较少, 且链上验证证明的速度也很快, 链上交易处理的效率较高. 但链下创建证明的计算开销较大, 其效率较低. 创建证明的低效率会影响到单笔交易的确认时间. Validium 将交易数据保存于链下, 相比 Zk Rollup, 减少了网络通信的开销, 性能相对较好. 此外, Zk Rollup 不能支持复杂智能合约的世界状态, 其灵活性较差.

(2) 基于博弈论的交易打包

基于博弈论的交易打包是在链下采用经济惩罚机制来约束链下执行者, 保证执行者难以作恶.

Truebit[42]采用用户、链外执行者、验证者等角色来实现博弈约束. 其中, 用户 (task giver) 负责上传智能合约的代码, 并支付佣金. 链外执行者 (solver), 用于执行智能合约, 并且提供保证金. 验证者 (verifier), 用于重新执行任务, 也需要提供保证金. 如果执行者和验证者的结果不一致, 链上作为仲裁者, 验证答案. 提供正确答案的一方可以获取用户的佣金, 被验证错误的一方会从保证金中扣除整个验证过程所需的手续费. 如果链外执行者的结果在验证期间没有异议, 链外执行者可以获得佣金. 由于整个过程涉及保证金扣除, 退还等复杂业务逻辑, 基于博弈论的交易打包往往会在链下维护一个支持智能合约的虚拟机, 实现上述业务逻辑. 图 12 展示了 Truebit 的交易上链过程, 其中用空白方框表示区块.

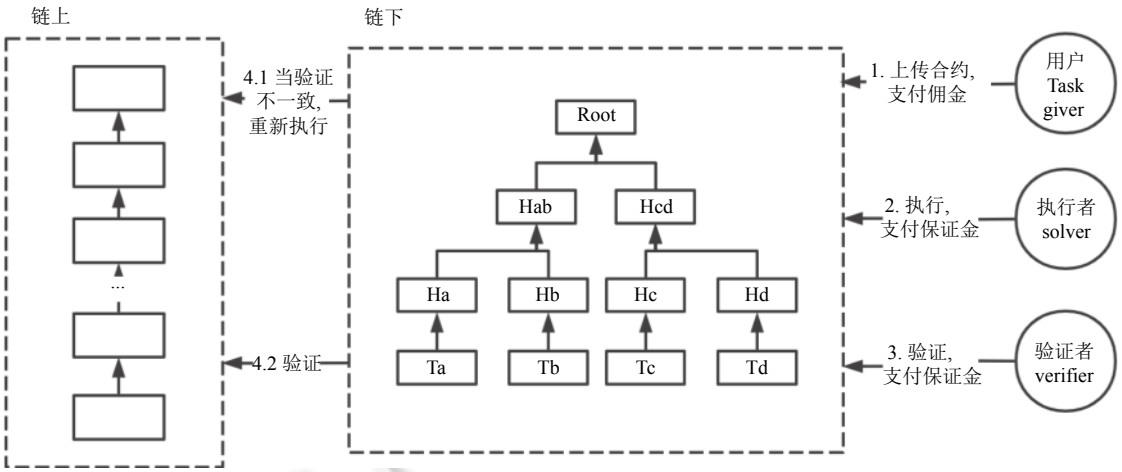

图 12 Truebit[42]

基于博弈论的交易打包没有采用密码学计算, 消耗较少的计算资源. 但是其需要交易在上链前经过链下验证, 难以保证链下的高效, 延长了单笔交易的确认时间. Optimistic Rollup[41]也是以太坊 Layer2 的一种解决方案, 其采用基于博弈论的经济约束, 保证提交数据的正确性, 并且在链下提供虚拟机, 用于支持智能合约等复杂的世界状态. 但是, 经济学制约难以完全保证系统安全性. 此外, Optimistic Rollup 等待验证, 会产生一定的时间开销.

2 账本数据结构

账本数据结构是指节点内组织和存储交易数据的方式. 常见的账本结构有两类, 分别是链式账本和图式账本. 表 4 从账本结构、节点和边的表示、其粒度以及性能等方面, 对比了链式账本和图式账本.

表 4 账本数据结构对比表

对比项	链式账本	图式账本
结构	链式结构	有向无环图
节点表示	区块	交易/区块, 以交易为主
边表示	时间序列关系	以偏序关系为主, 也可以包括验证关系, 父子关系, 叔父关系等
分叉处理	保守, 选择最长或者最重链	乐观, 优先接受
冲突取舍的粒度	以分叉链为粒度	以交易为粒度
记账效率	低, 一个时间段内, 账本结构只能接受一个有效块	高, 一个时间段内, 账本结构可以接受所有的分叉块
安全性	高, 有保守的消除分叉机制	低, 允许分叉
代表项目	比特币, 以太坊	DAGCoin, IOTA, Byteball, Conflux

2.1 链式账本优化

以区块链为代表的分布式账本是由若干区块通过记录前一区块的哈希值连接成的链式账本结构. 以比特币为

例, 如图 13 所示, 链上第 1 个区块为创世区块, 每个区块由块头和块体两部分组成. 块头是由前一区块哈希值, 时间戳, 难度目标, 随机数, 交易 Merkle 树的根值组成的结构. 块体存放了 Merkle 树的方式组织的若干交易数据.

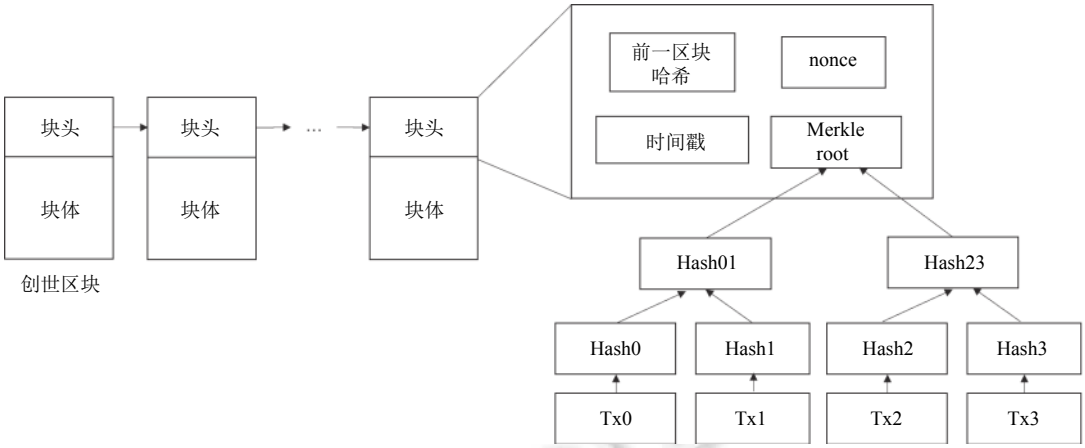

图 13 传统账本的数据结构

根据交易处理数的计算公式, 当交易容量为固定大小时, 系统处理交易的吞吐量 (*TPS*) 是由区块大小和区块生成时间决定的. 因此, 链式账本结构的性能提升主要依赖块的大小和出块时间.

2.1.1 块的大小

根据吞吐量公式, 在出块时间和交易大小不变的情况下, 增加块的大小会提升系统吞吐量. 比特币将区块的大小设置为 1 MB[1]. 由于比特币性能难以满足现实需求, 比特币社区曾致力于增加区块的大小. Bitcoin Unlimited[43], 一种调整块大小的工具, 完全取消了区块大小的限制, 使得矿工可以自由调整块的最大容量, 而不仅是局限于 1 MB. Garzik[44]也在比特币社区中提出将比特币的区块大小由原来的 1 MB 增大为 2 MB. 2017 年 Bitcoin Cash[45] 作为比特币的硬分叉项目, 将块的大小限制增加为 8 MB, 并于 2018 年将块大小又扩大了 4 倍, 即 32 MB.

但是, 仅增加块的大小会使得网络传输区块的时间变长. 在规定时间内, 部分节点无法收到该区块, 造成不同节点的账本有不同的账本数据, 造成分叉的安全隐患. 此外, 在交易大小不变的情况下, 增加块的大小会使得块内交易数量增加, 其他收到区块的网络节点需要更长的时间来验证交易, 影响系统性能.

2.1.2 出块时间

在块和交易大小不变的情况下, 减小出块时间也可以提升吞吐量. 比特币是通过其难度动态调整公式来调整出块时间. 令其实际出块时间为 *actualTime*, 期望出块时间为 *expectTime*, 当前难度为 *currentDiff*, 调整后难度为 *newDiff*.

$$newDiff = currentDiff \times \frac{expectTime}{actualTime} \tag{6}$$

如果将其预期出块时间小于 10 min, 根据比特币难度动态调整公式, 预期时间减小, PoW 中出块难度减小. 全网节点会更容易出块, 更容易出现并发、分叉的情况. 分叉会带来较多系统孤块, 造成存储和计算资源的浪费. 而且, 如第 2.1.1 节所述, 分叉会带来安全问题, 系统需要最长链法则来处理分叉, 这会增加交易的确认时间. Kiayias 等人[46]认为, 合适出块速率的时间往往取决于一轮完整的信息传播. 以太坊是采用了最重子链 (GHOST 协议[47]), 将出块时间调整为 15 s[2], 将出块速度提高到 40 倍, 充分利用叔父块来处理临时分叉, 保证账本分叉可以快速沿着最重子链的方向收敛, 减少了分叉.

区块链结构允许节点每次只加入一个有效区块. 虽然在高并发场景下, 链式账本允许并发块加入, 但是这些并发的块是彼此冲突的. 系统仅认可一条分叉链作为主链, 其需要加入最长链法则或者最重子树来选择分叉. 这使得系统在设定时间内只能出一个有效的区块, 多个有效区块构成一条主链, 而未进入主链的区块不会成为有效的数据, 带来了计算, 存储资源的浪费. 此外, 分叉使得部分诚实节点的区块会被加入分叉链, 无法成为主链区块, 也会影响主链的增长速度.

2.1.3 隔离见证

隔离见证是指系统将交易的签名, 脚本等数据封装至一个数据结构, 即见证, 并将见证数据和交易数据分隔开来. 如图 14 所示, 以比特币为例, 其签名, 脚本数据都存放在交易输入 (TxIn) 中. 隔离见证将签名、脚本数据单独存储在见证 (witness) 中, 使得签名数据的修改不会影响交易.

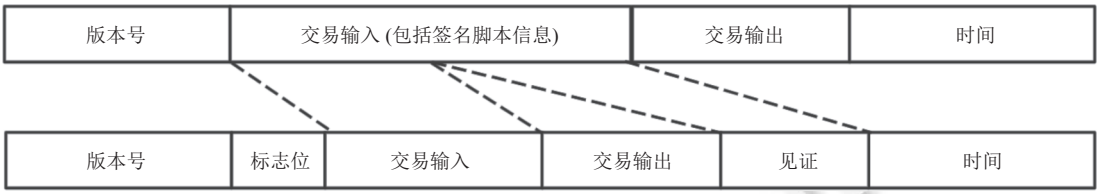

图 14 隔离见证[48]

Lombrozo 等人在比特币社区提出 BIP141 提案[48], 即隔离见证 (segregated witness). 由于系统会根据交易输入 (TxIn) 生成交易 id, 攻击者可以通过修改签名信息来改变交易 id. 隔离见证使得将签名信息隔离出来, 消除了比特币交易的延展性攻击. 此外, 隔离见证分离出签名数据, 提高了系统验证的效率. 交易数据集中可以提升检索性能. 在简单支付验证 (simplified payment verification, SPV) 中, 验证节点可以将交易的签名信息作为可选项, 减少了网络传输的数据量.

2.2 图式账本

为了尽可能减少孤块, 充分利用并发数据, 相关研究提出图式账本, 其通过有向无环图 (directed acyclic graph, DAG) 来组织区块, 形成如图 15 结构的图式账本, 实现并行记账.

图账本可表示为 $D=<V, E>$. V 表示图中的节点集合, 即区块/交易集合. 第 1 个顶点, 为创世区块, 没有出度, 有多个入度, 其他顶点允许有多个入度和出度; E 表示图中连接顶点 V 的边集. 在 D 中, E 中的边都是有向边, 即满足有向性; 而且对于 V 中任意两个顶点 u, v, 存在偏序关系 $u \leqslant v$.

此外, 图账本满足无环性, 即 DAG 中的节点 $v_1, v_2, \ldots, v_n$, 满足 $\forall i \in [1, n-1]$, $(v_i, v_{i+1}) \in E$, 存在偏序关系, 但是不存在 $(v_n, v_1) \in E$[49]. 高政风等人[49]对于 DAG 拓扑结构方面进行分析, 说明了 DAG 的无环性使得图式账本容易增加交易数据, 难以删除交易数据, 账本的所有交易可以通过其偏序关系进行定序, 即实现了易增难删、交易可定全序的拓扑性质.

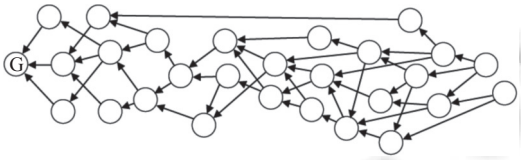

图 15 图式账本[50]

DAG 中除创世区块外的每个节点有多个入度和出度, 即所有的并发块可以加入 DAG. 节点不需要相互竞争指向同一个前驱节点, 系统无需选择分叉, 无需串行化地同步维护一条链, 实现了交易数据的异步并发写入[51]. DAG 共识使得系统并发块都会成为有效块, 对账本数据产生贡献. 因此, DAG 结构适合于高并发场景.

在不可信场景下, 链式结构为了解决分叉带来的双花问题, 会采用最长链或最重子链原则来抛弃分叉链, 选择主链. 但是分叉链中也存在无双花问题的交易, 其由诚实节点生成, 但其工作量未被认可, 造成了计算、存储资源的浪费. 而 DAG 考虑分支的工作量, 不会舍弃分支上的交易. DAG 采用乐观包容的策略, 接受分叉, 并通过交易定序机制来检测冲突交易, 从而解决双花问题. 但是, 定序机制, 如计算权重等, 会消耗节点的计算和存储资源, 影响性能提升. 分叉过多, 交易定序较复杂, 会影响交易确认的时间. 因此, 基于 DAG 的共识需要规定先序块的选择, 减少更多分叉的出现.

图式账本根据 DAG 节点的粒度, 可以分为两类, 分别是区块 DAG 和交易 DAG. 区块 DAG 是指账本中的节

点是以区块为单元. 区块 DAG 的每个出块节点需要将若干交易打包成区块. 而打包是一个消耗计算和通讯资源的过程. 打包区块会使得交易频繁进行哈希计算. 交易在加入 Merkle 树之前, 本身需要计算一次哈希, 其在 Merkle 树需要再计算若干次哈希. 最后, 生成区块时, 区块头也需要再次计算哈希. 因此, 区块 DAG 的性能开销主要存在于其打包区块的过程. 目前, 基于区块 DAG 设计的共识协议有 SPECTRE[52], PHANTOM[53]等. 北京大学技术团队研发的数瑞可信图式账本也是基于 DAG 的区块链存证系统, 其采用随机见证共识 (nRW), 通过可验证随机函数选择节点来将数据的共识和存储[54], 实现对数据全生命周期的使用过程存证和溯源[55]. Conflux[36]在 DAG 结构的基础上优化, 提出了树图结构. 每个节点会根据分叉选择协议 (如最长链, 最重子链或 GHAST[37]) 来选出中枢链, 将中枢链的最后一个节点作为父节点, 用父边连接, 保证了链的横向收敛, 还采用引用边, 用于连接其叔父节点, 表示该块与叔父块的时序关系, 有助于交易的快速排序.

交易 DAG 是指账本中的节点是以交易为单元的. 交易 DAG 将交易作为账本节点, 消除了区块打包机制. 但由于账本节点的粒度减小至交易, 每个交易可以指向多个前置交易, 交易 DAG 生成过程中会产生较多的哈希引用, 带来了系统性能开销. 2015 年 Lerner 等人提出了基于交易的图式账本 DagCoin[56], 采用确认分数机制来避免双花. 随后, Popov 等人提出 IOTA[57], 其采用后序节点的验证, 即加入交易的节点需通过马尔可夫蒙特卡洛 (MCMC) 随机游走策略来随机选出 DAG 中两个没有被直接验证的先序节点进行验证, 并将验证关系作为边. 其交易节点在 DAG 中的深度越深, 其被直接或者间接验证的次数也越多, 安全性越高. 这种验证方式缩小了验证的范围, 节约了验证的计算开销, 快速扩展了账本, 提升了系统性能. Churyumov 等人提出 Byteball[50], 基于交易 DAG 的去中心化存储系统, 其将交易作为单元, 并采用了见证人机制来对单元共识.

3 共识机制

对比单节点账本系统而言, 分布式账本系统涉及多个节点, 需要各个节点相互通信, 对于即将加入账本的区块达成一致 (即共识), 实现交易决策.

比特币账本系统的区块共识包括以下阶段, 分别是: 节点打包, 全网节点竞争出块权, 其余节点验证记账, 等待区块确认. 其节点较多, 需要大量通信, 影响系统的出块效率. 传统的竞争出块方式是 PoW 机制 (工作量证明机制), 即节点通过枚举, 进行哈希计算, 竞争工作量, 从而获得出块权.

在全网范围内的节点竞争出块时, 节点越多, 网络并发的概率越大, 账本越容易出分叉, 而账本仅认可最长链为合法链, 使得区块无法在加入账本时被立即确认. 系统需要等待若干区块, 造成区块的确认时间较长. 此外, 算力强的节点可以在不同的分叉链上生成交易, 造成双花问题. 因此, 比特币控制系统每 10 min 出块来减少并发和分叉, 抑制双花, 但是这也造成了出块速度较低. 因此, 分叉的抑制或者减少是设计区块链共识算法的关键.

本节描述了传统区块链共识的过程, 并结合其底层数据结构, 说明工作量证明共识的交易处理速度慢, 交易确认时间较长, 性能较差的问题, 并介绍了 3 种共识优化机制. 其中, 通信投票机制是指多节点相互通信, 每个节点通过多阶段的表决, 参与投票, 得出决策. 所有节点共同参与决策表决, 最终系统可以得到确定性的结果, 交易数据得以快速被确认. 见证人机制缩小了参与决策的节点数量, 节省了节点计算和通信资源, 提升系统吞吐量. 分阶段共识将共识进行阶段性划分, 并令每个节点承担不同的任务, 提升了系统的并发性, 从而影响共识性能. 后文表 5 从交易处理速度、交易确认速度、通信复杂度等方面, 对比上述 3 种机制的典型共识.

3.1 通信投票机制

通信投票机制是多节点相互通信, 对决策进行投票, 实现系统一致性决策. PBFT、Raft、PoS 等都是基于通信投票机制的共识.

BFT 类共识解决了拜占庭将军问题[58], 即在若干个节点中, 可能存在恶意节点内, 节点需要通过一致性的共识协议来做出决策.

PBFT (practical Byzantine fault tolerance)[59]是 Castro 等人提出了一种拜占庭容错 (BFT) 的共识算法. PBFT 将共识阶段分为预准备, 准备和提交 3 个阶段. 经过 3 个阶段的广播通信后, 各个节点达成一致, 再执行客户端的区

块. 在由 $n=3f+1$ 个节点组成的分布式账本网络中, PBFT 要求在准备和提交阶段, 每个节点必须广播验证和提交消息. 网络中节点收集到至少 $2f+1$ 个来自其他节点的验证或提交消息, 才可以进入下一阶段. 因此, PBFT 最多可容忍 f 个恶意节点, 其适合用于联盟链或私有链. Tendermint 也是一种 BFT 类的优化共识, 其通过两轮投票即可达成一致, 并通过锁定块来保证链不会分叉[60]. HotStuff[61]是一种 BFT 三阶段投票协议, 其采用阈值签名, 即令主节点作为签名的收集者, 创建和收集阈值签名, 当签名数量超过阈值时, 系统对消息达成一致. 收集者收集阈值签名, 使得非收集者节点之间无需广播通信, 实现了线性复杂度的网络通信, 减少了网络的通信开销. SBFT[62]也采用阈值签名的方法, 其设计了提交收集者 (C-collection) 的节点角色, 用于收集提交阶段的阈值签名, 将其整合并发送; 设计了执行收集者 (E-collection) 的节点角色, 用于收集执行阶段的门限签名, 将签名整合并发送, 也实现了线性通信复杂度的 BFT 协议. Kogias 等人提出了 Byzcoin[10], 一种将工作量证明和实用拜占庭容错协议结合的混合共识协议. 其通过工作量证明机制来生成新区块, 获得其出块节点并广播. Byzcoin 设置时间窗口, 使得系统中只有在当前时间窗口内出块的节点, 成为 PBFT 共识的委员会节点. 委员会节点通过 PBFT 协议来对区块进行投票共识. Byzcoin 通过工作量证明共识来动态选择参与 PBFT 共识的委员会节点, 降低了系统的通信开销, 也提高了 PBFT 共识的扩展性.

表 5 共识对比表

对比项	PoW	PBFT	RAFT	DPoS	dBFT	Bitcoin-NG
性能机制	—	多节点决策	多节点决策	见证人	见证人	分阶段
交易处理速度	较慢, 10 min (以比特币为例)	(取决于网络速度)	—	快, 部分节点共识, 基于随机顺序	快, 部分节点共识	快, 一次PoW竞争可以出多个块
通信复杂度	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$
交易确认速度	慢, 60 min (以比特币为例)	较快, 因为全网节点参与决策	较快, 只需要日志被同步到半数节点	较快, 因为出块快, 共识节点数量少	快, 继承了PBFT	慢, 除了需要块确认外, 在下一个key block加入账本时, 需要等待一个网络延迟时间
使用场景	公有链	联盟链, 私有链	私有链	公有链	公有链	公有链
容错问题	Byzantium	Byzantium	Crash	Byzantium	Byzantium	Byzantium
可扩展性	较好, 节点算力竞争	较差, 需多节点互相通信	较差, 多节点互相通信	较好, 存在见证节点选举	较好, 存在见证节点选举	较好, 本质是PoW

CFT 类共识 (crash fault tolerance) 是一类非拜占庭容错的共识. 其假设系统中不存在恶意节点, 仅解决节点崩溃, 宕机等问题. 因此, 其适用于联盟链或私有链环境下. Raft, Paxos 共识解决了节点崩溃情况下的共识问题.

Paxos[63]将节点分为提议者 (proposer), 决策者 (acceptor) 和学习者 (learner) 这 3 种角色, 其共识过程分为准备 (prepare) 和决策 (accept) 两部分, 每个阶段都采用了大多数 (majority) 读写机制, 保证了其 $n/2$ 的容错能力. Chubby[64], ZooKeeper[65]等分布式协调服务都借鉴了 Paxos 相关算法. Raft 是 Paxos 的一种优化算法, 其具有可理解性和易于实现的特点. Raft[66]是一种唯一领导者的协议, 其将节点分为领导者 (leader)、跟随者 (follower) 和候选者 (candidate) 这 3 种角色, 其将共识分为 2 部分, 分别是基于任期的领导节点选举和日志复制. 选举阶段通过投票的方式选出领导节点. 当领导节点被选举出来后, 客户端的所有请求都会发送至领导节点. 该领导节点会调度这些并发请求的顺序, 并且通过日志复制和复制状态机来保证领导节点与跟随着节点的状态一致性. Raft 算法要将所有操作请求转发到领导节点, 再由领导节点发日志到各个跟随者节点同步. 由于领导节点是一个单节点, 所有操作都要发到领导节点上处理, 在高并发情况下, 领导节点会成为性能瓶颈. 因此, Raft 共识往往用于节点数量较少的分片场景中, 每个片由一个 Raft 节点组来管理. 目前, HyperLedger Fabric 排序阶段支持采用 Paxos, Raft 算法来实现共识[67].

基于 BFT 的通信投票共识令全网节点共同决策, 账本很少有分叉的情况, 从而交易可以立即被确认. CFT 共识采用领导节点决策, 交易也可以很快被确认. 但是, CFT 共识不支持拜占庭容错, 其仅考虑节点宕机、网络异常

等问题, 适合用于较为安全的私有链, 联盟链环境下.

3.2 见证人机制

在分布式账本系统中, 随着参与共识的节点增多, 系统消耗的通讯, 计算资源越多, 使得吞吐量呈现下降趋势. 为了解决多节点带来的性能问题, 很多共识采用了见证人机制, 即采用部分节点, 作为见证人, 参与共识决策. 通过牺牲去中心化来提升性能. 常见的见证人共识有 DPoS[68], dBFT 共识[69].

DPoS (delegated proof of stake) 共识[68]是指系统通过节点选举投票, 选出部分节点, 即见证人节点, 由这些见证人节点按照随机顺序陆续出块记账. 见证人节点出块后, 会将区块广播, 其余节点验证记账. 对于分叉问题, DPoS 采用多数见证人确认的机制, 即待块后续 2/3 以上的见证人节点确认交易后, 该区块的交易被完全确认, 且不可逆. dBFT (delegated Byzantine fault tolerance) 共识是指系统中的代币持有节点通过投票, 选出若干共识节点, 再由这些共识节点通过 PBFT 共识出块[69]. 一方面, dBFT 缩小了共识节点的范围, 减少了节点通信的次数. 另一方面, dBFT 继承了 PBFT 共识的优势, 可以快速确认区块.

见证人机制减少了出块节点的数量, 节点通信和计算竞争较少, 提升了系统性能. 但在高并发情况下, 由于共识节点数量较少, 资源有限, 共识节点会造成性能瓶颈.

DPoS 算法出块节点的出块顺序是按照既定的随机顺序. 因此, 出块节点之间不存在竞争, 节省了节点竞争的时间和资源开销, 出块速度较快. dBFT 共识对共识和记账阶段进行划分, 不同节点负责不同的角色, 使得节点的工作量减少, 效率更高. 而且, 其继承了 PBFT 共识无需交易确认的特点, 共识性能较好.

EOS 项目采用了 DPoS 共识, 通过投票, 选出 21 个有记账权的见证人节点, 其允许每个见证人节点连续出 6 个块, 平均 0.5 s 出一个块[70]. 为了提升区块的确认时间, EOS 还采用 BFT 共识来决定见证人的出块顺序, 将交易的确认时间减少至 1 s[70]. NEO 项目采用 dBFT 共识, 通过代理投票来实现见证人共识机制[69].

3.3 分阶段机制

分阶段机制是将共识划分为打包, 竞争, 出块, 记账等若干子阶段, 一方面可以基于阶段进行调整优化, 另一方面, 可以令节点执行不同的阶段任务, 阶段任务并行执行, 从而提升系统并发度. 图 16 展示了多阶段任务并行执行优化.

(1) 阶段优化

Bitcoin-NG 采用关键块 (key block) 和微块 (micro block)[71], 其中关键块用于节点选举, 微块负责记录账本信息. Bitcoin-NG 将其出块过程拆分为打包关键块、竞争出关键块, 打包微块等阶段, 实现了一次竞争出多个块, 提升了出块效率.

Bitcoin-NG 的节点先通过打包关键块, 进行 PoW 共识. 竞争出关键块的节点获得当前一轮时间的出块权. 接下来, 获得出块权的节点负责在当前一轮时间内生成微块, 直到下一个出关键块的节点出现. 其他节点负责验证区块, 记账, 并参与到下一轮关键块的出块竞争. Bitcoin-NG 中关键块的出块可以对应于传统共识的选举过程, 而微块的出块可以对应于记账过程. 当一个关键块出块后, 该出块节点会生成多个微块, 直到下一个关键块出现. 该设计使得系统节点仅做一次工作量证明, 便可以生成多个账本区块, 从而减少了竞争计算的频率, 提升了性能.

此外, Bitcoin-NG 中的当前一轮时间 epoch 是指一个关键块的出现到下一个关键块出现的时间段. 其出块过程可以看作是多个 epoch, epoch 之间可以并行, 即节点在完成记账的同时, 可以参与工作量证明, 竞争下一个 epoch 的出块权. 进一步提升了共识的性能. 但是, 当关键块出块频繁, 并且记账节点快速更换时, 由于网络存在延迟, 上一个 epoch 的微块会经过较长时间后, 才能到达账本其他节点, 造成账本其他节点存在分叉. 因此, 在每个关键块加入账本时, Bitcoin-NG 会等待一个网络延迟时间, 即等待上一个 epoch 中网络延迟的微块加入账本. 这也会增加了交易的平均确认时间.

(2) 节点角色设计

dBFT 和 Algorand 将共识进行阶段性划分, 并令节点负责不同子阶段的任务, 即具有不同的角色, 使得不同任务的节点可以并行工作. dBFT 令选举出来的一部分节点负责打包和竞争出块任务, 另一部分节点负责验证记账任务. Algorand[72]将随机选出的领导节点负责打包区块, 另一部分节点通过改进的二元拜占庭协议来对区块验证和共识.

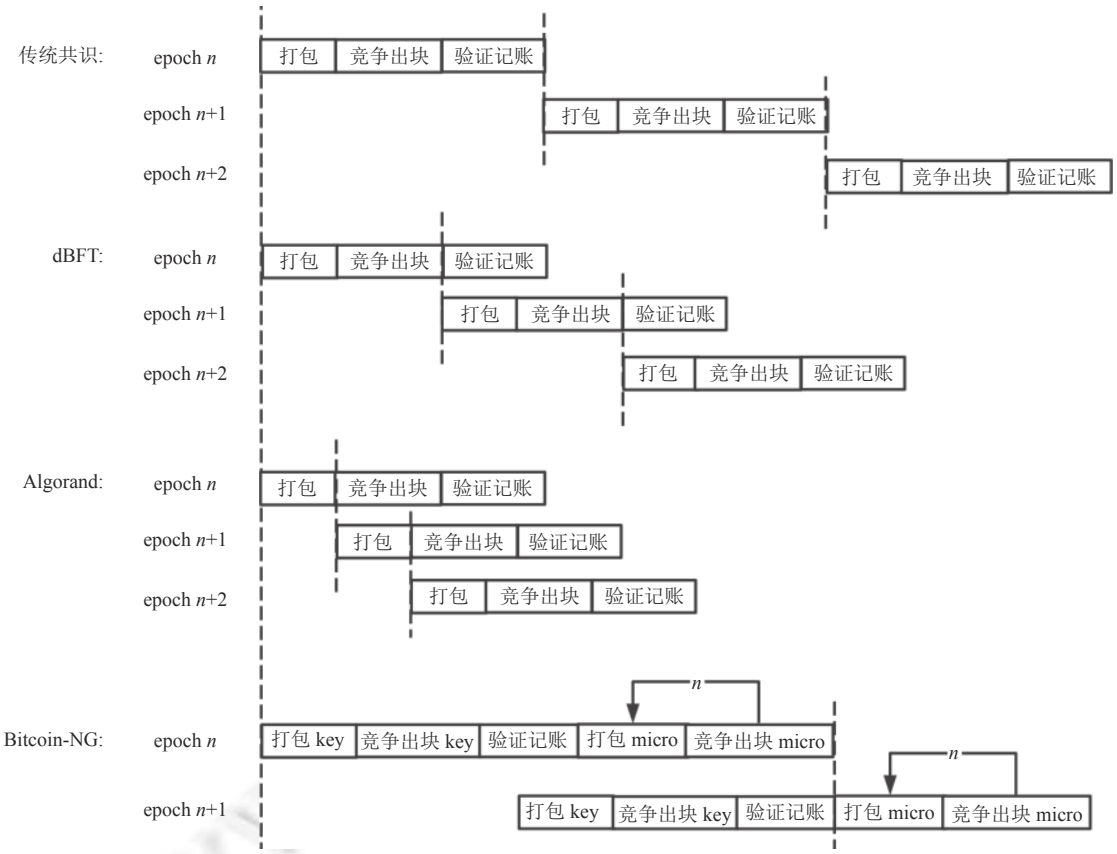

图 16 分阶段共识

令 PoW 共识的打包时间为 P, 竞争出块时间为 C, 验证记账时间为 V. PoW 共识平均出块时间为 $CT=P+C+V$, 其交易的确认时间为 $6CT$. 假设 dBFT 和 Algorand 的打包方式、打包数据大小、竞争出块难度近似于 PoW 共识, 并令 n 为 epoch 数量, 那么 dBFT 的平均出块时间 DT、Algorand 的平均出块时间 AT 分别为:

$$DT=((P+C)\times n+V)/n=P+C+V/n \tag{7}$$

$$AT=(nP+C+V)/n=P+C/n+V/n \tag{8}$$

可见, 在多轮出块的情况下, 上述项目的出块时间开销是小于传统工作量共识的时间, 具有较好的交易处理能力.

4 消息通讯

分布式账本系统底层是一个去中心化的 P2P 的网络. 每个节点有对应的地址, 节点之间相互传输消息, 无需中心节点的协调. 任意一个节点受到攻击或者宕机, 都不会对系统产生影响. 由于系统需要进行共识、交易广播等过程, 底层网络需要大量通信. 消息通讯的优化可以大幅提升分布式账本系统的性能.

系统中消息通讯的性能优化往往是通过减少底层网络通信的频率、压缩网络通信的数据、构建消息分发网络来提升性能.

4.1 消息路由

分布式账本消息传送的简单路由方式是洪泛法 (flooding)[73], 即每个节点在收到消息之后, 会发送给其连接路径上的节点. 若节点收到重复消息, 则冗余消息会被丢弃. 频繁的洪泛会有大量数据传输, 造成广播风暴.

比特币采用 Gossip 协议来进行传播. Gossip 协议不会将数据广播到其所有关联的节点, 而是随机选几个节点广播, 直到所有节点收到该消息. Gossip 避免了消息在同一时刻发送多个节点, 但该方式仍会在网络中产生大量的

冗余消息.

Kademila 协议[74]通过分布式散列表 (DHT), 实现结构化的 P2P 网络. Kademila 协议会为网络节点分配 id, 根据节点 id, 网络呈现二叉树结构, 其中每个节点维护一个 DHT, 用于节点消息转发. 节点之间根据已有的散列表和节点 id 的异或距离进行查找. 节点之间在查找时会交换部分节点信息, 使得系统可以快速准确查找地址.

Kademlia 支持节点的精确查找, 不会产生较多的冗余信息, 减少了网络中的数据传输. 其不会随着节点数量的增加而提升通信开销. Kademlia 算法的通信复杂度为 O(log n), 通信开销较少, 避免了网络拥堵. 目前, 该算法已被应用于以太坊、RapidChain 等项目中.

4.2 消息处理

消息处理主要包括了两部分. 一方面, 系统可以消除不必要的冗余信息, 消除网络带宽. 另一方面, 系统可以采用密码学方法进行消息压缩, 进一步减小网络中的数据.

(1) 冗余处理

传统分布式账本的交易消息提交到网络后, 会先通过 P2P 网络广播到各个节点, 然后每个节点将交易存放至其内存池中. 当网络中的节点竞争到出块权时, 该节点会将交易打包成区块, 同步至全网节点. 在这个过程中, 交易在网络中被传输两次, 一次是作为首次提交至网络的交易被网络传输, 另一次是作为区块的交易被全网广播, 可见交易传输是存在冗余的.

Corallo 在比特币论坛中提出致密区块中继 (compact block relay)[75] , 即比特币提案 BIP152. 致密区块不再采用传统的完整区块结构, 而是采用由交易 ID 列表组成的数据结构. 如图 17 所示, 在高带宽模式下, 节点 A 收到区块后, 无需验证, 将致密区块发送至节点 B. 节点 B 依据交易列表, 再向 A 请求缺失交易. 而在低带宽模式下, 节点 A 需要在验证区块的有效性后, 才可以传递区块头和致密区块.

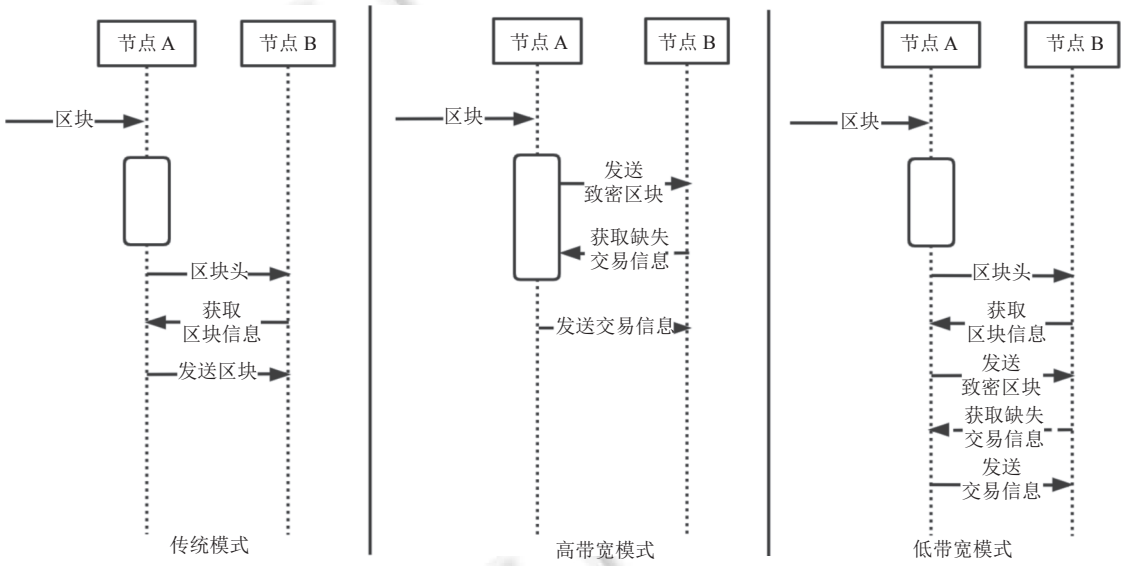

图 17 致密区块中继[75]

在比特币中, 一个完整交易的大小约是 400–600 B[76], 而交易 id 仅有 32 B[75]. 致密区块合理利用了内存的信息, 用交易 id 来代替交易的详细数据, 减少了网络中传输的交易数据, 提高系统的性能. 此外, 在低带宽环境下, 发送节点在转发区块之前, 会验证区块有效性, 会过滤部分无效区块, 减少了冗余消息.

(2) 数据压缩

分布式账本可以通过二次哈希、布隆过滤器、可逆式布鲁姆查找表等密码学技术来实现数据压缩, 减少网络带宽, 提升网络数据传递的性能.

● 哈希压缩

哈希压缩是指在致密区块交易 id 的基础之上, 节点再计算轻量级哈希, 限制交易字符串的长度, 获得新交易 id, 即 TXID-HASH = hash(TXID), 实现压缩. 由于哈希碰撞, 一个 TXID-HASH 会对应多个交易. 接收节点会将收集所有匹配的交易 ID 作为下一阶段的候选集. 接收节点通过排序候选集交易、遍历候选集来重新计算 Merkle 树根, 从而找到正确的交易.

二次哈希在 BIP512 的基础上压缩了更多的传输数据, 提升区块链网络的性能, 但是, 二次计算哈希还需要接收节点做少量排序, 并枚举碰撞, 带来了一定的计算开销.

Ding 等人提出了 Txilm[77]算法, 即 Monoxide 公链系统底层的区块压缩算法, 采用了二次哈希的思想. 为了减少计算开销, Txilm 采用了 CRC32 或 CRC64 作为二次哈希的算法, 产生 32–64 B 的哈希值, 并且采用 CRC32-Merkle 树, 提升节点的计算速度[77]. 此外, Txilm 采用规范交易排序规则 (CTOR) [77], 将区块和内存池中的交易按照哈希来排序, 使得内存池中有二义性的哈希 id 相邻, 降低了解决二义性的计算成本, 支持更短的哈希和更高的压缩比.

● 可逆式布鲁姆查找表

可逆式布隆查找表 (IBLT)[78]是通过哈希将所有的交易数据处理, 并将处理后的数据异或到一个固定大小的数据结构. IBLT 是固定大小的, 不会随着交易数据的增加而增加. 没有交易数据的节点是无法读取和破译 IBLT, 而那些有交易数据的节点, 可以通过将其自己的交易生成到一个 IBLT, 通过将两个 IBLT 比对, 抵消掉相同的交易, 可以获得两个 IBLT 中交易集合的差异.

为了提升 IBLT 的比对效率, 分布式账本系统往往采用布隆过滤器来缩小接收节点的交易集合. 布隆过滤器 (Bloom filter)[79]由一个二进制位数组和一系列的哈希函数组成. 初始时, 其二进制位数组全部为 0, 当给定一个待查询的元素时, 该元素经过一系列哈希函数计算会映射出一系列的值, 并将所有值在位数组的偏移量处置为 1. 布隆过滤器会存在误判的情况, 即会将可能不在该集合中交易放进来, 因此它可以缩小接收节点交易的范围.

Ozisik 等人提出了 Graphene[80], 结合布隆过滤器和可逆式布隆查找表 (invertible Bloom lookup table, IBLT), 减少区块链交易通信数据. Graphene 的发送节点会为区块内交易数据生成一个 IBLT 和一个布隆过滤器, 并将其发送给接收节点. 接收节点会先用布隆过滤器来有效过滤内存池中的集合, 再用 IBLT 来计算 2 个节点之间区块交易的差异, 减少了网络中交易的二次传播.

该方法将网络传输的交易 id 列表进行哈希计算, 将其压缩到固定大小的 IBLT. 接收节点则利用了接收节点内存池中的交易数据, 减少了通讯的传输数据. 但该方法需要对交易 id 列表进行计算, 生成布隆过滤器和可逆式布鲁姆查找表. 接收节点还需要过滤遍历内存池交易, 计算并比对发送节点和接收节点的 IBLT, 这会消耗一定的计算资源. 在安全方面, 该方法没有直接在网络中传输交易 id 列表, 保证了数据的安全传输. IBLT 使得没有交易数据的节点无法根据 IBLT 来读取和破译交易信息. 表 6 对比了致密区块、Graphene 和 Txilm 这 3 种消息处理机制.

表 6 消息处理对比表

对比项	致密区块 (BIP512)	Graphene	Txilm
方法	致密区块	可逆式布隆查找表、布隆过滤器	二次哈希、Merkle tree枚举
传输内容	块头、交易id列表	IBLT、Bloom filter	Merkle tree root、TXID-HASH 列表
压缩倍数	10	—	80
引入新数据结构	未引入	引入	未引入

4.3 消息分发

区块链分发网络可追溯到内容分发网络 (content distribution network, CDN). CDN 是将多个服务器节点连接形成的网络平台, 为用户提供内容的分散存储和高速缓存服务, 实现内容的快速、稳定分发. 区块链分发网络 (blockchain distribution network, BDN) 将多个节点组成中继网络, 为区块链节点提供中继转发、交易和区块的缓存以及快速分发服务. BDN 将分布式账本的网络通信解耦, 节点无需关心消息通讯, 仅需要通过网关接入到网络中, 将信息传递到中继节点即可, 中继节点负责 P2P 网络通信数据的存储和转发.

bloXroute 是 Kuzmanovic 等人提出的第 1 个区块链分发网络, 可以支持 BitcoinCash、Conflux、Ethereum 等多种区块链网络[81]. 如图 18 所示, 不同于传统 P2P 网络, BDN 网络由分布式账本节点、网关、中继服务器 3 部分构成. 每个需要接入网络的分布式账本节点, 都需要通过网关接入. 中继服务器负责路由转发. 当交易在网络中首次传播时, 交易会经过网关到达中继服务器, 中继会为交易分配一个 sid, 生成一个交易/sid 对, 并将交易/sid 对发送至其他中继服务器和网关. 其他中继服务器和网关会缓存并且转发该交易/sid 对. 当区块生成后, 网关会将会区块的交易映射成 sid, 并将其广播至所有区块链节点. 区块链节点的网关会将 sid 反向映射回交易, 恢复成区块. 如果网关中没有存储某个 sid, 网关会请求中继服务器, 求得交易/sid 对. 此外, Marlin 也是一种分布式账本的中继网络, 其通过经济激励机制, 对网络中传输数据的节点进行奖励划分, 使得发送信息越多越快的节点可以得到越多的奖励[82].

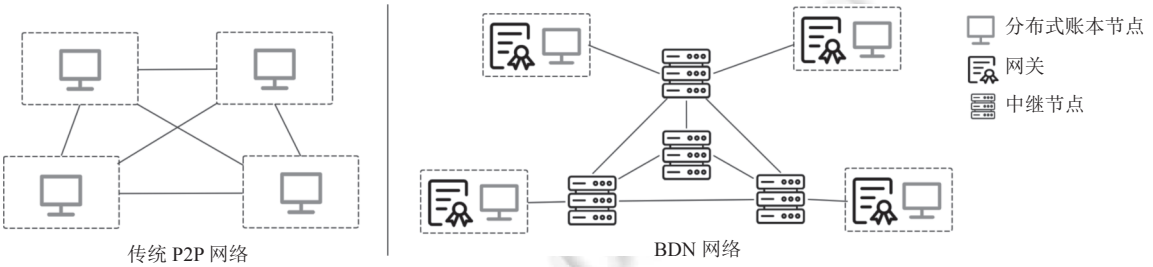

图 18 分布式账本网络

区块链分发网络通过建立中继服务器, 建立网络层, 将网络通信解耦, 减少了分布式节点的网络通信开销. bloXroute 通过生成交易映射, 并建立生成中继和网关的缓存, 令交易信息通过轻量 sid 的方式在网络中传输, 提高了传输效率. 此外, bloXroute BDN 网络不需要每个中继节点在接收到区块数据时校验其有效性, 即不需要块的跳跃性传播. bloXroute BDN 网络通过节点流水式发送每一个块的数据包, 不需要节点等待整个块的信息. 在数据流水式发送的过程中, 仅有区块链节点收到的块和交易的信息会被验证, 无需其他中继验证, 提高了消息传播的效率.

5 总结与展望

分布式账本具有广阔的应用前景. 针对账本系统的性能瓶颈问题, 本文从系统设计的角度, 对分布式账本性能优化的方法总结分类, 从体系架构设计, 账本数据结构, 共识机制和消息通讯 4 个方面来分析各个方法的关键技术、代表性的解决方案以及其对性能的影响.

在体系架构设计上, 主要有节点分片、多链、交易流程优化、链上链下混合负载 4 种典型的设计优化技术: 采用分片技术对节点进行分组组织, 减少冗余数据、提升系统并行度; 引入多链结构分担系统负载, 通过侧链、公证人模式等机制, 实现同构/异构多条链之间的互联互通; 采用基于节点角色的任务分配、流程并行化等方式优化交易处理流程; 采用通道、交易打包等技术, 将链上交易负载转移至链下, 减少了链上的交易处理量. 账本架构设计的性能优化主要是通过转移负载、调整交易处理流程来提升系统性能. 但是分片、流程优化等性能优化技术难以对于已有账本进行改造, 开发成本较高. 多链和链下设计难以保证数据的安全性, 会为系统带来较大的安全隐患. 未来需进一步考虑其安全性设计.

其次, 在数据结构设计上, 主要有链式和图式两种结构: 链式账本通过增加块的大小、减小出块时间等方式来提高系统出块数量, 提升交易处理量; 图式账本可以提高系统的并发操作能力, 提升有效交易的记账效率. 账本数据结构的优化方法是通过修改系统数据设置、完善数据结构设计, 从而提升系统的记账能力, 使得账本系统记录更多的交易数据. 其中, 调整出块大小、出块时间等链式账本的优化方法对性能的提升有限; 图式账本的设计可以显著提升性能, 但是其乐观接受分叉数据, 使得账本数据的安全性有待提升和优化.

在共识机制设计上, 性能优化方法主要包括通信投票机制、见证人机制和分阶段机制: 通信投票机制通过极大地减少账本分叉, 实现交易的快速确认; 见证人机制将参与共识的节点范围缩小, 减少了计算和通信开销; 分阶段机制将共识过程进行划分, 令不同节点负责不同子过程, 使得不同任务的节点可以并行工作. 共识优化帮助分布

式账本系统进行快速决策, 决定记录数据内容、记账权的归属. 通信投票机制有效地减少交易确认时间, 但其依赖于节点之间的交互通信, 会影响系统的交易处理效率. 见证人和分阶段机制通过进行节点分工、阶段划分提升了系统的交易处理效率, 但其难以保证交易的快速确认.

在消息通讯设计上, 主要是从消息路由、处理、分发方面优化, 通过减少底层网络通信的频率、消除网络中数据传输量、构建消息分发网络等方式, 减少通信开销. BDN 还将数据传输和上层逻辑解耦, 降低了上层逻辑的开发和优化难度. 但是其往往会带来系统计算、存储资源的开销.

目前, 分布式账本技术、系统及应用尚处于发展的初期, 其性能研究还面临着以下几个方面的挑战.

(1) 性能评价指标及评估方法

目前分布式账本系统的通用性能评价指标主要是交易处理数量和交易延迟时间. HyperLedger 的性能和规模工作小组[83]发布了区块链性能指标白皮书, 其中定义了性能评估中与平台无关的关键指标, 主要包括读延迟、读吞吐量、交易延迟和交易吞吐量. Zheng 等人[6]通过分析区块链的交易流程, 提出区块链性能的整体和详细指标, 其整体性能指标包括平均响应延迟、CPU 平均交易数、内存每秒交易数等, 详细性能指标包括合约执行时间、共识时间、RPC 响应率等.

针对性能指标, 主要通过在线性能监控、负载模拟测试等方法采集性能数据, 建立定性/定量的性能评价模型. 例如, Zheng 等人[6]采用日志收集方法, 实现了一个性能监控框架. 该框架为每个区块链节点创建日志解析器, 其通过为验证节点创建守护进程, 收集和解析区块链的日志信息, 并提供了个性化接口, 方便数据收集. 此外, 该工作还设计了数据收集器, 用于收集验证节点的日志数据, 并计算性能指标. 与 RPC 数据收集方法相比, 日志收集方法具有低开销、可扩展的特点. 该工作通过对以太坊, Parity, HyperLedger Fabric 和 CITA 共 4 个主流区块链系统进行日志监控, 使用该框架获取了吞吐量、CPU 平均交易数等性能指标信息, 并与 RPC 数据收集的方法对比, 说明了该性能监控框架对于性能测试的有效性和灵活性. Blockbench[84]提供了基准测试框架, 通过 IWorkloadConnector 接口来集成被测的区块链系统, 实现应用部署、加载工作负载、交易监控, 以及状态查询等功能, 获得性能参数.

已有的性能评价指标及评估方法多局限于特定区块链项目、通用的性能指标, 尚难以针对分布式账本的技术特点, 度量和评价系统性能, 识别性能瓶颈, 分析、比较不同分布式账本之间的性能差异.

(2) 性能基准测试集

性能基准测试采用具有相似测试目的测试集, 通过自动化测试工具来模拟多种正常、峰值以及异常请求条件来对不同系统的各项性能指标进行测试, 使得测试者更全面地了解系统性能情况, 检测出系统设计中的性能瓶颈.

Dinh 等人[84]提出了 Blockbench 私有链测试框架, 测试了 Ethereum、Parity、HyperLedger Fabric 这 3 种私有链的性能. Blockbench 将请求负载分为两类, 分别是用于评估区块链应用层性能的宏基准测试和用于测试区块链执行层、数据层、共识层性能的微基准测试. Blockbench 为所有请求负载设计了不同测试程序, 采用智能合约的模式, 加载和运行测试程序, 其中 YCSB 实现 key-value 存储, 用于评估系统中的 NoSQL 数据库、Smallbank 实现了转账功能、EtherId 实现了域名注册的功能、IOHeavy 实现了随机读写、CPUHeavy 实现了大数组的快速排序. 该工作采用基于 YCSB、Smallbank、EtherId 的负载进行宏性能测试, 测试了 Ethereum, Parity, HyperLedger Fabric 这 3 个区块链系统的吞吐量、延迟、扩展性、容错和安全性; 采用 IOHeavy、CPUHeavy 的负载, 对区块链项目的执行、数据、共识层进行微基准测试, 发现 Parity 中服务器的交易签名造成了系统性能瓶颈等问题, 表明了当前的区块链不适合大规模数据处理工作.

基准测试设计需遵循可测量、可重复、可对比等设计原则. 目前, 大多数分布式账本系统性能测试的工作负载尚具有一定的局限性, 典型应用场景选取及其负载特性的分析、负载模拟和测试集的构建, 是账本性能研究的一个重要方向.

(3) “不可能三角形问题”

分布式账本技术与很多的分布式系统一样, 存在一个“不可能三角形”, 即去中心化、性能、安全性这 3 方面的组合无法同时达到最优[20]. 如图 19 所示, 以比特币系统、分片技术和基于公证人的共识机制为例, 显示了不同的设计权衡策略: 比特币侧重去中心化和安全性设计, 降低了交易处理性能[85]; 分片保证了系统的去中心化和高性能, 但

是却造成片内诚实节点的算力被稀释, 降低了安全性; 公证人模式在第三方可信的情况下, 其安全性和性能较好, 但却依赖于第三方公证人节点, 无法保证完全去中心化. 性能优化方法会不同程度地影响系统的安全性、去中心化. 分片设计的交易映射、跨片通信、分片重组等机制、多链设计的跨链链接通信机制、链上链下混合模式中的链下交易、链式账本的最长链策略、图式账本的交易定序问题等性能优化技术, 都容易引入安全漏洞, 成为安全性攻击的脆弱环节. 因此, 性能设计需与分布式设计、安全性设计综合考虑, 并针对特定的应用场景需求进行权衡优化.

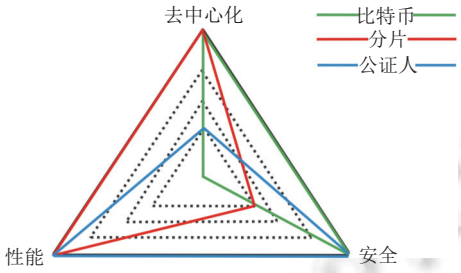

图 19 分布式账本不可能三角[85]

References:

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/en/bitcoin-paper

[2] Wood G. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 2014, 151: 1–32.

[3] Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, de Caro A, Enyeart D, Ferris C, Laventman G, Manevich Y, Muralidharan S, Murthy C, Nguyen B, Sethi M, Singh G, Smith K, Sorniotti A, Stathakopoulou C, Vukolić M, Cocco SW, Yellick J. HyperLedger Fabric: A distributed operating system for permissioned blockchains. In: Proc. of the 13th EuroSys Conf. Porto: ACM, 2018. 30. [doi: 10.1145/3190508.3190538]

[4] Dai HN, Zheng ZB, Zhang Y. Blockchain for Internet of Things: A survey. IEEE Internet of Things Journal, 2019, 6(5): 8076–8094. [doi: 10.1109/JIOT.2019.2920987]

[5] Zheng ZB, Xie SA, Dai HN, Chen XP, Wang HM. An overview of blockchain technology: Architecture, consensus, and future trends. In: Proc. of the 2017 IEEE Int'l Congress on Big Data (BigData Congress). Honolulu: IEEE, 2017. 557–564. [doi: 10.1109/BigDataCongress.2017.85]

[6] Zheng PL, Zheng ZB, Luo XP, Chen XP, Liu XZ. A detailed and real-time performance monitoring framework for blockchain systems. In: Proc. of the 40th IEEE/ACM Int'l Conf. on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP). Gothenburg: IEEE, 2018. 134–143.

[7] Zheng XY, Zhu YX, Si XM. A survey on challenges and progresses in blockchain technologies: A performance and security perspective. Applied Sciences, 2019, 9(22): 4731. [doi: 10.3390/app9224731]

[8] Kuzlu M, Pipattanasomporn M, Gurses L, Rahman S. Performance analysis of a HyperLedger Fabric blockchain framework: Throughput, latency and scalability. In: Proc. of the 2019 IEEE Int'l Conf. on Blockchain (Blockchain). Atlanta: IEEE, 2019. 536–540. [doi: 10.1109/Blockchain.2019.00003]

[9] Hao Y, Li Y, Dong XH, Fang L, Chen P. Performance analysis of consensus algorithm in private blockchain. In: Proc. of the 2018 IEEE Intelligent Vehicles Symp. (IV). Changshu: IEEE, 2018. 280–285. [doi: 10.1109/IVS.2018.8500557]

[10] Kokoris-Kogias E, Jovanovic P, Gailly N, Khoffi I, Gasser L, Ford BA. Enhancing Bitcoin security and performance with strong consistency via collective signing. In: Proc. of the 25th USENIX Conf. on Security Symp. Austin: USENIX Association, 2016. 279–296.

[11] Lepore C, Ceria M, Visconti A, Rao UP, Shah KA, Zanolini L. A survey on blockchain consensus with a performance comparison of PoW, PoS and pure PoS. Mathematics, 2020, 8(10): 1782. [doi: 10.3390/math8101782]

[12] Zhang PY, Song J. Research advance on efficiency optimization of blockchain consensus algorithms. Computer Science, 2020, 47(12): 296–303 (in Chinese with English abstract). [doi: 10.11896/jsjkx.200700020]

[13] Zhou QH, Huang HW, Zheng ZB, Bian J. Solutions to scalability of blockchain: A survey. IEEE Access, 2020, 8: 16440–16455. [doi: 10.1109/ACCESS.2020.2967218]

[14] Mao ZL, Liu YN, Sun HP, Chen Z. Research on blockchain performance scalability and security. Netinfo Security, 2020, 20(3): 56–64 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-1122.2020.03.008]

[15] Luu L, Narayanan V, Zheng CD, Baweja K, Gilbert SL, Saxena P. A secure sharding protocol for open blockchains. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 17–30. [doi: 10.1145/2976749.2978389]
[16] Al-Bassam M, Sonnino A, Bano S, Hrycyszyn D, Danezis G. Chainspace: A sharded smart contracts platform. arXiv:1708.03778, 2017.
[17] Zamani M, Movahedi M, Raykova M. RapidChain: Scaling blockchain via full sharding. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 931–948. [doi: 10.1145/3243734.3243853]
[18] Wang JP, Wang H. Monoxide: Scale out blockchain with asynchronous consensus zones. In: Proc. of the 16th USENIX Conf. on Networked Systems Design and Implementation. Boston: USENIX Association, 2019. 95–112.
[19] What is blockchain sharding? 2021. https://thepalmtree.network/en/what-is-blockchain-sharding/
[20] Kokoris-Kogias E, Jovanovic P, Gasser L, Gailly L, Syta E, Ford B. OmniLedger: A secure, scale-out, decentralized ledger via sharding. In: Proc. of the 2018 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2018. 583–598. [doi: 10.1109/SP.2018.000-5]
[21] Antchain Introduction. 2020. https://antchain.antgroup.com/docs/11/171879
[22] Thunder Chain. 2021. https://blockchain.xunlei.com/site/docnew.html#11
[23] NEAR Protocol Specification. 2020. https://nomicon.io/Architecture.html
[24] Pagh R, Rodler FF. Cuckoo hashing. Journal of Algorithms, 2004, 51(2): 122–144. [doi: 10.1016/j.jalgor.2003.12.002]
[25] WeCross. 2021. https://gitee.com/WeBank/WeCross
[26] Back A, Corallo M, Dashjr L, Friedenbach M, Maxwell G, Miller A, Poelstra A, Timón J, Wuille P. Enabling blockchain innovations with pegged sidechains. 2014. https://www.blockstream.com/sidechains.pdf
[27] The launch of the liquid network. 2018. https://blockstream.com/2018/10/10/en-liquid-launch/
[28] Lerner SD. RSK. RootStock Core Team. White Paper, 2015.
[29] Lerner SD. RSK, 2015.https://academy.rsk.dev.br/courses/dev/03/rsk
[30] Thomas S, Schwartz E. A protocol for interledger payments. 2015. https://interledger.org/interledger.pdf
[31] Alt chains and atomic transfers. 2021. https://bitcointalk.org/index.php?topic=193281.0
[32] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework. White Paper. 2016.
[33] Kwon J, Buchman E. A network of distributed ledgers. Technical Report, Cosmos, 2018. 1–41.
[34] Poon J, Buterin V. Plasma: Scalable autonomous smart contracts. White Paper, 2017. 1–47.
[35] Qi J, Chen XS, Jiang YP, Jiang JY, Shen TX, Zhao SX, Wang S, Zhang G, Chen L, Au MHA, Cui HM. BIDL: A high-throughput, low-latency permissioned blockchain framework for datacenter networks. In: Proc. of the 28th ACM SIGOPS Symp. on Operating Systems Principles. ACM, 2021. 18–34. [doi: 10.1145/3477132.3483574]
[36] Li CX, Li PL, Zhou D, Xu W, Long F, Yao A. Scaling Nakamoto consensus to thousands of transactions per second. arXiv:1805.03870, 2018.
[37] Li CX, Li PL, Zhou D, Yang Z, Wu M, Yang G, Xu W, Long F, Chi-Chuih A. A decentralized blockchain with high throughput and fast confirmation. In: Proc. of the 2020 USENIX Annual Technical Conf. USENIX ATC 20. 2020. 515–528.
[38] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments. 2015. https://www.readkong.com/page/the-bitcoin-lightning-network-scalable-o-chain-instant-8521874
[39] Raiden Network 2.0.0 Documentation. https://raiden-network.readthedocs.io/en/stable/
[40] Roll_up. 2018. https://github.com/barryWhiteHat/roll_up
[41] Rollups. 2022. https://ethereum.org/en/developers/docs/scaling/layer-2-rollups/#rollups
[42] Teutsch J, Reitwießner C. Truebit: A scalable verification solution for blockchains. arXiv:1908.04756, 2019.
[43] Bitcoin Unlimited FAQ. 2022. https://www.bitcoinunlimited.info/faq/what-is-bu
[44] Garzik J. bip-0102.mediawiki. 2015. https://github.com/bitcoin/bips/blob/master/bip-0102.mediawiki
[45] Reiff N. Bitcoin vs. Bitcoin Cash: What is the difference? 2022. https://www.investopedia.com/tech/bitcoin-vs-bitcoin-cash-whats-difference/
[46] Kiayias A, Panagiotakos G. Speed-security tradeoffs in blockchain protocols. IACR Cryptology ePrint Archive, 2015, 2015: 1019.
[47] Sompolinsky Y, Zohar A. Secure high-rate transaction processing in Bitcoin. In: Proc. of the 19th Int'l Conf. on Financial Cryptography and Data Security. San Juan: Springer, 2015. 507–527. [doi: 10.1007/978-3-662-47854-7_32]
[48] Segregated Witness. 2015. https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
[49] Gao ZF, Zheng JL, Tang SY, Long Y, Liu ZQ, Liu Z, Gu DW. State-of-the-art survey of consensus mechanisms on dag-based distributed ledger. Ruan Jian Xue Bao/Journal of Software, 2020, 31(4): 1124–1142 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5982.htm [doi: 10.13328/j.cnki.jos.005982]
[50] Churyumov A. Byteball: A decentralized system for storage and transfer of value. 2016. https://byteball.org/Byteball.pdf

[51] Cao Y, Zhang C, Ding ZY, Jiang XW. Blockchain Technology on DAG: Principle and Practice. Beijing: Machinery Industry Press, 2018 (in Chinese).

[52] Sompolinsky Y, Lewenberg Y, Zohar A. SPECTRE: A fast and scalable cryptocurrency protocol. IACR Cryptology ePrint Archive, 2016: 1159.

[53] Sompolinsky Y, Wyborski S, Zohar A. PHANTOM and GHOSTDAG: A Scalable Generalization of Nakamoto Consensus. In: Proc. of the 3rd ACM Conf. on Advances in Financial Technologies. 2021. 51–70. [doi: 10.1145/3479722.3480990]

[54] Wu Yi. Design and implementation of a consortium ledger consensus based on DAG technology [MS. Thesis]. Beijing: Peking University, 2020 (in Chinese with English abstract).

[55] BDLedger White paper. 2021. https://public.internetapi.cn/?dir=docs

[56] Lerner SD. DagCoin: A cryptocurrency without blocks. White Paper, 2015.

[57] Silvano WF, Marcelino R. Iota tangle: A cryptocurrency to communicate Internet-of-Things data. Future Generation Computer Systems, 2020, 112: 307–319. [doi: 10.1016/j.future.2020.05.047]

[58] Lamport L, Shostak R, Pease M. The Byzantine generals problem. In: Malkhi D, ed. Concurrency: The Works of Leslie Lamport. San Diego: ACM Books, 2019. 203–226.

[59] Castro M, Liskov B. Practical Byzantine fault tolerance. In: Proc. of the 3rd Symp. on Operating Systems Design and Implementation. New Orleans: USENIX Association, 1999. 173–186. [doi: 10.1145/3335772.3335936]

[60] Buchman E. Tendermint: Byzantine fault tolerance in the age of blockchains. Technical Report, Guelph: University of Guelph, 2016.

[61] Yin M, Malkhi D, Reiter MK, Gueta GG, Abraham I. HotStuff: BFT consensus with linearity and responsiveness. In: Proc. of the 2019 ACM Symp. on Principles of Distributed Computing. Toronto: ACM, 2019. 347–356. [doi: 10.1145/3293611.3331591]

[62] Gueta GG, Abraham I, Grossman S, Malkhi D, Pinkas B, Reiter M, Seredinschi DA, Tamir O, Tomescu A. SBFT: A scalable and decentralized trust infrastructure. In: Proc. of the 49th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN). Portland: IEEE, 2019. 568–580. [doi: 10.1109/DSN.2019.00063]

[63] Lamport L. Paxos made simple. ACM Sigact News, 2001, 32(4): 18–25.

[64] Burrows M. The Chubby lock service for loosely-coupled distributed systems. In: Proc. of the 7th Symp. on Operating Systems Design and Implementation. Seattle: USENIX Association, 2006. 335–350.

[65] Hunt P, Konar M, Junqueira FP, Reed B. ZooKeeper: Wait-free coordination for internet-scale systems. In: Proc. of the 2010 USENIX Conf. on USENIX Annual Technical Conf. Boston: USENIX Association, 2010. 11.

[66] Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. In: Proc. of the 2014 USENIX Conf. on USENIX Annual Technical Conf. Philadelphia: USENIX Association, 2014. 305–320.

[67] Ordering service. 2020. https://hyperledger-fabric.readthedocs.io/zh_CN/latest/orderer/ordering_service.html

[68] Larimer D. Dpos consensus algorithm-the missing white paper: Steemit, 2017. https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[69] Wang Q, Yu JS, Peng ZN, *et al*. Security analysis on dBFT protocol of NEO. In: Proc of the 24th Int'l Conf. on Financial Cryptography and Data Security. Kota Kinabalu: Springer, 2020. 20–31. [doi: 10.1007/978-3-030-51280-4_2]

[70] Xu B, Luthra D, Cole Z, Blakely N. 2018. EOS: An architectural, performance, and economic analysis. https://hackernoon.com/eos-an-architectural-performance-and-economic-analysis-43a466064712

[71] Eyal I, Gencer AE, Sirer EG, van Renesse R. Bitcoin-NG: A scalable blockchain protocol. In: Proc. of the 13th USENIX Conf. on Networked Systems Design and Implementation. Santa: USENIX Association, 2016. 45–59.

[72] Chen J, Micali S. Algorand: A secure and efficient distributed ledger. Theoretical Computer Science, 2019, 777: 155–183. [doi: 10.1016/j.tcs.2019.02.001]

[73] Tseng YC, Ni SY, Chen YS, Sheu JP. The broadcast storm problem in a mobile ad hoc network. Wireless Networks, 2002, 8(2–3): 153–167. [doi: 10.1023/A:1013763825347]

[74] Maymounkov P, Mazières D. Kademlia: A peer-to-peer information system based on the XOR metric. In: Proc. of the 1st Int'l Workshop on Peer-to-peer Systems. Cambridge: Springer, 2002. 53–65. [doi: 10.1007/3-540-45748-8_5]

[75] Corallo M. Compact Block Relay. 2016. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki

[76] Analysis of bitcoin transaction size trends. 2015. https://tradeblock.com/blog/analysis-of-bitcoin-transaction-size-trends

[77] Ding DH, Jiang X, Wang JP, Wang H, Zhang XB, Sun Y. Txilm: Lossy block compression with salted short hashing. arXiv: 1906.06500v1, 2019.

[78] Goodrich MT, Mitzenmacher M. Invertible Bloom lookup tables. In: Proc. of the 49th Annual Allerton Conf. on Communication, Control, and Computing (Allerton). Monticello: IEEE, 2011. 792–799. [doi: 10.1109/Allerton.2011.6120248]

[79] Broder A, Mitzenmacher M. Network applications of Bloom filters: A survey. Internet Mathematics, 2004, 1(4): 485–509. [doi: 10.1080/15427951.2004.10129096]

[80] Ozisik AP, Andresen G, Bissias G, Houmansadr A, Levine B. Graphene: A new protocol for block propagation using set reconciliation. In: Proc. of the 2017 Data Privacy Management, Cryptocurrencies and Blockchain Technology. Oslo: Springer, 2017. 420–428. [doi: 10.1007/978-3-319-67816-0_24]

[81] Klarman U, Basu S, Kuzmanovic A, Sirer EG. bloXroute: A scalable trustless blockchain distribution network. IEEE Internet of Things Journal, 2019, 12(11): 1–15.

[82] Labs M. Design and analysis of a decentralized relay network. 2019. https://www.marlin.pro/whitepaper

[83] Hyperledger blockchain performance metrics. White Paper, 2022. https://www.hyperledger.org/learn/publications/blockchain-performance-metrics#

[84] Dinh TTA, Wang J, Chen G, Liu R, Ooi BC, Tan KL. BLOCKBENCH: A framework for analyzing private blockchains. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1085–1100. [doi: 10.1145/3035918.3064033]

[85] Zhang CG, Zhang YF, Li XH, Nie TZ, Yu G. Survey of new blockchain techniques: DAG based blockchain and sharding based blockchain. Computer Science, 2020, 47(10): 282–289 (in Chinese with English abstract). [doi: 10.11896/jsjkx.191000057]

附中文参考文献:

[12] 张彭奕, 宋杰. 区块链共识算法效能优化研究进展. 计算机科学, 2020, 47(12): 296–303. [doi: 10.11896/jsjkx.200700020]

[14] 毛志来, 刘亚楠, 孙惠平, 陈钟. 区块链性能扩展与安全研究. 信息网络安全, 2020, 20(3): 56–64. [doi: 10.3969/j.issn.1671-1122.2020.03.008]

[49] 高政风, 郑继来, 汤舒扬, 龙宇, 刘志强, 刘振, 谷大武. 基于 DAG 的分布式账本共识机制研究. 软件学报, 2020, 31(4): 1124–1142. http://www.jos.org.cn/1000-9825/5982.htm [doi: 10.13328/j.cnki.jos.005982]

[51] 曹源, 张翀, 丁兆云, 姜新文. DAG区块链技术: 原理与实践. 北京: 机械工业出版社, 2018.

[54] 吴仪. 一种基于DAG技术的联盟账本共识机制的设计与实现 [硕士学位论文]. 北京: 北京大学, 2020.

[85] 张长贵, 张岩峰, 李晓华, 聂铁铮, 于戈. 区块链新技术综述: 图型区块链和分区型区块链. 计算机科学, 2020, 47(10): 282–289. [doi: 10.11896/jsjkx.191000057]

石晶(1993－), 女, 博士生, CCF 学生会员, 主要研究领域为区块链系统.

蔡华谦(1990－), 男, 博士, 特聘副研究员, CCF 专业会员, 主要研究领域为分布式系统, 软件中间件.

张奥(1995－), 男, 博士生, 主要研究领域为区块链应用.

刘譞哲(1980－), 男, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为服务计算, 系统软件.

白晓颖(1973－), 女, 博士, 研究员, 博士生导师, 主要研究领域为计算机软件.