软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(3):1040−1064 [doi: 10.13328/j.cnki.jos.007276] [CSTR: 32375.14.jos.007276]

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

区块链分片技术研究进展*

唐海波[1,2], 张 焕[1,2], 张 召[1,2], 金澈清[1,2], 周傲英[1,2]

[1](区块链数据管理教育部工程研究中心 (华东师范大学), 上海 200062)
[2](华东师范大学 数据科学与工程学院, 上海 200062)
通信作者: 张召, E-mail: zhzhang@dase.ecnu.edu.cn

摘 要: 云原生数据库基于云基础设施提供高可用、可弹性伸缩的数据管理, 近年来得到了快速发展. 区块链作为一种透明、防篡改、可追溯的数据库系统, 其中区块链分片是对区块链系统进行扩容的最直接且最有潜力的方案, 利用云基础设施的弹性伸缩特点可以实现更灵活的扩缩容. 首先总结当前区块链分片解决的 3 个关键技术问题: 节点划分的安全性、高效链上数据分片以及跨片交易处理, 分别梳理这 3 个问题的研究现状, 对每个问题下相应的方案进行介绍和对比, 也讨论了将这些方案运用在云原生环境下面临的新挑战. 随后, 围绕这 3 个维度, 从对区块链系统整体影响的角度, 对所有方案进行全面的分析和对比. 最后, 分析区块链分片技术发展趋势, 指出几个值得进一步探索的研究方向.
关键词: 区块链; 分片技术; 共识机制; 可扩展性
中图法分类号: TP311

中文引用格式: 唐海波, 张焕, 张召, 金澈清, 周傲英. 区块链分片技术研究进展. 软件学报, 2025, 36(3): 1040–1064. http://www.jos.org.cn/1000-9825/7276.htm
英文引用格式: Tang HB, Zhang H, Zhang Z, Jin CQ, Zhou AY. Research Progress on Blockchain Sharding Technology. Ruan Jian Xue Bao/Journal of Software, 2025, 36(3): 1040–1064 (in Chinese). http://www.jos.org.cn/1000-9825/7276.htm

Research Progress on Blockchain Sharding Technology

TANG Hai-Bo[1,2], ZHANG Huan[1,2], ZHANG Zhao[1,2], JIN Che-Qing[1,2], ZHOU Ao-Ying[1,2]
[1](Engineering Research Center of Blockchain Data Management (East China Normal University), Ministry of Education, Shanghai 200062, China)
[2](School of Data Science and Engineering, East China Normal University, Shanghai 200062, China)

Abstract: Cloud-native databases leverage cloud infrastructure to provide highly available and elastically scalable data management, and they have experienced rapid development in recent years. As a transparent, tamper-proof, and traceable database system, blockchain sharding is the most direct and promising solution for scaling up blockchain systems. By taking advantage of the elastic scalability of cloud infrastructure, more flexible scaling can be achieved. This study first summarizes three key technical challenges addressed by current blockchain sharding: the security of node partitioning, efficient on-chain data sharding, and cross-shard transaction processing. It reviews the research status of these three issues, introduces and compares the corresponding solutions for each issue, and also discusses the new challenges these solutions face in cloud-native environments. Then, around these three dimensions, a comprehensive analysis and comparison of all solutions are conducted from the perspective of the overall impact on the blockchain system. Finally, the paper analyzes the development trends in blockchain sharding technology and points out several research directions that deserve further exploration.
Key words: blockchain; sharding technology; consensus mechanism; extensibility

* 基金项目: 国家重点研发计划 (2021YFB2700100); 上海市优秀学术/技术带头人计划 (23XD1401100)
本文由“云原生数据库技术与系统”专题特约编辑杜小勇教授、杨晓春教授、祝园园教授推荐.
收稿时间: 2024-05-26; 修改时间: 2024-07-16, 2024-08-19; 采用时间: 2024-08-29; jos 在线出版时间: 2024-09-13
CNKI 网络首发时间: 2024-12-20

近年来随着云服务技术的发展及普及, 基于云服务的数据库系统得到了快速发展, 由于这类系统具备高可用、弹性伸缩、按需计费的特点, 越来越多企业选择将本地数据库服务迁移到云端[1]. 云原生数据库系统通常采取计算存储分离架构, 它们能够对计算或存储单独进行扩缩容, 具备更强的扩缩容能力. 区块链作为一种透明、防篡改、可追溯的新型数据库系统, 近年来被广泛应用于众多领域[2–4], 并且许多应用对区块链系统的存储和交易处理能力都提出了较高的需求, 联盟链场景下这些需求更为迫切[5–8]. 为此, 针对区块链扩容方案相继被提出, 这些方案分为链上扩容和链下扩容两类[9–15], 其中区块链分片被认为是最直接且最有潜力的链上扩容方案. 分片技术来源于分布式数据库领域[16], 它将系统中的所有节点划分成多个工作组 (分片), 在保证系统安全性的前提下, 每个分片负责存储部分数据和处理部分交易[17,18], 以此来降低数据存储冗余度, 同时增加交易处理并行度. 若可以结合云原生数据库相关技术, 就能够实现更加弹性、灵活的分片方案.

分片区块链系统与分布式数据库系统存在许多相似之处, 所不同的是, 区块链系统中存在恶意节点作恶的可能, 例如恶意节点故意沉默或者发不同消息给不同节点, 与其他恶意节点合谋、篡改数据等. 因此, 采用分片对区块链系统扩容需要解决以下几个关键问题.

(1) 节点划分的安全性: 绝大部分分片方案首先将所有节点划分成多个工作组 (即分片), 每个分片负责系统中部分的存储和交易处理任务, 或者只将存储或者交易处理任务进行划分. 每个分片中的节点运行拜占庭共识协议保证一致性, 划分节点后保证系统的安全性至关重要. 由于每个分片中的节点数量远小于系统所有节点数目, 每个分片能够容忍的拜占庭节点数目较小. 一旦恶意节点集中在某个分片, 数量可能超过共识协议所能容忍的上限, 导致恶意节点就能够操控该分片的共识结果, 破坏整个系统的安全性. 因此, 如何保证系统的安全性是区块链分片首要解决的问题.

(2) 高效数据分片与恢复: 早期的数据分片方案不对节点进行分组, 它们只将每份数据划分后随机分配给若干个节点并保证可用性. 数据分片方案需要考虑恶意节点的攻击, 例如删除或篡改数据本地数据, 这会导致部分数据不可恢复. 另一部分数据分片方案首先将节点划分成多个分片, 随后让每个分片负责存储一部分数据的存储以及执行访问了本地数据的交易, 若交易访问了多个分片的数据还会触发跨片交易. 因此, 这部分数据分片策略通常结合负载中的访问模式. 另外, 数据分片还需要考虑节点之间或分片间负载均衡, 当负载发生变化或系统需要进行扩容或缩容时, 需要对数据重新划分, 或者支持高效的动态扩缩容.

(3) 跨片交易的 ACID 与性能优化: 跨片交易本质上是一种特殊的分布式事务, 为了保证事务的 ACID, 对于分布式事务的处理普遍存在开销大、性能低、对系统性能影响大的问题. 在拜占庭网络环境下, 每个分片内节点间的一致性需要拜占庭共识协议来保证, 节点间的消息需要经过签名、验签等流程, 这进一步增加了跨片交易处理开销, 即使少量的跨片交易也会造成系统性能显著下降. 因此, 对于跨片交易的处理, 一方面需要保证事务 ACID 特性, 另一方面是提高跨片交易处理性能, 减少其对系统性能的影响.

区块链分片最早在公链场景下被提出[19], 由于公链采用基于工作量证明[20]共识协议, 导致系统的出块速度慢, 交易吞吐低. 为了提高出块速度, 研究人员开始对节点进行分组, 多个组并行验证交易, 接着, 一部分工作在此基础上对链上数据进行了分片. 然而, 数据层发生的变化会影响系统交易的执行, 例如, 数据分片后可能会带来跨片交易. 因此, 当前的分片工作开始从系统整体的角度去设计方案, 在降低存储开销的同时, 还应考虑对交易执行性能的影响. 另外, 随着云原生等技术的快速发展, 出现了少数基于新架构来优化分片系统性能的工作[21–24], 这些工作发现借助云平台伸缩性强、灵活的特点, 可以实现扩展性更强的分片方案, 但没有进一步分析新场景下当前方案会遇到的具体挑战.

区块链分片技术发展迅猛, 但相关综述仍较少且内容不够全面, 亟需结合最新的技术对其进行全面的综述. 黄华威等人[25]的综述未涉及交易数据分片的相关技术, 也缺乏对节点划分、状态数据分片和跨片交易处理最新进展的介绍. 谭朋柳等人[26]的综述同样未涉及交易数据分片、状态数据分片以及跨片交易处理的相关技术. Yu 等人[27]的综述不仅未涉及交易数据分片和状态数据分片的技术方案, 也缺乏对节点划分和跨片交易处理最新进展的介绍. Berger 等人[28]的综述则未涉及交易数据分片、状态数据分片和跨片交易处理. 除此以外, 以上所有综述工作均未结合云原生数据库技术, 分析将现有分片方案运用在云原生环境下面临的新挑战, 本文涵盖了上述所有方

面, 内容更具完整性. 为了深入介绍区块链分片的进展, 本文首先总结区块链分片的 3 个关键技术问题, 从多个维度对每个问题的相关方案进行分析和对比. 接着, 从对系统整体影响的角度, 对所有方案进行更全面的分析和对比. 最后, 总结了区块链分片中仍需解决的问题.

本文第 1 节根据对区块链系统资源的划分情况, 将分片方案分为网络分片、交易分片和数据分片 3 个层级. 第 2–4 节对引言中总结的 3 个关键技术问题的相关方案进行梳理, 分析将其运用在云原生环境下存在的挑战. 第 5 节对所有方案进行综合分析和对比. 最后, 在第 6 节总结区块链分片研究尚待解决的问题与挑战.

1 区块链分片技术与类型

区块链分片技术通过对系统的网络、存储和计算进行分片来实现扩容. 在区块链分片技术发展过程中, 研究人员最先实现了对网络和计算资源的划分, 增加了交易处理并行度, 提高了系统吞吐. 随后, 一部分工作在此基础上, 进一步对系统的存储进行扩展, 同时提出相应的协议来处理和优化存储分片后系统执行层面临的新挑战, 例如跨片交易. 总体而言, 现有的分片方案对系统资源的划分可以分为网络分片、交易分片、数据分片 3 个层级.

(1) 网络分片, 网络分片负责将系统中的所有节点划分成多个分片, 同时保证每个分片的安全性, 每个分片中的恶意节点数量不超过共识协议能够容忍的上限. 网络分片是许多分片方案的基础, 此时系统中的每个节点仍然存储全量的数据, 并且重复执行所有交易.

(2) 交易分片, 交易分片将系统中的交易分配给不同分片处理, 每个分片只负责验证一部分交易, 最早的分片工作就属于这一类. 交易分片没有对链上数据进行划分, 所有节点存储全部的交易和状态数据, 存储开销大. 由于存储不分片, 系统中不存在跨片交易, 交易处理能力能够随着分片数目的增加线性提升.

(3) 数据分片, 数据分片将链上的数据 (包括交易数据和状态数据) 划分给不同的节点, 每个节点仅维护部分数据. 若交易需要访问的数据位于其他分片, 需要从其他分片获得所需数据. 对于同时也进行了网络和交易分片的系统, 每个分片只处理访问了本地数据的交易, 若分片只有交易访问的部分数据则会触发跨片交易. 同时对网络、交易和数据都进行分片的方案也被称为全分片.

图 1 是网络分片、交易分片以及数据分片 (全分片) 的示意图, 网络分片将系统中的节点划分成 3 个分片, 每个分片中的节点依旧存储全量数据, 重复执行所有的交易; 在交易分片方案中, 每个分片的节点仅验证和执行一部分的交易, 仍存储全量数据; 在数据分片 (全分片) 方案中, 每个分片中的节点不仅只验证和执行一部分交易, 同时仅存储一部分的数据.

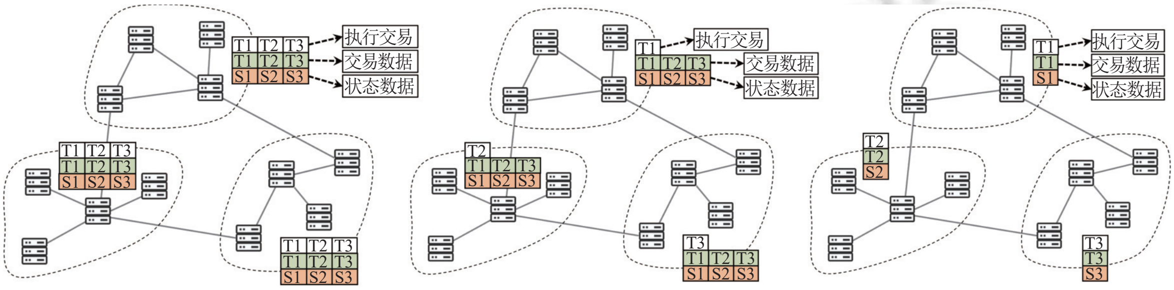

图 1 网络分片、交易分片与数据分片 (全分片) 示意图

除了根据被划分的资源, 将所有方案分为以上 3 类. 从分片系统架构的角度分析, 现有分片方案也可以分为扁平化分片架构和层级化分片架构两类. 这两类方案的区别在于所有分片之间的关系是否对等, 层级化的分片架构能够弥补传统扁平化架构的缺陷, 赋予分片系统更高的可扩展性.

(1) 扁平化分片架构, 在传统扁平化分片系统中, 所有分片之间的关系对等, 分片间的协作开销大. 分片之间的协作通常基于一个随机的协调者来完成, 协调者与跨片参与方之间的网络延时可能较高, 并且跨片协调者的随机

性不利于形成对跨片交易的批处理, 这些因素共同导致了扁平化架构中跨片交易性能低. 随着分片数量的增加带来的跨片交易数量增加, 进一步加剧了扁平化的缺陷对可扩展性的影响.

(2) 层级化分片架构, 在层级化分片架构中, 分片之间的关系不再对等, 分片按照树型组织. 除了根分片, 所有分片在上一层都存在父亲分片, 这些分片承担着协调下层特定分片之间的跨片交易, 跨片交易的协调者固定. 与此同时, 层级化分片架构相比扁平化架构多了一个扩展维度, 若系统中的跨片交易数量较多, 系统可以通过增加上层分片层数, 对系统跨片交易处理能力进行扩展.

图 2 展示了传统扁平化分片区块链架构, 以及基于端边云的层级化分片区块链架构. 扁平化架构中的分片之间关系对等, 对于跨片交易, 每次从参与者以外的分片随机选择一个第三方分片作为跨片交易协调者, 绝大多数的分片方案都属于此类. 在基于端边云场景的层级化分片架构中, 底层分片由靠近用户的边缘设备运行, 中间层由靠近边缘设备的网络设备运行, 顶层则在云端服务器上运行. 底层分片之间的跨片交易可以由指定的距离最近的父亲分片协调, 实现跨片交易批处理并且具有较低延时. 若参与跨片交易的分片在中间层没有公共祖先, 则由云端负责协调. 区块链系统可以借助云端可伸缩特性, 继续增加上层分片的数量或层数来对系统进行纵向扩展.

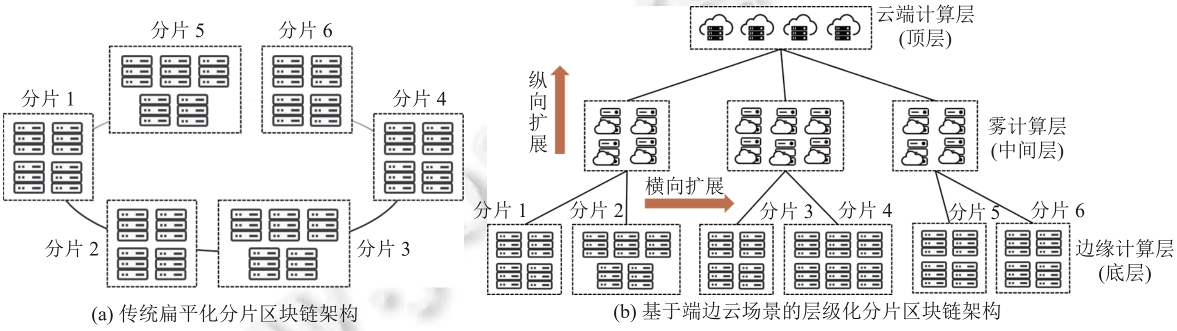

图 2　传统扁平化分片架构与层级化分片区块链架构

2 节点划分的安全性

绝大部分的分片工作首先将节点划分为多个组, 不同组的节点随后负责存储不同的数据或者处理不同的交易, 保证节点划分后系统的安全性至关重要. 本节将这类工作分为两类进行综述: 基于概率的安全性保障方案以及基于片间协作的安全性保障方案, 前者主要思想是通过保证节点划分过程尽可能随机, 让划分后每个分片中的恶意节点占比与系统划分前接近, 从概率上保证恶意节点数量不可能共识协议保证安全的上限. 后者通过分片之间相互监督, 及时发现分片的作恶行为, 对每个分片安全性的要求没有前者严格. 在本小节最后, 对这些工作进行了总结, 并对相应技术的应用情况进行介绍.

2.1 基于概率保障安全性

由于系统中存在拜占庭节点, 分片前系统只有一个分片, 该分片中的节点数量较多, 因此能够容忍的拜占庭节点数量也较多. 而节点被划分到不同分片以后, 每个分片中的节点数量都小于未分片时候, 此时每个分片能够容忍的拜占庭节点数量也随之降低, 一旦较多的恶意节点被划分到某个分片中就会影响系统整体的安全性. 以片内运行 PBFT[29]共识算法为例, 假设系统中共有 N 个节点, 被分片成了 m 个分片, 在分片之前整个系统最多能够容忍的恶意节点数量 $f = N/3-1$, 划分后的每个分片中的节点数目为 N/m, 每个分片能够容忍的恶意节点数量为 $f = N/(m\times3)-1$, 分片后系统更容易被攻击.

Elastico[19]是首个基于概率来保证分片后系统安全性的工作, Elastico 对节点的划分主要包括身份建立、委员会分配和随机数生成这 3 个阶段, 每经过一段固定时间系统重新划分节点. 在身份建立阶段, 所有节点经历一个挖矿过程生成自己的身份, 并让最先解决难题的 c 个节点组成一个根委员会. 在委员会分配阶段, 根委员会中的节点

根据其他节点身份的后 s 位确定它们被划分到的分片. Elastico 让每个组委会中节点数量达到一定规模, 减少委员会中的恶意节点数目占大多数的概率. Elastico 能够保证每个组委会内的恶意节点数目不超过其总节点数的 1/3, 系统能够容忍的总恶意节点数目不超过 $N/4$. 在随机数生成阶段, 根分片生成新的随机数广播给所有节点, 该随机数用于新一轮的节点划分.

Elastico 划分出的分片规模较小 (每个分片约 100 个节点), 能够形成较多的分片, 带来较高的交易处理并发度. 然而, 当系统中总恶意节点数目达到 1/4 时, 系统安全性迅速下降, 每个分片每出一个区块的失败率达到了 2.76%, 分片容易被恶意节点控制. OmniLedger[30]通过 VRF[31]选举和无偏随机数提高划分过程中的随机性来提高系统安全性, 首先在节点身份生成阶段, 节点需要将其创建的身份广播到链上, 只有被大多数节点验证通过的节点身份才被认为有效. 随后, 节点基于 VRF 选举出一个主节点, 主节点负责生成一个无偏随机数广播给其他节点, 所有节点根据随机数得到当前应该加入的分片. OmniLedger 结合密码学并且增加分片中的节点数量来提高分片安全性, 系统整体能够容忍的恶意节点比例与 Elastico 相同.

Elastico 和 OmniLedger 都存在整体容忍的恶意节点数量较少的问题, 另外由于 OmniLedger 中的每个分片规模较大, 导致能够划分出的分片数目较少, 同时每个分片共识开销增大. 最后, Elastico 和 OmniLedger 每次经过一段时间将所有节点重新划分, 来减少较多的恶意节点被划分到同一个分片的概率. 然而, 节点在片间迁移需要拉取其他分片的历史交易和状态数据, 节点需要在本地重新构建可验证索引, 给节点本身带来巨大的 I/O 开销, 同时大量节点的重分片会给整个系统带来巨大的网络开销.

针对 Elastico 和 OmniLedger 存在的这些问题, RapidChain[32]做到分片内节点数比 OmniLedger 少, 并且系统整体能够容忍更多的恶意节点, 从而支持更多的分片数目. 在系统刚开始阶段, RapidChain 首先基于一个随机种子创建关于所有节点的二部图, 逐层均匀地选择出部分节点形成一个参考委员会. 参考委员会基于一个生成的无偏随机数对剩余节点进行划分, 并将划分结果通知给节点. RapidChain 基于布谷鸟规则 (cuckoo rule)[33]的提出轻量级划分方案, 每次重划分只需迁移分片中的一小部分节点就能保证分片的安全, 划分后每个分片仅包含 $\log N$ 个节点, 分片规模较小并且相对均衡. RapidChain 的每个分片中使用 Ren 等人[34]提出的共识协议, 片内能够容忍不超过 1/2 的恶意节点, 正因为单个分片容错率较高, 所以系统整体能够容忍的恶意节点数量能够比 Elastico 和 OmniLedger 多.

Elastico、OmniLedger 和 RapidChain 均属于静态节点重划分, 假设系统的分片数目固定不变, 但在真实应用中, 随着负载变化系统可能对系统进行扩容或缩容. 然而, 分片数量的变化会影响系统的安全性, 如何在这两者之间取得平衡对于区块链系统是非常重要的. SkyChain 将动态节点重划分建模为马尔科夫决策过程, 重划分策略考虑的变量包括重划分频率、分片数量以及区块大小, 并对分片后系统的安全性进行评估, 利用深度强化学习获得在不同负载下高性能且安全的节点划分方案. 在云原生环境中, 这类方案面临着新的安全性挑战. 由于云原生环境下每个逻辑节点被解耦为共识和存储服务, 并且这些服务分别运行在多个计算节点和存储节点上, 为了避免系统中绝大多数逻辑节点被恶意节点控制, 逻辑节点的阶段或者存储服务应该尽可能均匀划分在不同机器. 然而, 当对逻辑节点的计算或存储进行扩容时, 新引入的计算或存储节点可能是恶意的, 导致逻辑节点被恶意节点控制, 如果较多的节点都被控制将影响整个逻辑节点的安全性, 系统扩容前的划分方案无法继续保证分片安全性.

SharPer[35]让一定地理范围内的大量节点负责运行同一个分片, 它基于一个确定性的概率来保证每个分片的安全: 每个分片中的总节点数一定远大于恶意节点数, 因此恶意节点数目不会超过共识协议能够保证分片安全的上限. 这种方式由于每个分片中节点数量多, 最终能够支持的分片数目较少, 单个分片的负载较高. 在云原生环境中, 使用大规模的节点来运行单个分片会带来巨大经济成本. AHL[36]借助可信执行环境 (TEE)[37]来避免恶意节点对系统的影响, 从而系统能够支持更多的分片. 节点拥有 TEE 的加持也能够应对云原生环境下扩缩容时的安全问题, 节点可以依靠 TEE 提供的隔离执行环境屏蔽恶意节点的影响.

在基于概率的安全性保障方案中, 每个分片只会独立验证自己处理的交易, 任意一个分片被恶意节点控制都会导致整个系统不安全, 因此要求每个分片的出故障的概率保持在极低的水平, 这会导致每个分片内的节点数量较多、划分出的分片少, 节点重划分较频繁等问题.

2.2 基于片间协作保障安全性

在基于片间协作的安全保障方案中, 每个分片的交易验证结果会交由其他分片再次验证, 避免恶意节点控制某个分片后对系统造成破坏. 由于能够通过片间的相互监督阻止分片作恶, 这类方案对分片故障的容忍度更高. 因此每个分片中的节点数量可以相对较少, 从而产生更多的分片, 并且无需对节点定期重划分. 然而, 片间相互验证交易会带来额外的网络开销, 对交易处理性能造成影响.

对采用 PoW 共识的区块链系统进行分片后, 每个分片的算力只有分片前的 $1/k$, 其中 k 为划分出的分片数目, 恶意节点攻击系统的难度大大降低. 为了解决这个问题, Monoxide[38]提出连弩挖矿 (Chu-ko-nu mining) 来对矿工的算力放大, 每个矿工同时加入多个连续分片的交易共识, Merkle 同时包括了多个分片的区块, 矿工的哈希计算的输入包括该 Merkle 树的树根, 矿工一次性可以对来自多个分片的区块进行共识. 然而, 在云原生的环境下, Monoxide 的安全性和性能均遇到新的挑战. 例如如果对 Monoxide 的节点进行扩容, 原本诚实的节点有可能被恶意的节点掌控, 如果恶意节点的算力超过全部算力的 50%, 系统将被恶意节点控制. 在性能方面, 连弩挖矿要求每个矿工同步其他分片的区块数据, 但不同分片节点可能分布在不同地理区域, 这导致区块同步效率低, 从而影响整体交易的处理延时.

SSChain[39]设计了一个两层的分片架构来保证系统安全性, 在对系统划分出 m 个分片后, SSChain 采取激励机制组织一部分的节点形成一个根链, 根链拥有超全网 50% 的算力, 下层分片验证过的交易最终由根链批量确认, 确保系统的安全性, 但这会增加交易延时, Monoxide 不存在这个问题. 另外, 如果激励机制没有让根链分片中算力超过全网 50%, 那么系统被恶意节点攻击的风险就会增加. 在云原生环境下, 节点资源动态变化, 更容易导致根链的算力未超过全部的 50% 情况发生. 同时, 由于云原生环境下的资源竞争和成本问题, 现有策略可能由于激励不足导致根链算力低于全网 50%, 因此激励机制需要保证根链算力的绝对优势.

Monoxide 和 SSChain 能够保证采用 PoW 共识协议的公链分片系统安全性, 联盟链通常采用基于投票的共识协议, 它们不适用于联盟链. CoChain[40]基于片间协作来提高采用 PBFT 共识协议的分片系统安全, 能够容忍每个分片拥有不超过 2/3 的恶意节点, 因此每个分片中的节点数目可以较小. 这是因为 CoChain 发现只要恶意节点不超过片内节点总数的 2/3, 节点就无法收齐足够对非法交易的投票 ($2f+1$), 非法交易不会被验证通过. 恶意节点只能够进行某些特殊类型的攻击, 例如让整个分片沉默, 生成相同区块高度的交易来使得片内账本分叉. 为了应对这个问题, CoChain 中的每个分片验证完交易时, 会将共识结果发送至其他 m 个监管分片, 同时保证这 m 个分片中超过 2/3 是诚实的, 监管之间分片对分片的共识结果再次进行共识, 并设置超时机制来检查分片的沉默攻击, 这种方式会带来额外的网络开销, 导致交易的确认延时较高. 在云原生环境下, 如果不同分片的节点跨域不同地理区域, 将会导致交易延时进一步提高. 同时, CoChain 仍无法解决云原生平台中节点扩缩容后系统安全性无法保障的问题, 因为逻辑节点扩容后, 云平台中的恶意计算节点也有可能会影响监管分片的安全性, 进而破坏系统的安全. 同时, 不同分片节点若位于不同的数据中心, 这也将增加片间相互监督对系统性能的影响.

Benzene[41]也通过分片相互监督来提高系统的安全性, 但每个监督分片对收到的来自源分片的交易独自验证, 源分片的诚实节点根据收到的来自其他分片投票决定是否提交, 这种方式下的交易网络延时比 CoChain 低. 如果恶意节点同时生成区块高度相同的区块来产生分叉, 诚实节点只将投票最多的区块上链. 另外, Benzene 在节点中引入 TEE 来降低分片内矿工验证来自其他分片交易的开销. Benzene 与 AHL 类似, 它们都借助 TEE 提供的隔离执行环境执行操作, 防止恶意节点对系统的影响, 因此它们即使被运用在了云原生环境下, 也能够持续保证系统的安全性.

2.3 保障系统安全性方案总结

根据本节介绍的典型节点划分方案, 从方案类别、分片内的节点规模、片内容忍恶意节点数量、系统整体容忍的恶意节点数量、片内交易确认延时, 是否需要对节点重划分、采取的片内共识协议 7 个方面进行了对比, 并整理得到表 1.

1) 分片内节点规模方面: Elastico 每个分片的节点规模小但系统的安全性低, OmniLedger 增加了分片中的节

点个数提高安全性, RapidChain 在保证安全性基础上, 分片中的节点数目比 OmniLedger 少. SharPer 将一定地理范围内的大量节点划分在一个分片来保证安全性, 分片中的节点规模较大. AHL 借助可信硬件来避免恶意节点作恶, 能够减少分片中的节点数目. 基于协作的安全保障方案由于分片间互相监督, 分片如果作恶够被及时检测出, 因此也可以减少分片中的节点数量.

表 1　典型节点划分方案对比

类别	方案	分片内的节点规模	片内容忍恶意节点数量	整体容忍恶意节点数量	片内交易确认延时	是否节点需重划分	片内共识协议
基于概率的安全保障方案	Elastico	小	1/3	1/4	低	是	PBFT
	OmniLedger	大	1/3	1/4	低	是	PBFT
	RapidChain	中	1/2	1/3	低	是	文献[34]
	SharPer	大	1/3	—	低	否	PBFT
	AHL	小	1/3	—	低	是	PBFT
基于协作的安全保障方案	Monoxide	小	1/2	1/2	中	否	PoW
	SSChain	小	1/2	1/4	中	否	PoW
	CoChain	小	2/3	1/3	高	否	PBFT
	Benzene	小	1/3	1/3	中	否	PBFT

2) 在片内能够容忍恶意节点数量方面: Elastico、OmniLedger 和 Benzene 的每个分片能够容忍不超过片内节点总数 1/3 的恶意节点, RapidChain 通过在片内运行 Ren 等人[34]提出的共识协议, 让分片可以容忍不超过片内总节点数 1/2 数量的恶意节点. Monoxide 和 SSChain 的分片采用 PoW 协议运行, 因此最多可以容忍不超过片内总节点数 1/2 的恶意节点. CoChain 只需抵御被攻击分片的故意沉默、分叉攻击, 因此每个分片最多允许的恶意节点数量不超过片内节点数的 2/3.

3) 整体容忍恶意节点数量方面: Elastico、OmniLedger 和 SSChain 整体能够容忍的恶意节点数量最少, 不超过总节点数的 1/4, RapidChain、CoChain 和 Benzene 能够容忍恶意节点不超过 1/3, Monoxide 能够容忍的恶意节点数量最多, 这是因为它避免了算力稀释对安全性造成的影响.

4) 片内交易确认延时方面: 基于概率的安全保障方案 (Elastico、OmniLedger、RapidChain、SharPer 和 AHL) 的单笔片内交易确认延时最低, 这是因为它们的交易只需被一个分片的共识模块验证. Monoxide 的矿工每次需要同步其他分片的区块, 增加了交易延时. SSChain 和 Benzene 的延时也较高, 因为片内交易确认需要经过其他分片的验证, CoChain 的交易延时最高, 因为验证分片之间需要运行 PBFT 协议对分片的共识结果共同达成共识.

5) 节点重划分方面: 基于概率的安全保障方案 (Elastico、OmniLedger、RapidChain、AHL) 需要定期对节点进行重划分. SharPer 的每个分片的安全性是确定的, 无需定期对节点进行重划分. 基于协作的保障方案 (Monoxide、SSChain、CoChain 和 Benzene) 不需要对节点进行重划分, 即使系统中出现不安全分片, 也能够基于片间协作发现恶意分片的作恶行为.

6) 支持的分片内共识协议: Elastico、OmniLedger、SharPer、AHL、CoChain 和 Benzene 分片内使用 PBFT 共识协议, Elastico、OmniLedger 中的节点身份由节点经历一个类似挖矿的过程获得. RapidChain 采用 Ren 协议来支持容忍更多的恶意节点, Monoxide 和 SSChain 中的分片使用 PoW 共识.

基于概率和基于协作的保障方案目的都在于避免恶意节点对系统造成破坏. 前者主要设计不同的节点划分策略, 从概率上保证划分出来的每个分片尽可能安全, 后者从监督分片行为角度出发, 每次对分片的共识结果进行多方验证. 基于概率的保障方案的主要挑战在于节点重划分算法的设计, 基于协作的保障方案的主要挑战在于精准发现恶意分片, 同时由于每一笔片内交易都需要经过其他分片的验证, 平均一笔交易的验证开销比传统方式下的要高. 如果能够将前者的重划分频率降低到一个较低的水平, 每笔交易依然只需要涉及到的分片进行验证, 可以减少为了保证系统安全对性能造成的牺牲, 这一部分也有相关的优化工作.

在相关技术的应用方面, 目前绝大多数开源分片区块链项目都采用基于概率保障的方案来保证系统的安全性, 例如以太坊 2.0[42]、Harmony[43]、NEAR[44]和 Zilliqa, 它们都通过随机分配节点, 并且周期性地重分配节点来防

止恶意节点操控某个分片. 具体而言, 以太坊 2.0 信标链使用去中心化的随机数生成机制 (RANDAO) 来指定每个分片中的验证者, 同时基于使用 VDF (可验证延迟函数)[45]来延迟揭示随机数, 以防止攻击者干预随机数的生成. NEAR 基于 VRF 来指派负责划分节点的分片. Harmony 同时结合了 VRF 和 VDF 来保证节点分配的随机性和安全性. Zilliqa 中的节点首先运行 PoW 获得身份, 其中部分高信誉节点构成一个委员会, DS 委员会再基于 VRF 对剩余的节点进行划分.

3 高效数据分片与恢复

数据分片主要研究降低节点的存储开销并保证数据的可用性, 同时还需关注数据分片可能对交易处理性能的影响. 本节首先围绕只对数据进行分片的系统 (没有全分片), 介绍 3 种数据分片技术: 基于群组的数据分片、基于门限秘密共享的数据分片和基于纠删码的数据分片. 然后, 对于采取了全分片的系统, 介绍了基于负载的分片技术. 最后, 对以上介绍的所有方案进行总结, 并对相应技术的应用情况进行介绍.

3.1 基于群组的数据分片技术

基于群组的数据分片方案首先将所有节点划分为多个群组, 每个群组维护一份完整的链上数据, 数据在每个群组内的不同节点之间分片存储. Abe 等人[46]基于分布式哈希表 (distributed Hash table, DHT)[47]将链上交易数据均匀地划分到不同节点, 根据哈希表能够快速定位数据所在的节点, 其中分布式哈希表基于 Chord 算法[48]实现. 为了提高数据的可用性, 同一份数据也被冗余地存储在群组内的多个节点上. 这种方案简单直接地降低了存储开销, 但由于划分策略没有考虑负载中的数据访问模式, 客户端查询历史交易数据时可能需要从多个节点获取数据, 这会造成分片后的数据查询性能低.

为了能够在数据分片后交易数据的存储开销和查询性能之间取得一个平衡, CUB[49]将该问题定义为多目标交易数据分配问题 (blocks assignment optimization, BAO), 并证明其为一个 NP-hard 问题. 随后提出相应的启发式算法指导交易数据分配. 主要思想是根据一段时间内数据的访问模式, 增加系统中频繁地被多个节点访问数据的冗余度. 若新区块添加进来时系统没有额外的存储空间, CUB 减少当前查询热度最低数据的存储冗余度来, 以此来换取多余的存储空间.

Abe 等人的方案和 CUB 均支持每个群组内节点的动态加入和离开, 在 Abe 等人的工作中, 节点间通过维护一致的哈希环来确定每份交易数据存储位置, 当节点成员发生变动, 节点检查哈希环上此时相邻的节点的变化, 然后向哈希环上的前序或者后续节点请求数据. CUB 通过设计启发式算法来决定每次移动到新节点的数据, 当有节点离开时, 检查要离开节点中存储的数据在系统中是否存在副本, 否则安排系统中剩下的某个节点存储这部分数据.

以上两种方案均假设群组内的节点相互信任, 缺乏对拜占庭节点作恶情况的处理, 并且只对交易数据进行了分片, 每个节点仍然保留完整的状态数据. 在云原生的环境中, 这样的信任假设尤为不合理, 云原生下资源的动态性增加了逻辑节点被恶意的存储节点影响的风险. Jidar[50]根据节点的兴趣偏好对数据进行分组, 为了防止拜占庭节点作恶, 节点基于 Merkle 证明来验证数据的正确性. 当客户端广播交易时, 将交易以及交易输入的 Merkle 证明发送给其他节点, 节点在本地验证通过后存储数据. 当节点查询交易时, 根据其他节点返回存储的数据片段以及证明来恢复完整的数据. 由于节点只存储其感兴趣的数据, 由于对冷门数据感兴趣的节点数量可能很少, 这会导致这部分数据分片后的可用性变差. 另外, 这两个方案能够减少的存储开销有限, 假设系统中有 f 个恶意节点, 为了最终能够恢复出数据, 每份数据至少在 $2f+1$ 个不同节点中进行存储. 在云原生环境下, 由于不同逻辑节点的存储服务可能运行在相同的存储节点上, 导致恶意存储节点能够通过同时影响多个逻辑节点, 从而破坏数据的可用性. 同时, 对节点的存储扩容也可能导致原本正常的逻辑节点被恶意节点操控. 在性能方面, 由于云原生采取存算分离架构, 计算层需要向单独的存储服务请求数据, 导致交易延时增加. 另外, 节点可能分布在不同地理区域云服务中心, 这些云存储之间数据相互隔离、网络延时高, 这会造成数据恢复效率低.

以上所有工作只对交易数据进行了分片, LDV[51]根据车载社交网络中用户节点的兴趣爱好将节点划分成不同兴趣组, 每个组存储一部分相同的交易和状态数据, 同一节点可能同时参与多个群组. 然而仍可能存在一些冷门的

数据, 很少或没有节点对其进行存储, 导致这部分数据可用性低, 类似于 Jidar. 为此 LDV 设置了少量的全节点来存储所有数据. 考虑到车载网络数据具有时效性, LDV 会定期删除节点中过时的历史数据, 以减少存储开销, 而全节点中的历史数据不会被删除.

3.2 基于门限秘密共享的数据分片技术

基于门限秘密共享的数据分片方案相较于基于群组的数据分片方案, 具有更低的存储开销和更高的数据可用性. 张国潮等人[52]利用改进的 Shamir 门限算法为单个区块数据生成 n 个数据块, 然后将这 n 个数据块分别发送给不同的节点进行存储. 当系统需要恢复原始数据时, 只需获取任意 k 个数据块即可完成, 无需获取全部 n 份数据块, 每个节点的存储开销仅为分片前的 $1/k$, 门限参数 n 和 k 可由用户自行定义. 假设系统中总节点数为 N, 恶意节点数量为 f, 所有数据经过节点签名, 恶意节点无法篡改数据, 若诚实节点数 $N–f$ 大于等于 k, 系统能够从节点中成功恢复出原始数据. 然而, 该类方案在云原生环境下同样面临数据可用性难以保证问题, 资源的动态变化易导致逻辑节点的被恶意节点影响, 即使数据经过签名, 恶意节点无法篡改数据, 但可以故意沉默, 导致无法收齐 k 个数据块恢复数据. 另外, 节点如果跨不同地域也会导致数据恢复效率低.

Raman 等人[53]通过将链上的冷数据归档到少量备份节点, 减少每个节点的存储开销. 为了防止因为备份节点遭受恶意节点攻击, 导致数据可用性降低, Raman 等人提出了一种数据动态分配策略, 定期将数据片段在不同节点上进行迁移, 减少因恶意节点攻击导致部分数据无法恢复的风险. 此外, 该方案还对原始数据进行加密, 并将加密数据与私钥一起分片存储, 不仅提高了数据的保密性, 私钥分片存储避免了由于节点受到攻击而导致私钥丢失的风险. 以上两种方案也只对交易数据进行了分片.

3.3 基于编码的数据分片技术

另一类工作基于纠删码 (erasure code, EC)[54]来对链上数据分片存储, 同时设置一定数量的冗余编码块应对恶意节点的攻击, 数据的存储开销低于基于群组的分片方案. BFT-store[55]基于纠删码对交易数据进行分片, 假设系统中共有 N 个节点, 其中有 f 个恶意节点. BFT-store 以连续 $N–2f$ 个区块的总数据为单位进行编码, 得到 $N–2f$ 个数据块, $2f$ 个冗余块, 数据块和冗余块的大小相同, 分别被存储在系统的不同节点上. 每个数据块对应一个原始的区块数据, 如图 3 所示. 这种划分方式对于保证交易数据查询性能有利的, 因为节点拥有交易所在区块的完整数据, 节点可以立即生成交易证明, 无需向其他节点请求数据. BFT-Store 存在数据恢复开销大的问题, 即使节点只需要一小部分的数据, BFT-store 一次性会恢复出连续 $N–2f$ 个区块的数据. 这种基于编码的数据分片方式与基于门限秘密共享的方案类似, 都是将原始数据转换为多个带签名的数据段, 系统只需从部分诚实节点获取这些数据段即可恢复出原始数据. 它们在云原生环境下都面临着类似的挑战, 包括节点扩容后系统的安全性难以保证, 如果节点跨越不同地域的数据中心将会导致数据恢复延时高.

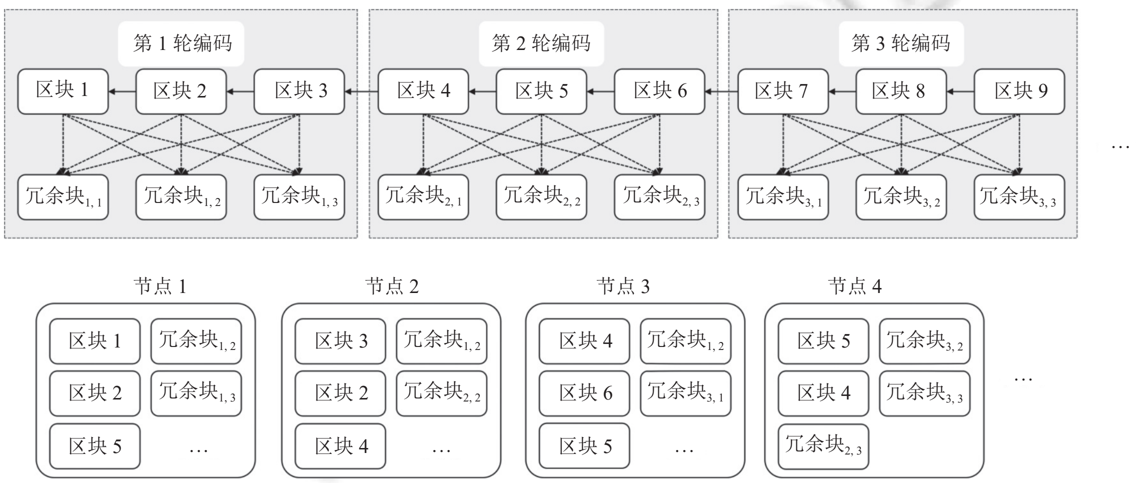

图 3 基于纠删码的交易数据分片示意图

在存储开销方面, 除去需要少量的存储空间用来存储冗余块, 每个区块的数据只在单个节点上存储一次, 因此 BFT-store 将链上区块数据存储开销从 O(N) 降低到了 O(1). 节点只需从任意 N–2f 个节点处拿到编码块 (数据块/冗余块) 就能恢复出原始数据. BFT-store 也支持节点的动态加入和删除, 当节点成员发生变动时, 由一个主节点请求足够多的编码块恢复出原始数据, 然后将原始数据发送给旧的节点和新节点, 节点对收到的原始数据重新编码, 保留本地需要存储的数据块或者冗余块, 重编码期间系统能够继续对外提供服务. BFT-store 重编码网络开销较大, 主节点在恢复原始数据块后需要将原始数据块重新发送回所有节点. 同时, BFT-store 依赖协调重编码过程的主节点诚实可信.

PartitionChain[56]针对 BFT-store 重编码开销大的问题进行了优化, 其主要思想是当新节点加入网络后, 只有新节点需要存储新的冗余块, 旧节点中的编码块只需更新聚合签名. 具体而言, 新加入的节点首先从一定数目的旧节点中恢复出原始数据, 然后对数据重新编码, 保证旧节点中的编码块无需改变, 只要在本地存储一些新的冗余块. 随后, 新节点根据恢复出的原始数据, 使用自己的私钥对每个旧节点中的编码块进行签名, 将签名结果发送至旧节点. 旧节点将新节点的签名聚合到本地数据块的签名中, 证明新节点已经对这部分数据进行了确认. 相应地, 旧节点也会对新冗余块的签名发送给新节点. PartitionChain 在重编码过程不依赖一个主节点的协调, 旧节点也无需对完整数据进行重编码.

对编码块进行冗余存储不仅能够增加数据的可用性, 同时也会增加节点的数据查询性能[57]. Li 等人[58]分析认为, 影响基于纠删码的存储分片系统数据查询性能的原因主要有两个: 数据块的存储冗余度以及数据的存放位置, 提出了一个针对存储开销和数据查询延时的多目标优化模型, 指导编码块的冗余度设置以及数据存放位置的选择, 平衡数据分片后的节点存储开销和数据查询效率.

Zhang 等人[59]提出了一种将多群组与纠删码结合的数据分片框架, 该框架首先将所有节点划分成不同的群组, 每个群组只存储一部分链上数据, 在每个群组内部, 节点之间的数据利用纠删码分片存储. 与 BFT-store、PartitionChain 等方案对比, 该框架首先以群组为粒度对数据进行分片, 进一步降低了存储开销. 此外, 该框架在群组间对编码块冗余存储来提高数据可靠性. 考虑到冗余块的数量及其存放位置影响着系统的存储开销、群组间的存储均衡度以及数据可用性. Zhang 等人提出了一种基于 NSGA-II[60,61]的编码块分配方案. 该方案能够让划分结果在上述几个目标之间达到平衡.

3.4 基于负载的数据分片技术

对于采取了全分片的系统, 数据的分片策略对交易执行性能的影响非常大. 例如, 不合理的状态划分策略可能导致片间负载不均衡[62−64], 部分分片的资源利用率低. 如果交易需要访问的状态涉及多个分片, 还会触发跨片交易, 即使少量的跨片交易, 也会显著降低系统性能. 尽管研究人员已提出多种方案来优化跨片交易性能, 其处理开销仍然远高于片内交易. 因此, 一些全分片工作基于负载中的访问模式, 通过优化状态放置策略来减少跨片交易数量, 同时保持分片间负载均衡.

为了均衡每个分片的负载, Chainspace[65]根据账户地址的哈希值将所有账户随机分配到不同的分片中, 计算方式为 SHA256 (账户地址) mod k, 其中 k 是分片的数量, 交易则被分配到存储相应输入账户的分片中处理. Monoxide 根据账户地址的前 k 位将账户随机分配到 2^k 个分片中. OmniLedger 和 RapidChain 都是基于 UTXO 账户模型的区块链系统, 类似于 Monoxide, 它们也根据 UTXO 的地址前缀来将 UTXO 分配到不同的分片. 早期的这些分片方案由于忽略了负载中的访问模式, 这种随机划分账户的方式会引发大量跨片交易, 严重影响系统的性能. 因此, 随后的一些工作尝试在保证系统负载均衡的同时较少的跨片交易.

OptChain[66]提出了一种轻量级的交易放置策略, 它将历史负载建模成一种交易网络图, 其中每个点表示一笔交易, 边表示交易之间的依赖关系. OptChain 通过分析历史交易网络图中的访问模式, 在接收到新交易时, 对每种可能的划分方案进行评分. 评分的因素包括减少的跨片交易数量和分片间的负载均衡度. 然而, OptChain 仅适用于采用 UTXO 账户模型的区块链系统.

Shard Scheduler[67]提供了一种轻量级、支持所有账户模型的状态划分方案. 其主要思想是, 对于一笔跨片交

易, 若交易访问的本地状态与其他所有分片历史交易数目大于其参与的片内交易数目, 说明状态和当前分片中状态的访问局部性较低, 在这种情况下, Shard Scheduler 将状态迁移至当前负载最低的其他分片中. 这种策略主要考虑了片间的负载均衡, 会带来较多的跨片交易.

BrokerChain[68]首先将历史负载建模为一张“账户-交易”图. 与 OptChain 不同的是, 该图中的节点代表账户, 节点权重表示账户被访问的次数, 边权重表示账户之间历史交易次数. BrokerChain 基于 METIS[69]对图进行划分, 位于相同子图中的状态将被划分到一个分片. METIS 能够尽可能减少被割边的权重之和, 同时保持子图之间的节点权重和均衡, 以达到减少跨片交易同时保持片间负载均衡. 然而, 分片的负载实际上来自于交易 (即图中的边), 并且片内交与跨片交易的负载不相同, 这导致了 BrokerChain 划分结果负载不均衡.

TxAllo[70]与 BrokerChain 采用相同的方法对历史负载进行建模, 不同之处在于, TxAllo 避免了 BrokerChain 上述问题. 在对建模出的图进行划分时, TxAllo 首先利用社区发现算法[71,72]将图划分成不固定数目的初始子图, 每个子图包含一批互不相交的状态. 由于社区发现算法无法提前确定初始子图的数量, TxAllo 设计了贪心策略对初始划分结果进行调整, 贪心策略的优化目标是划分后系统的整体吞吐量, 影响吞吐的因素包括跨片交易数量和负载均衡度.

在此之前的工作都普遍认为跨片交易是影响分片系统性能最主要的因素, 而研究人员发现这个结论基于系统中的每个分片此时负载较高, 在系统整体负载较低的时候, 负载不均衡对性能的影响最为严重, 低负载分片的计算资源空闲, 系统资源利用率低, 而高负载分片会造成交易等待处理, 这些问题相比跨片交易对性能的影响更大. 基于这个观察, LB-Chain[73]基于两层 LSTM 模型来预测账户可能发生的交易数量, 以此来指导负载较低时候的状态划分, 让分片间的负载尽可能均衡.

数据划分不仅需要考虑可能引发跨片交易、负载不均衡问题, 当系统增加/减少分片数量时, 需要对数据在所有分片间进行重新划分, 这会导致分片间进行大量数据迁移. 同时, 由于区块链通常基于 LSM-tree 的存储引擎存储状态数据索引, LSM-tree 的读放大和写放大问题会给节点重建状态索引带来巨大 I/O 开销, 数据分片要能够以更轻量的方式处理该任务[74,75]. S-Store[76]采用一致性哈希进行状态划分, 减少分片扩缩容时片间迁移的数据量. 此外, S-Store 还提出了一种新型聚合 Merkle 树 AMB 树, 以减少可验证索引的重建开销. 然而, 采用一致性哈希划分状态数据忽略了负载中的访问模式, 可能会导致大量跨片交易.

在云原生环境下, 如何合理划分状态数据, 减少跨片交易需要面临更多新的挑战. 由于系统可能部署在跨地域的多个云服务中心, 部署在相同云服务中心的分片之间网络延时低, 而部署在不同云服务中心的分片网络延时高, 这导致了分片之间的跨片交易处理开销不完全相同, 而当前所有基于负载的划分方案均假设分片间跨片交易处理开销相同. 于此同时, 由于成本、可用资源等方面的原因, 不同云服务中心申请到的计算或存储资源可能也不同, 因此划分给不同云服务中心的计算或存储任务要求并不是均衡的. 如何在这种复杂场景下实现高效的状态数据划分, 是将这类工作运用到云原生场景面临的新挑战.

3.5 数据分片方案总结

根据本节介绍的链上数据分片方案, 从是否交易分片、是否状态数据分片、对节点的信任假设、数据可用性、跨片交易数量、负载均衡、扩缩容开销、支持账户类型 8 个方面对这些方案进行对比, 并整理得到表 2.

1) 是否交易分片: 早期的数据分片工作不对系统交易处理进行分片, 节点在收到交易后, 首先恢复出交易需要的所有数据, 所有节点重复处理所有交易, 这类工作包括文献 [46,52,53,57,59]、CUB、BFT-Store、Partition Chain. 而其余数据分片工作同时对交易处理也进行了分片.

2) 是否状态数据分片: 所有工作都对交易数据进行了分片存储, 只有部分工作对状态数据进行了分片, 这一部分方案能够进一步减少系统的存储开销.

3) 节点的信任假设: 文献 [46] 和 CUB 假设所有节点诚实可信, 它们只关注节点可能因为发生故障而宕机, 无法处理恶意节点作恶的情况, 存在信任假设过强的问题. 其他方案均对节点采取拜占庭信任假设, 并能够应对恶意节点的作恶行为.

表 2　链上数据分片方案对比

类别	方案	交易分片	状态数据分片	节点信任假设	数据可用性	跨片交易数量	负载均衡	扩缩容开销	支持账户类型
基于群组的分片方案	文献[46]	×	×	CFT	低	—	√	小	UTXO
	CUB	×	×	CFT	低	—	√	小	UTXO
	Jidar	√	×	BFT	低	—	×	大	UTXO
	LDV	√	√	BFT	高	—	×	大	所有
基于门限秘密共享的分片方案	文献[52]	×	×	BFT	高	—	√	大	所有
	文献[53]	×	×	BFT	高	—	√	大	所有
基于编码的分片方案	BFT-Store	×	×	BFT	高	—	√	大	所有
	PartitionChain	×	×	BFT	高	—	—	小	所有
	文献[57]	×	×	BFT	高	—	√	大	所有
	文献[59]	×	×	BFT	高	—	√	大	所有
面向负载的分片方案	Chainspace	√	√	BFT	高	多	√	大	账户余额
	Monoxide	√	√	BFT	高	多	√	大	账户余额
	OmniLedger	√	√	BFT	高	多	√	大	UTXO
	RapidChain	√	√	BFT	高	多	√	大	UTXO
	OptChain	√	√	BFT	高	少	√	大	UTXO
	Shard Scheduler	√	√	BFT	高	多	√	大	所有
	BrokerChain	√	√	BFT	高	少	×	大	账户余额
	TxAllo	√	√	BFT	高	少	√	大	账户余额
	LB-Chain	√	√	BFT	高	多	√	大	所有
	S-Store	√	√	BFT	高	多	√	小	所有

4) 数据可用性: 由于文献 [46] 和 CUB 依赖对节点的特殊假设, 恶意节点的作恶行为可能导致部分数据不可恢复. Jidar 根据节点兴趣对数据进行划分, 冷门数据可能因为只被极少数节点存储导致可用性. LDV 设置少量的全节点来存储所有数据, 这种做法可以避免 Jidar 部分数据的可用性低的问题.

5) 跨片交易数量: Shard Scheduler 中的跨片交易数量可能较多, 因为状态划分时主要目标是负载均衡. LB-Chain 主要优化了低负载情况下分片之间的负载均衡, 没有考虑跨片交易数量. S-Store 基于一致性哈希对状态进行重划分, 忽略了负载中访问局部性, 易造成大量跨片交易. Chainspace、Monoxide、OmniLedger 和 RapidChain 对账户或 UTXO 进行随机划分, 会造成大量的跨片交易.

6) 负载均衡: Jidar 和 LDV 将每个数据划分至对其感兴趣的节点中, 忽略了划分到每个节点中的数据规模, 这可能导致部分节点的存储和交易处理负载过高. BrokerChain 基于 METIS 对交易图进行划分, METIS 的目标是每个分片中账户被访问的频率相近, 而不是负载相近. 其余方案通过在数据划分策略中添加一定程度的随机性, 或者将负载均衡作为划分算法目标函数中的一部分, 来保证节点或分片之间的负载均衡.

7) 扩缩容开销: 文献 [46]、CUB 以及 S-Store 均通过一致性哈希来对数据在节点或分片间进行划分, 系统在扩缩容时只会迁移一部分数据, 同时 S-Store 设计一种新型可验证数据结构减少重建状态树的开销. PartitionChain 基于聚合签名来避免在有新节点加入系统后, 旧节点需要对所有数据重新编码. 其余方案在系统发生扩容/缩容时, 数据需要在所有节点/分片间重新划分.

8) 支持账户类型: 文献 [46]、CUB、Jidar、OmniLedger、RapidChain、OptChain 只适用于采用 UTXO 账户模型的区块链系统, 而以太坊以及联盟链通常采取账户余额模型. Chainspace、Monoxide、BrokerChain 和 TxAllo 支持采用账户余额模型的区块链系统, 不支持 UTXO 账户系统, 其余方案不局限于某种账户类型.

早期的数据分片工作主要关注如何降低存储开销, 同时保证数据的可用性, 先对交易数据进行了分片, 随后进一步对状态数据进行了分片. 随着分片技术的发展, 数据分片工作开始与交易执行相结合, 研究如何降低数据分片

对交易处理性能的影响, 研究包括维持负载均衡、减少跨片交易数量、降低扩所容开销问题, 目标达到对系统的存储和执行同时扩展的目的.

数据分片技术被广泛应用于各种区块链项目, 包括 Filecoin[77]、Storj[78]等. Filecoin 是基于 IPFS[79]构建的中心化存储项目, 旨在全球范围提供一个去中心的存储市场, 它设置激励层鼓励矿工提供存储空间和数据查询服务. Storj 是 StorjLabs 发起的一个面相云平台的去中心化存储项目, 旨在提供更加安全、低成本的存储方式, 并且 Storj 已经兼容 AmazonS3 存储系统. Filecoin 和 Storj 都基于纠删码来降低系统的存储开销, 同时保证数据的可用性. 以太坊 2.0 和 Harmony 则采用全分片模式, 首先将节点划分成不同的组, 每个组的节点存储一部分交易和状态数据, 处理访问了相应状态的交易. Zilliqa 仅进行了交易分片, 没有对数据进行分片.

4 跨片交易正确性与性能优化

跨片交易的正确性与性能优化主要探讨如何在保证跨片交易原子性与隔离性的同时, 提高其处理性能, 减少对系统性能的影响. 为此, 本节首先将当前跨片交易处理方案分为 3 类进行综述: 基于两阶段提交的方案、基于确定性事务技术的方案以及基于共识的跨片交易方案. 此外, 还介绍了通过放宽对跨片交易原子性要求提高性能的方案. 最后, 对以上介绍的所有方案进行了总结, 并对相应技术的应用情况进行介绍.

4.1 基于两阶段提交的跨片交易处理方案

两阶段提交技术来源于分布式数据库领域, 它是处理分布式事务的经典方案. 跨片交易本质上也是一种分布式事务, 众多分片方案都基于两阶段提交来处理跨片交易, 由于包含多趟的片间消息传递, 处理延迟高、对系统性能影响大. OmniLedger 提出由客户端主导、基于两阶段提交的跨片交易处理协议, 并让它承担跨片交易协调任务. 当客户端收到一笔跨片交易时, 客户端向交易的输入涉及到的分片 (input shard) 发出锁定资产请求, 输入分片将客户端指定的资产标记为已经被花费, 随后向客户端回复接受该笔交易的证明. 当客户端收到所有输入分片的接受证明时, 将收到的证明以及交易发送给输出分片 (output shard), 输出分片在验证完消息后更新本地状态. OmniLedger 存在有被恶意用户攻击的可能, 恶意用户在处理完跨片交易的第 1 阶段后, 不继续处理第 2 阶段, 破坏交易的原子性. AHL 指定系统的一个分片来协调所有的跨片交易, 协议的安全性高, 但被指定的分片容易成为性能瓶颈. 图 4 展示了基于两阶段提交的跨片交易处理协议流程.

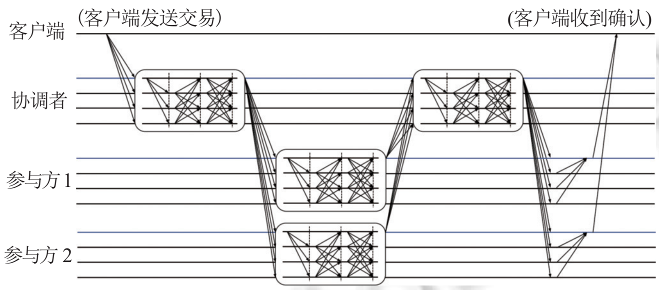

图 4　基于两阶段提交方案示意图

针对基于两阶段提交跨片交易处理协议性能低的问题, ByShard[80]将当前协议分为 3 种类型: 线型、星型和分布式, 同时考虑不同场景下用户对隔离级别、延时和中止率的要求, 提出了弹性跨片交易处理框架 OEM. 该框架共支持 18 种跨片交易处理协议, 以在交易吞吐、延时、交易中止率、隔离级别之间进行取舍. 用户能够根据场景选择不同的协议, 但协议根本上仍属于两阶段提交这一类.

最近一些工作开始从架构的角度优化跨片交易处理性能, Saguaro 提出了一种基于端边云场景的多层级架构, 旨在优化两阶段提交跨片交易处理方案. 在该架构中, 按照树形组织所有分片, 跨片参与方利用最近公共祖先协调跨片交易, 降低网络延迟对交易处理性能的影响. 另外, Saguaro 还提出了一种乐观的跨片交易处理协议, 其假设不同分片间的跨片交易不存在冲突, 参与者在收到跨片交易后直接在本地处理. 参与者会为每一笔已经执行但尚未

提交的交易维护一个状态依赖表, 并将依赖信息传输至上层分片, 由上层延迟检查交易中是否存在冲突. 在负载较为倾斜的情况下, 这种方式下会导致有较多的跨片交易因为冲突而被中止.

Pyramid[81]中的所有分片同样按照树形组织, 每个叶子分片负责存储系统的一部分的状态, 执行访问该状态的交易. Pyramid 通过牺牲存储开销来换取跨片交易处理性能, 父亲分片存储其所有孩子分片的全部状态数据, 这样孩子分片之间的跨片交易就能够直接在相应父亲分片中以片内交易进行处理. 然而, 数据的多副本存储会带来维护上下层分片之间数据一致性的开销, Pyramid 设计了一种基于两阶段提交和两阶段锁的协议来同步片间数据, 这种数据同步方式性能低, 并且在同步芙蓉过程中叶子分片中冲突的交易都会被中止, 在倾斜的负载下对系统性能的影响较大.

对于基于两阶段的跨片交易处理方案而言, 每个跨片交易的参与方仅能验证和执行本地的片内交易或跨片子交易, 并且无法确定其他参与方对交易是否验证通过. 跨片交易也可能与本地其他交易冲突, 因此处理跨片交易需要基于锁的协议, 同时依赖第三方分片来进行协调. 协调者经过两个阶段确认所有参与者对跨片交易的验证和预执行成功后, 才让每个参与者提交子交易. 这种多阶段、阻塞式的处理方式造成了基于两阶段提交的方案性能差, 针对这类跨片交易处理协议的优化在一定程度上提高了跨片交易性能, 但普遍存在局限性.

在云原生环境下, 基于两阶段提交的跨片交易处理方案的原子性以及性能均面临着新的挑战. 由于资源的动态性, 跨片交易协调者若发生故障将破坏交易的原子性, 并且导致大量交易长时间不能提交. 例如当协议第 1 阶段结束后, 协调者故障将会致跨片交易的第 2 阶段无法继续进行, 与其冲突的其他交易也将被中止. 同时, 分片之间可能跨越不同的地理区域, 不同分片间节点的通信延时高, 导致基于两阶段提交的跨片交易处理方案的开销进一步增加.

4.2 基于确定性事务技术的跨片交易处理方案

确定性事务处理技术[82−84]是数据库领域的一个重要技术分支. 系统首先对交易进行排序, 为每笔交易生成一个全局编号. 所有节点基于相同的快照、按照相同的顺序处理所有交易, 以此来保证节点间数据一致性. 近年来, 一些研究人员将确定性事务处理技术应用在跨片交易处理. 通过让跨片交易参与方确定性地处理跨片交易, 降低在参与者之间协调跨片交易开销. 在交易执行期间, 节点需要相互发送最新读写集消息, 以保证所有节点基于相同的快照处理交易, 协议流程如图 5 所示.

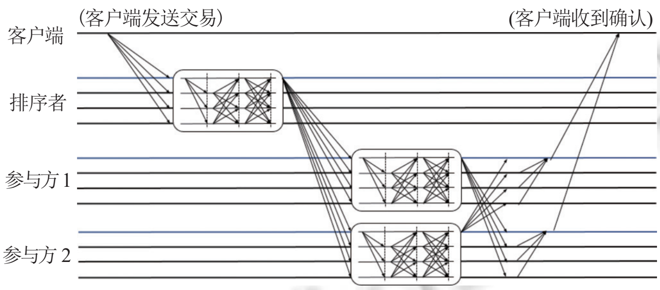

图 5 基于确定性排序方案示意图

Meepo[85]基于确定性事务思想处理跨片交易, 系统以 cross-epoch 为单位处理系统中的跨片交易, 处理完一个 cross-epoch 中所有跨片交易后再处理下一个 cross-epoch. 具体而言, 在每个 cross-epoch 开始时, 各个分片将需要处理的跨片交易及其相应的读写集发送给其他参与方, 参与在收齐所有消息后按照相同的顺序串行地处理这些交易. 随后, 分片开始处理下一个 cross-epoch 的跨片交易, 直到当前区块中的所有跨片交易处理完成, 再开始生成新的区块. Meepo 假设所有分片位于同一个数据中心, 片间网络通信延时低, 每个 cross-epoch 开始时分片之间的通信不会带来太多的性能损失.

CoCoS[86]使用单一排序服务对所有分片的交易进行排序, 每个分片在接收到交易及顺序后基于排序锁并发执行交易, 在处理的过程中分片间也需要发送相应的读写集. 这种设计存在以下几个问题: 首先, 由于所有交易都要

经过单个排序服务排序, 排序服务可能成为系统性能瓶颈. 其次, 即使是分片交易也要由全局排序服务定序, 然后发送到执行节点, 这会增加片内交易的延迟. 此外, CoCoS 假设排序服务可信, 并且所有交易的读写集提前已知.

为了打破交易读写集必须提前已知的假设, Prophet[87]首先让不同参与方的部分节点对跨片交易进行预执行, 将得到的交易读写集发送给排序服务. 为了防止恶意节点向排序服务报告错误的读写集, 参与方在真正执行跨片交易时候会验证读写集与预执行节点所提供的读写集是否相同. Prophet 也基于唯一的排序服务来对所有交易排序, 因此与 CoCoS 一样, 交易排序服务可能成为性能瓶颈.

相较于基于两阶段提交的处理方案, 基于确定性事务技术的方案虽然能够减少跨片交易协调的开销, 但仍然存在一些问题. 首先, 目前基于确定性的方案依赖第三方可信交易排序服务, 导致系统的可扩展性较差. 其次, 每个分片的片内交易也需要经过排序服务定序后再发送到各自的分片, 与传统处理方式中每个分片独自对片内交易排序相比, 这增加了交易的平均延迟. 最后, 跨片交易参与者在执行交易的过程中需要向其他分片发送必要的读写集, 这也会带来额外的网络开销和等待时间.

基于确定性事务技术的跨片交易处理方案在云原生环境下主要面临安全性低和网络延时高两个挑战. 基于确定性事务技术的方案通常依赖一个全局的排序服务为所有交易定序, 随着系统业务量的增加, 需要对排序服务进行扩容, 如果扩容导致排序者被恶意节点控制, 将会破坏系统安全性. 另外, 由于节点可能分布在不同的地理区域的数据中心, 跨地域节点之间通信延时高. 基于确定性事务的处理方案首先需要将排序结果发送给不同的参与方, 同时参与方之间需要频繁的交换读写集信息, 这些都会导致云原生环境中这类方案会的交易处理延时较高.

4.3 基于共识的跨片交易处理方案

基于共识的跨片交易处理方案通过让所有参与者对跨片交易达成共识, 来保证跨片交易的 ACID. 与基于两阶段提交和确定性事务的处理方案相比, 基于共识的处理方案不需要引入额外的排序服务或协调者, 跨片交易的处理仅需要交易的参与方参与.

SharPer 基于共识来处理跨片交易, 通过在所有参与方的节点之间运行 PBFT 对跨片交易达成共识. 具体而言, 共识的主节点首先向所有参与方节点广播共识 *propose* 消息, 节点收到消息并验证交易通过后, 向所有参与方的节点广播 *accept* 消息, 假设每个分片中恶意节点数量为 f, 每个节点收齐来自每个参与者至少 $2f$+1 个 *accept* 消息后, 表明分片中绝大多数节点同意提交交易, 随即向所有参与方的节点广播 *commit* 消息. 若节点收到收齐来自每个参与者至少 $2f$+1 个 *commit* 消息, 跨片交易被提交. 当跨片交易参与方较多, 或者每个分片中的节点数目较多时, SharPer 存在网络开销大的问题, 图 6 展示了 SharPer 跨片处理协议流程.

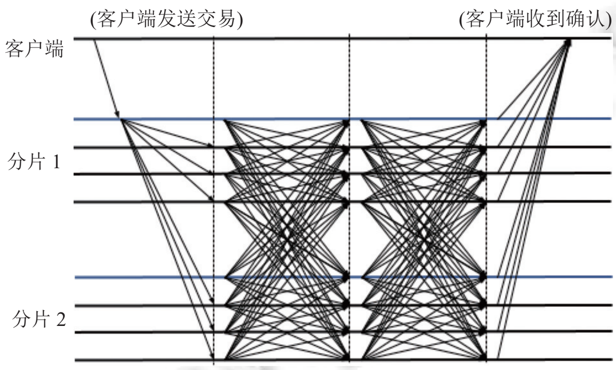

图 6　基于共识的跨片交易处理

Chainspace 通过层级化片间共识来降低共识开销. 客户端首先向所有参与方发送跨片交易, 每个参与方收到跨片交易后首先独自对交易进行验证. 然后, 分片将包含了本地节点签名的验证结果发送给其他参与方, 让其他参与方对共识结果进行验证. 如果分片收齐并验证了其他参与方也决定提交交易, 那么表明交易可以立即提交, 协议流程如图 7 所示. 层级化的片间共识流程增加了共识协议步骤, 但降低了共识中的通信复杂度, 适用于参与方数量或者节点数目较多的场景.

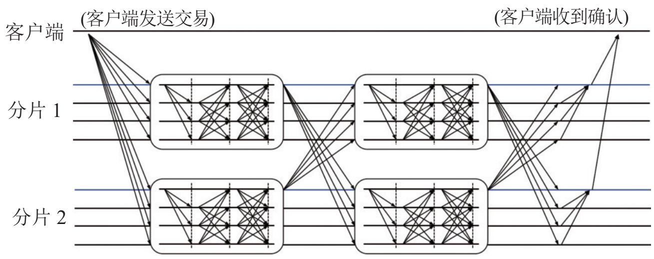

图 7　层级化跨片共识示意图

RingBFT[88]基于线型的层级共识协议进一步降低了片间通信负载度. RingBFT 限制每笔跨片交易同时只能有一个参与方对其进行投票, 发送交易给下一个参与方时候带上其他参与方的投票, 当最后一个参与方验证结束后就可以根据投票结果决定是否提交交易. 由于分片可能同时处理多笔跨片子交易, 为了避免产生死锁, RingBFT 将所有分片按照编号大小组织成环, 每次跨片共识从编号最小的参与方出发, 沿着特定方向依次让环上属于交易参与者的分片投票.

基于共识的跨片交易处理协议优势在于无需依赖第三方的协调者或排序服务, 系统中参与方不同的跨片交易互不影响, 可以并行处理, 具有较好的可扩展性. 但这类方案的代价是处理过程网络开销大, 尤其是在参与方数量多, 或者节点数量多的场景. 相关的优化方案也围绕如何降低网络开销来展开, 包括前面介绍的线性共识、层级共识等策略.

如何降低云原生环境下跨片交易延时是基于共识的方案主要面临的新挑战. 云原生平台可能分布在不同的地理区域, 这意味着不同分片的节点可能分布在不同数据中心, 不同数据中心的节点之间通信延时高. 而基于共识的跨片交易处理方案依赖大量的消息传递, 特别是 PBFT 等协议需要包含了多个阶段来容忍败拜占庭节点的攻击. 层级的跨片共识方案只能降低网络复杂度, 无法降低跨片交易延时.

4.4　最终原子性跨片交易处理方案

前面介绍的方案都以严格保证跨片交易的原子性为前提, 保证正确性同时减少对系统性能的影响. 与此同时, 存在另一部分工作, 它们通过放宽对跨片交易的原子性要求, 跨片交易只需要最终达到原子性, 设计了更加高效的协议降低跨片交易对性能的影响.

RapidChain、Monxide 与 SSChain 的做法相似, 由于只支持 UTXO 账户模型, 它们将跨片交易拆分成分别在输入分片和输出分片执行的子交易. 客户端首先将交易发送给输出分片, 输出分片拆分交易, 然后向输入分片发送其需要处理的子交易. 当输出分片验证所有输入分片的子交易已经提交, 再执行本地需要处理的部分. 在这些方案中, 跨片交易在输入分片和输出分片不需要同时提交, 在输入分片将本地子交易执行完后, 本地其他交易可以继续执行, 异步验证跨片交易. 这种方式能够极大地降低跨片交易对性能的影响. 但这种方案不能保证交易的隔离性, 假设有多笔跨片交易同时访问了不同分片的相同状态, 由于子交易不会对资产进行上锁而是直接将余额花费掉, 若这些跨片交易在不同分片中的顺序不同且分片中的余额只够处理一部分交易, 这可能导致最终所有的交易都无法提交.

BrokerChain 通过引入做市商账户, 在基于账户-余额模型的区块链系统中实现跨片交易最终原子性. 做市商账户的作用是扮演跨片交易的中间商, 允许将跨片交易拆分成多笔子交易, 它在多个分片中具有相同的地址和充足的余额. 具体而言, 当发生转账交易时, 源分片首先通过片内交易将资产转移至本分片内的做市商账户, 随后将证明发送给目标分片. 目标分片收到证明后, 再从做市商账户在本分片中余额扣除一部分资产转移到目标账户. 源分片在处理完本地跨片子交易后可以继续处理其他交易, 异步地确认目标分片中的子交易是否执行成功. BrokerChain 仅能够支持普通转账逻辑, 不能支持复杂合约跨片交易的最终原子性. 此外, 同一市商账户在多个分片中都被存储, 给存储也带来了一定冗余.

在对数据一致性要求不高的场景中, 牺牲跨片交易的原子性允许协议以无锁、异步的方式处理跨片交易, 能

够将跨片交易对系统的影响降到最低. 然而, 这些工作仅支持 UTXO 账户模型系统, 或者账户余额模型下的简单转账合约, 一部分工作还存在假设过强的问题, 缺乏通用性.

4.5 跨片交易处理技术总结

根据本节介绍的现有跨片交易处理解决方案, 从是否完全状态分片、跨片交易处理延时、跨片交易处理吞吐、隔离性、支持账户模型、安全性这 6 个方面进行了对比, 并整理得到表 3.

表 3 跨片交易处理方案对比

类别	方案	完全状态分片	跨片交易处理延时	跨片交易处理吞吐	隔离性	支持账户类型	安全性
基于两阶段提交的跨片方案	OmniLedger	√	高	低	√	UTXO	低
	AHL	√	高	低	√	所有	高
	ByShard	√	高	低	√	所有	高
	Saguaro	×	中	中	√	所有	高
	Pyramid	×	低	高	√	所有	高
基于确定性事务的跨片方案	Meepo	√	低	高	√	所有	高
	CoCoS	√	低	高	√	所有	高
	Prophet	√	低	高	√	所有	高
基于共识的跨片方案	SharPer	√	低	高	√	所有	高
	Chainspace	√	高	低	√	账户余额	高
	RingBFT	√	高	低	√	所有	高
最终原子性跨片方案	RapidChain	√	—	高	×	UTXO	高
	Monxide	√	—	高	×	UTXO	高
	SSChain	√	—	高	×	UTXO	中
	BrokerChain	×	—	高	×	账户余额	高

1) 是否完全状态分片: 完全状态分片是指分片之间维护的状态数据没有交集, Saguaro 的上层分片存储下层分片的交易快照, 用于在乐观处理跨片交易时检查冲突. Pyramid 通过牺牲存储换取跨片交易处理性能, 上层父亲分片存储了其孩子分片中的所有状态数据. BrokerChain 将相同地址的账户存储在多个分片中, 以能够在采用账户余额区块链系统中支持跨片交易最终原子性, 不同分片之间存储的账户存在冗余.

2) 跨片交易处理延时: OmniLedger、AHL 和 ByShard 基于两阶段提交处理跨片交易, 片间网络通信的阶段多, 导致了延时高. Saguaro 选择靠近参与方的上层分片协调跨片交易, 一定程度上减少了基于两阶段提交方案的处理延时. 为了减少基于共识的跨片交易处理方案网络开销大的问题, 通过分层 (Chainspace)、线型投票 (RingBFT) 的共识方式导致协议的阶段多, 造成了跨片交易延时高. 基于确定性事务的方案由于分片间协调开销小, 因此交易延时也低.

3) 跨片交易处理吞吐: 基于两阶段提交的跨片交易处理方案性能最低, Saguaro 的多层级架构在一定程度上提高了这类跨片交易处理方案性能. Pyramid 中的跨片交易可以在上层分片以片内交易的方式处理, 因此吞吐较高. Chainspace 与两阶段提交方案都存在较多阶段的片间通信, 导致了吞吐较低 RingBFT 每次仅允许一个参与方对跨片交易投票, 吞吐较低. SharPer 的通信阶段少, 吞吐较高. 基于确定性事务处理技术以及最终原子性的方案吞吐最高.

4) 隔离性: 实现最终原子性跨片交易的方案 (RapidChain、Monxide、SSChain 和 BrokerChain) 让每个参与方以无锁、异步的方式处理各自的跨片子交易, 分片可以对正在被其他跨片交易访问的账户操作, 无法保证交易处理的隔离性. 基于两阶段提交的方案通过对被访问的状态上锁来保证隔离性. 基于确定性事务和共识的方案因为会事先确定所有交易的顺序, 因此也可以保证交易的隔离性.

5) 支持账户类型: OmniLedger、RapidChain、Monxide 和 SSChain 只支持 UTXO 账户模型, 这些工作都面向公链区块链系统, Chainspace 和 BrokerChain 支持账户余额模型, 其他方案不限账户类型.

6) 协议安全性: 绝大多数跨片交易处理协议都具有比较高的安全性. OmniLedger 基于客户端来协调跨片交

易, 但客户端有可能在过程中作恶, 例如在处理完第 1 阶段后故意沉默, 破坏交易原子性. SSChain 设置激励层让节点自发地组成一个根链来验证所有跨片交易, 但如果根链的算力没有超过全网算力的 50%, 跨片交易的最终原子性将无法保证.

在不放松对跨片交易原子性要求的前提下, 当前跨片处理方案主要包括了基于两阶段提交、基于确定性事务技术以及基于共识这 3 类. 这些工作在保证跨片交易 ACID 的基础上, 对跨片交易性能优化进行了探索, 主要从减少跨片交易处理的协调开销、增加交易处理的并发度、减少网络开销等方面去优化. 此外, 另一部分研究人员开始从分片架构的角度来优化跨片交易性能, 如 Saguaro、Pyramid 按照树状结构组织分片, 对分片任务进行进一步分工, 特别是 Saguaro, 借助端边云架构的可伸缩性能够灵活地对上层进行扩展.

在跨片交易处理技术应用方面. 绝大多数公链分片项目都基于最终原子性来处理跨片交易, 联盟链下的方案更加多样. 在以太坊 2.0 中, 信标链在收到跨片交易后, 首先将跨片交易通知给分片链, 分片链在收到跨片交易后开始处理交易. 信标链定期收集分片链的状态证明, 验证所有分片链最终都处理了跨片交易. Harmony 和 NEAR 的跨片交易处理方式与 RapidChain 类似, 跨片交易在源分片中处理完毕后, 再将交易以及相应的证明转发到目标分片进行处理. 由微众银行开发的 FISCO BCOS 采用两阶段提交协议来处理跨片/链交易, 让第三方中继链扮演跨片交易的协调者. 趣链[89]也基于最终原子性来处理跨片交易, 源分片收到跨片交易先处理, 随后将跨片交易转发给中继链, 由中继链转发给目标分片继续处理, 若跨片交易在目标分片执行失败, 中继链通知源分片回滚交易.

5 方案整体对比

为了更全面地评估前面介绍的所有分片方案, 本节围绕前面提到的 3 个关键技术问题, 对所有分片方案同时在这 3 个维度上的整体表现进行对比. 我们对前面提到的指标进行整合, 总结出 6 项评价综合指标并整理得到表 4, 包括系统架构、存储可扩展性、执行可扩展性、系统安全性、数据可用性以及通用性. 其中, 影响存储可扩展性的因素包括交易数据、状态数据是否分片、系统中数据冗余程度. 影响执行可扩展性的因素包括跨片交易数量、跨片交易处理性能、系统负载均衡度. 影响通用性的因素是是否依赖特殊的假设或条件.

(1) 系统架构: 早期的分片系统基本都采取了扁平化分片架构, 系统每次只能横向增加分片数目, 由于片间协调成本高, 限制这些系统可扩展性的最大因素在于跨片交易. AHL、CoCoS、Prophe 和 SSChain 采用两层分片架构, 其中 AHL 唯一的第 2 层分片负责协调所有跨片交易. CoCoS 和 Prophet 位于第 2 层的排序服务负责为所有分片的交易定序. SSChain 位于第 2 层的根链负责验证所有跨片交易的最终原子性. Pyramid 和 Saguaro 均采取了多层级分片架构.

(2) 存储可扩展性: 文献 [46,52,53,57,59], 以及 CUB、Jidar、BFT-Store、PartitionChain、Elastico 和 Pyramid 的存储可扩展性都较低. 这是因为 Elastico 的交易和状态数据均未分片, Pyramid 通过牺牲一部分存储来提高跨片交易处理性能. 其他方案仅对交易数据进行了分片, 状态数据未分片. LDV、Saguaro 和 SSChain 对交易数据和状态数据都进行了分片存储, 但不同分片之间存在一定的冗余度. LDV 中热点数据可能被绝大多数的节点存储, Saguaro 中的上层分片需要存储下层分片的交易快照, SSChain 根链中的节点会存储所有分片的数据, 这些因素导致了这 3 个系统存储可扩展性差于前面介绍的几个系统. 除去这些方案, 表中其余方案的不同分片节点存储的交易和状态数据完全没有交集, 因此存储可扩展性最高.

(3) 执行可扩展性: 状态数据未分片的系统不会触发有跨片交易, 因此执行可扩展性最高, 这些系统包括文献 [46,52,53,57,59] 和 CUB、Jidar. LDV 虽然对状态数据进行了分片, 但节点会存储其感兴趣的所有状态. Pyramid 通过牺牲一定存储扩展性, 来换得较高的执行扩展性. OptChain 和 TxAllo 基于系统负载研究状态划分策略, 通过减少跨片交易的同时兼顾负载均衡来获得较高的执行扩展性. Meepo、CoCoS 和 Prophet 通过降低跨片交易处理开销来提高执行可扩展性. RapidChain、Monxide、BrokerChain、SSChain 通过异步地保证跨片交易原子性来避免跨片交易对执行性能的影响. Saguaro 虽然通过多层级优化跨片交易开销, 但依旧使用两阶段提交来处理跨片交易, 跨片交易处理开销依旧较高. OmniLedger、Chainspace、RapidChain、Monxide、Elastico、S-Store、

Shard Scheduler 和 LB-Chain 由于状态划分策略的缺陷, 存在负载不均衡或者跨片交易数量过多的问题, 造成执行可扩展性差. RingBFT 的跨片交交易参与方只能串行地为跨片交易投票, 导致跨片交易处理延时高. AHL、ByShard、Chainspace、SharPer、CoChain 和 Benzene 由于跨片交易处理开销大, 也导致了执行可扩展性差.

表 4　分片方案整体对比

系统架构	方案名称	存储可扩展性	执行可扩展性	系统安全性	数据可用性	通用性
扁平化分片架构	文献[46]	低	高	低	高	弱
	CUB	低	高	低	高	弱
	Jidar	低	高	高	低	强
	LDV	中	高	高	高	强
	文献[52]	低	高	高	高	强
	文献[53]	低	高	高	高	强
	BFT-Store	低	高	高	高	强
	PartitionChain	低	高	高	高	强
	文献[57]	低	高	高	高	强
	文献[59]	低	高	高	高	强
	OptChain	高	高	—	高	强
	Shard Scheduler	高	低	—	高	强
	BrokerChain	高	低	—	高	弱
	TxAllo	高	高	—	高	强
	LB-Chain	高	低	—	高	强
	S-Store	高	低	—	高	强
	ByShard	高	低	—	高	强
	Chainspace	高	低	—	高	强
	RingBFT	高	高	—	高	强
	SharPer	高	低	高	高	强
	RapidChain	高	高	高	高	强
	Monxide	高	高	高	高	强
	Elastico	低	高	低	高	强
	OmniLedger	高	低	高	高	强
	CoChain	高	低	高	高	强
	Benzene	高	低	高	高	强
	Meepo	高	高	高	高	弱
	AHL	高	低	高	高	强
两层分片架构	CoCoS	高	高	高	高	弱
	Prophet	高	高	高	高	强
	SSChain	中	高	高	高	强
多层级分片架构	Pyramid	低	高	高	高	强
	Saguaro	中	中	高	高	弱

(4) 系统安全性: 文献 [46]、CUB 依赖系统中的节点诚实可信, 无法抵抗系统中恶意节点的攻击. 对于只对数据进行分片、交易不分片的系统而言, 系统的安全性始终能够较好的保证, 需要考虑的只是数据可用性. Elastico 由于划分出的分片中节点数量较少, 每个分片容易被恶意节点控制. 其余方案主要通过增加分片中的节点数量、更安全的节点分配算法、分片相互监督验证、硬件加持等方式增加了分片的安全性.

(5) 数据可用性: 绝大多数的分片方案都基于数据冗余存储以及密码学来保证数据的高可用. Jidar 中的每个节点只存储其感兴趣的数据, 这会导致一些冷门数据仅存在少数节点, 甚至没有节点存储这部分数据, 造成这部分数据的可用性降低. LDV 将历史数据归档在少量可信第三方节点中来降低存储开销, 但这会带来数据丢失的风险. 对于采取了全分片的系统而言, 由于它们的安全性普遍较高, 由每个分片来负责存储一部分数据能够保证较好的

数据可用性.

(6) 通用性: 文献 [46] 和 CUB 的通用性弱, 它们依赖所有节点诚实可信这一假设. BrokerChain 为了降低跨片交易处理开销, 引入做市商账户, 并且假设做市商账户中的余额充足. Saguaro 主要针对端边云场景下分片区块链的跨片交易进行优化, 场景具有局限性. Meepo 假设所有分片位于同一个数据中心, 片间的通信延时低, 否则每个 cross-epoch 内的片间通信会对性能造成较大的影响. CoCoS 假设全局唯一的交易排序服务诚实可信, 同样存在通用性较弱的问题.

6 挑战与总结

6.1 区块链分片研究挑战

近些年, 区块链分片技术得到了显著发展, 但它仍然是一个充满许多挑战的研究领域. 本文在此列出一些重要并且仍充满挑战的问题.

1) 负载感知的分片动态扩缩容: 当负载发生变化时, 系统通常需要增加或减少分片实现对系统的扩缩容. 在传统的分片区块链系统中, 分片数量变化需要对所有状态重新划分, 这会带来巨大额外的开销. 当前主要通过一致性哈希来对划分规则进行改进, 当增加或者减少分片时, 系统中只有部分状态需要重新划分, 避免了在片间引发大量的状态迁移. 然而, 这样的数据划分规则完全忽视了负载中的访问局部性特征, 会造成系统中出现大量的跨片交易, 对系统性能造成严重影响. 因此, 有必要研究一种负载感知的分片动态扩缩容方案, 在减少重划分开销的同时, 结合考虑负载中的访问模式, 在两者之间取得一个平衡.

2) 高性能可扩展的跨片交易处理: 当前跨片交易处理方案普遍存在性能低或者可扩展性差的问题. 基于两阶段提交的方案由于多阶段、阻塞式的处理方式, 跨片交易处理性能低. 虽然多层级的两阶段工作能够对跨片交易处理能力进行扩展, 但协议本质上仍属于两阶段提交的范畴. 基于共识的方案需要所有参与方的节点对跨片交易达成共识, 存在网络开销大、交易延时较高的问题. 基于确定性方案的排序服务需要预先对所有交易进行定序, 且当前工作只支持一个的排序服务, 不具有可扩展性. 总体来说, 目前仍缺乏一种协议具有较高的跨片交易处理性能, 同时具备可扩展性.

3) 面向复杂合约的跨片交易处理: 当前跨片交易处理方案存在对合约假设过强的问题, 不具有通用性. 具体而言, 这些工作通常假设合约逻辑简单, 如账户间转账, 合约通常涉及参与方少、读写状态少、每个跨片子交易的执行开销低. 然而, 联盟链的业务通常较为复杂, 合约可能涉及较多的参与方和状态, 并且交易执行开销大, 这些问题加剧了跨片交易对性能的负面影响. 最后, 现有方案仅支持非交互式跨片交易, 而交互式交易会导致交易持锁时间增加, 目前缺乏对这类交易的研究.

4) 高效低开销节点数据同步: 在对系统中的节点进行重划分时, 新节点在加入某个分片后, 需要从旧节点拉取分片的所有历史数据, 并在本地重新构建相应的可验证索引. 这个步骤会给节点带来较大的网络和 I/O 开销, 并影响节点整体的性能. 当前相关方案每次让新节点仅同步近期片内被访问的状态数据, 由单独的分片存储所有分片的冷数据并提供查询. 然而, 这种方式下一旦节点的交易需要访问冷数据, 就会引发跨片数据访问, 增加了交易处理延时. 因此, 亟需一种方案能够让新节点尽可能快速同步所有历史数据, 同时具有较低的网络和 I/O 的开销.

5) 面向云原生环境的区块链分片技术: 基于云原生平台的区块链分片技术存在诸多挑战. 首先, 如何在节点资源动态性的前提下, 避免恶意节点对系统造成攻击, 持续保证系统的安全是首要解决的问题. 其次, 随着多云 (multi-cloud) 时代的到来, 企业通常将应用部署在不同的云服务中心, 这些数据中心通常跨越不同的地理区域, 节点之间通信延时高, 造成交易处理开销大. 同时, 不同的云服务之间的异构性也造成了云原生场景下的区块链分片任务更加复杂. 这类工作目前处于起步阶段, 亟需研究相应的方案.

6.2 总　结

区块链分片技术是近年来热门的研究方向, 是最直接和最有潜力的链上扩容方式. 本文首先对现有区块链分片的类型进行梳理, 并分析当前区块链分片需要主要节点的关键技术问题. 随后, 本文分别综述、讨论了各个关键

技术问题的研究现状, 包括了节点划分安全性、高效数据分片以及跨片交易处理. 接着, 围绕这 3 个维度, 再对所有方案进行全面的分析和对比. 最后, 本文指出了几个值得进一步探索的研究方向. 总体而言, 区块链分片技术近些年得到了快速发展, 但仍然存在许多技术问题亟需解决, 特别是随着云原生技术的发展, 面向新架构的区块链分片技术更是一个充满机遇和挑战的领域.

References:

[1] Dong HW, Zhang C, Li GL, Feng JH. Survey on cloud-native databases. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 899–926 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6952.htm [doi: 10.13328/j.cnki.jos.006952]

[2] Korpela K, Hallikas J, Dahlberg T. Digital supply chain transformation toward blockchain integration. In: Proc. of the 50th Hawaii Int'l Conf. on System Sciences. Waikoloa Village: HICSS, 2017. 1–10.

[3] Azaria A, Ekblaw A, Vieira T, Lippman A. MedRec: Using blockchain for medical data access and permission management. In: Proc. of the 2nd Int'l Conf. on Open and Big Data (OBD). Vienna: IEEE, 2016. 25–30. [doi: 10.1109/OBD.2016.11]

[4] Kamilaris A, Fonts A, Prenafeta-Boldύ FX. The rise of blockchain technology in agriculture and food supply chains. Trends in food Science & Technology, 2019, 91: 640–652. [doi: 10.1016/j.tifs.2019.07.034]

[5] Shao QF, Jin CQ, Zhang Z, Qian WN, Zhou AY. Blockchain: Architecture and research progress. Chinese Journal of Computers, 2018, 41(5): 969–988 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2018.00969]

[6] Shao QF, Zhang Z, Zhu YC, Zhou AY. Survey of enterprise blockchains. Ruan Jian Xue Bao/Journal of Software, 2019, 30(9): 2571–2592 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5775.htm [doi: 10.13328/j.cnki.jos.005775]

[7] 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, Weed Cocco S, 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]

[8] Li HZ, Chen YJ, Shi X, Bai XQ, Mo N, Li WL, Guo R, Wang Z, Sun Y. FISCO-BCOS: An enterprise-grade permissioned blockchain system with high-performance. In: Proc. of the 2023 Int'l Conf. for High Performance Computing, Networking, Storage and Analysis. Denver: ACM, 2023. 68. [doi: 10.1145/3581784.3607053]

[9] Kim S, Kwon Y, Cho S. A survey of scalability solutions on blockchain. In: Proc. of the 2018 Int'l Conf. on Information and Communication Technology Convergence (ICTC). Jeju: IEEE, 2018. 1204–1207. [doi: 10.1109/ICTC.2018.8539529]

[10] Xie JF, Yu FR, Huang T, Xie RC, Liu J, Liu YJ. A survey on the scalability of blockchain systems. IEEE Network, 2019, 33(5): 166–173. [doi: 10.1109/MNET.001.1800290]

[11] Yang D, Long CN, Xu H, Peng SL. A review on scalability of blockchain. In: Proc. of the 2nd Int'l Conf. on Blockchain Technology. Hilo: ACM, 2020. 1–6. [doi: 10.1145/3390566.339166]

[12] Eklund PW, Beck R. Factors that impact blockchain scalability. In: Proc. of the 11th Int'l Conf. on Management of Digital Ecosystems. Limassol: ACM, 2019. 126–133. [doi: 10.1145/3297662.3365818]

[13] Chauhan A, Malviya OP, Verma M, Mor TS. Blockchain and scalability. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security Companion (QRS-C). Lisbon: IEEE, 2018. 122–128. [doi: 10.1109/QRS-C.2018.00034]

[14] Xu C, Zhang C, Xu JL, Pei J. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing. Proc. of the VLDB Endowment, 2021, 14(11): 2314–2326. [doi: 10.14778/3476249.3476283]

[15] Xu ZH, Chen L. L2chain: Towards high-performance, confidential and secure layer-2 blockchain solution for decentralized applications. Proc. of the VLDB Endowment, 2022, 16(4): 986–999. [doi: 10.14778/3574245.3574278]

[16] Corbett JC, Dean J, Epstein M, Fikes A, Frost C, Furman JJ, Ghemawat S, Gubarev A, Heiser C, Hochschild P, Hsieh W, Kanthak S, Kogan E, Li HY, Lloyd A, Melnik S, Mwaura D, Nagle D, Quinlan S, Rao R, Rolig L, Saito Y, Szymaniak M, Taylor C, Wang R, Woodford D. Spanner: Google's globally-distributed database. In: Proc. of the 10th USENIX Symp. on Operating Systems Design and Implementation. Hollywood: USENIX Association, 2012. 251–264.

[17] Wood G. Ethereum: A secure decentralised generalised transaction ledger. 2014. https://ethereum.github.io/yellowpaper/paper.pdf

[18] The Zilliqa Team. The Zilliqa technical whitepaper. 2017. https://docs.zilliqa.com/whitepaper.pdf

[19] Luu L, Narayanan V, Zheng CD, Baweja K, Gilbert S, 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]

[20] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf

[21] Wu CY, Amiri MJ, Qin HY, Mehta B, Marcus R, Loo BT. Towards full stack adaptivity in permissioned blockchains. Proc. of the VLDB Endowment, 2024, 17(5): 1073–1080. [doi: 10.14778/3641204.3641216]

[22] Wu CY, Mehta B, Amiri MJ, Marcus R, Loo TB. AdaChain: A learned adaptive blockchain. Proc. of the VLDB Endowment, 2023, 16(8): 2033–2046. [doi: 10.14778/3594512.35945]

[23] Tong X, Zhang Z, Jin CQ, Zhou AY. Blockchain for end-edge-cloud architecture: A survey. Chinese Journal of Computers, 2021, 44(12): 2345–2366 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.02345]

[24] Amiri MJ, Lai ZL, Patel L, Loo TB, Lo E, Zhou WC. Saguaro: An edge computing-enabled hierarchical permissioned blockchain. In: Proc. of the 39th Int'l Conf. on Data Engineering (ICDE). Anaheim: IEEE, 2023. 259–272. [doi: 10.1109/ICDE55515.2023.00027]

[25] Huang HW, Kong W, Peng XW, Zheng ZB. Survey on blockchain sharding technology. Computer Engineering, 2022, 48(6): 1–10 (in Chinese with English abstract). [doi: 10.19678/j.issn.1000-3428.0063887]

[26] Tan PL, Xu T, Tu RX. A review of research on blockchain sharding techniques. Computer Science, 2024, 51(11): 307–320 (in Chinese with English abstract).

[27] Yu GS, Wang X, Yu K, Ni W, Zhang JA, Liu RP. Survey: Sharding in blockchains. IEEE Access, 2020, 8: 14155–14181. [doi: 10.1109/ACCESS.2020.2965147]

[28] Berger C, Schwarz-Rüsch S, Vogel A, Bleeke K, Jehl L, Reiser HP, Kapitza R. Sok: Scalability techniques for BFT consensus. In: Proc. of the 2023 IEEE Int'l Conf. on Blockchain and Cryptocurrency (ICBC). Dubai: IEEE, 2023. 1–18. [doi: 10.1109/ICBC56567.2023.10174976]

[29] 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.

[30] Kokoris-Kogias E, Jovanovic P, Gasser L, Gailly N, 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]

[31] Micali S, Rabin M, Vadhan S. Verifiable random functions. In: Proc. of the 40th Int'l Conf. on Foundations of Computer Science. New York: IEEE, 1999. 120–130. [doi: 10.1109/SFFCS.1999.814584]

[32] 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]

[33] Sen S, Freedman MJ. Commensal cuckoo: Secure group partitioning for large-scale services. ACM SIGOPS Operating Systems Review, 2012, 46(1): 33–39. [doi: 10.1145/2146382.2146389]

[34] Abraham I, Devadas S, Nayak K, Ren L. Brief announcement: Practical synchronous byzantine consensus. In: Proc. of the 31st Int'l Symp. on Distributed Computing. Vienna: DISC, 2017. 41:1–41:4. [doi: 10.4230/LIPIcs.DISC.2017.41]

[35] Amiri MJ, Agrawal D, El Abbadi A. SharPer: Sharding permissioned blockchains over network clusters. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 76–88. [doi: 10.1145/3448016.3452807]

[36] Dang H, Dinh TTA, Loghin D, Chang EC, Lin Q, Ooi BC. Towards scaling blockchain systems via sharding. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 123–140. [doi: 10.1145/3299869.3319889]

[37] Costan V, Devadas S. Intel SGX explained. IACR Cryptology ePrint Archive, 2016. http://css.csail.mit.edu/6.858/2020/readings/costan-sgx.pdf

[38] Wang JP, Wang H. Monoxide: Scale out blockchains with asynchronous consensus zones. In: Proc. of the 16th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX Association, 2019. 95–112.

[39] Chen H, Wang YJ. SSChain: A full sharding protocol for public blockchain without data migration overhead. Pervasive and Mobile Computing, 2019, 59: 101055. [doi: 10.1016/j.pmcj.2019.101055]

[40] Li MZ, Lin Y, Zhang J, Wang W. CoChain: High concurrency blockchain sharding via consensus on consensus. In: Proc. of the 2023 IEEE Conf. on Computer Communications. New York: IEEE, 2023. 1–10. [doi: 10.1109/INFOCOM53939.2023.10228892]

[41] Cai ZT, Liang JY, Chen WH, Hong ZC, Dai HN, Zhang JT, Zheng ZB. Benzene: Scaling blockchain with cooperation-based sharding. IEEE Trans. on Parallel and Distributed Systems, 2023, 34(2): 639–654. [doi: 10.1109/TPDS.2022.3227198]

[42] Buterin V, Hernandez D, Kamphefner T, Pham K, Qiao Z, Ryan D, Sin J, Wang Y, Zhang YX. Combining GHOST and casper. arXiv:2003.03052, 2020.

[43] Avarikioti Z, Karakostas D. Harmony technical whitepaper. 2019. https://harmony.one/whitepaper.pdf

[44] NEAR Prorocol. The NEAR white paper. 2021. https://pages.near.org/papers/the-official-near-white-paper/

[45] Boneh D, Bonneau J, Bünz B, Fisch B. Verifiable delay functions. In: Proc. of the 38th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2018. 757–788. [doi: 10.1007/978-3-319-96884-1_25]

[46] Abe R, Suzuki S, Murai J. Mitigating bitcoin node storage size by DHT. In: Proc. of the 14th Asian Internet Engineering Conf. Bangkok: ACM, 2018. 17–23. [doi: 10.1145/3289166.3289169]

[47] Awerbuch B, Scheideler C. Towards a scalable and robust DHT. In: Proc. of the 8th Annual ACM Symp. on Parallelism in Algorithms

and Architectures. Cambridge: ACM, 2006. 318–327. [doi: 10.1145/1148109.1148163]

[48] Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Dabek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. on Networking, 2003, 11(1): 17–32. [doi: 10.1109/TNET.2002.808407]

[49] Xu ZY, Han SY, Chen L. CUB, a consensus unit-based storage scheme for blockchain system. In: Proc. of the 34th Int'l Conf. on Data Engineering (ICDE). Paris: IEEE, 2018. 173–184. [doi: 10.1109/ICDE.2018.00025]

[50] Dai XH, Xiao J, Yang WH, Wang CF, Jin H. Jidar: A jigsaw-like data reduction approach without trust assumptions for bitcoin system. In: Proc. of the 39th Int'l Conf. on Distributed Computing Systems (ICDCS). Dallas: IEEE, 2019. 1317–1326. [doi: 10.1109/ICDCS.2019.00132]

[51] Yang WH, Dai XH, Xiao J, Jin H. LDV: A lightweight DAG-based blockchain for vehicular social networks. IEEE Trans. on Vehicular Technology, 2020, 69(6): 5749–5759. [doi: 10.1109/TVT.2020.2963906]

[52] Zhang GC, Wang RJ. Blockchain shard storage model based on threshold secret sharing. Journal of Computer Applications, 2019, 39(9): 2617–2622 (in Chinese with English abstract). [doi: 10.11772/j.issn.1001-9081.2019030406]

[53] Raman RK, Varshney LR. Coding for scalable blockchains via dynamic distributed storage. IEEE/ACM Trans. on Networking, 2021, 29(6): 2588–2601. [doi: 10.1109/TNET.2021.3098613]

[54] Weatherspoon H, Kubiatowicz JD. Erasure coding vs. replication: A quantitative comparison. In: Proc. of the 1st Int'l Workshop on Peer-to-peer Systems. Cambridge: Springer, 2002. 328–337. [doi: 10.1007/3-540-45748-8_31]

[55] Qi XD, Zhang Z, Jin CQ, Zhou AY. BFT-store: Storage partition for permissioned blockchain via erasure coding. In: Proc of the 36th Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 1926–1929. [doi: 10.1109/ICDE48307.2020.00205]

[56] Du ZY, Pang XT, Qian HF. PartitionChain: A scalable and reliable data storage strategy for permissioned blockchain. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(4): 4124–4136. [doi: 10.1109/TKDE.2021.3136556]

[57] Hong ZC, Guo S, Zhou EY, Chen WH, Huang HW, Zomaya A. GriDB: Scaling blockchain database via sharding and off-chain cross-shard mechanism. Proc. of the VLDB Endowment, 2023, 16(7): 1685–1698. [doi: 10.14778/3587136.3587143]

[58] Li CL, Zhang J, Yang XM, Youlong L. Lightweight blockchain consensus mechanism and storage optimization for resource-constrained IoT devices. Information Processing & Management, 2021, 58(4): 102602. [doi: 10.1016/j.ipm.2021.102602]

[59] Zhang PY, Liu ZJ, Zhou MC. A novel block storage model for consortium blockchains. In: Proc. of the 2022 IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC). Prague: IEEE, 2022. 1437–1442. [doi: 10.1109/SMC53654.2022.9945312]

[60] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002, 6(2): 182–197. [doi: 10.1109/4235.996017]

[61] Priya R, Ramesh D, Udutalapally V. NSGA-2 optimized fuzzy inference system for crop plantation correctness index identification. IEEE Trans. on Sustainable Computing, 2022, 7(1): 172–188. [doi: 10.1109/TSUSC.2021.3064417]

[62] Han RC, Yu JS, Zhang R. Analysing and improving shard allocation protocols for sharded blockchains. In: Proc. of the 4th ACM Conf. on Advances in Financial Technologies. Cambridge: ACM, 2022. 198–216. [doi: 10.1145/3558535.3559783]

[63] Fynn E, Pedone F. Challenges and pitfalls of partitioning blockchains. In: Proc. of the 48th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks Workshops (DSN-W). Luxembourg: IEEE, 2018. 128–133. [doi: 10.1109/DSN-W.2018.00051]

[64] Bertino E, Atzeni P, Tan KL, Chen Y, Tay YC, Curino C, Jones E, Zhang Y, Madden S. Schism: A workload-driven approach to database replication and partitioning. Proc. of the VLDB Endowment, 2010, 3(1–2): 48–57. [doi: 10.14778/1920841.1920853]

[65] Al-Bassam M, Sonnino A, Bano S, Hrycyszyn D, Danezis G. Chainspace: A sharded smart contracts platform. In: Proc. of the 25th Annual Network and Distributed System Security Symp. San Diego: NDSS, 2018. 1–15. [doi: 10.14722/ndss.2018.23241]

[66] Nguyen LN, Nguyen TDT, Dinh TN, Thai MT. OptChain: Optimal transactions placement for scalable blockchain sharding. In: Proc. of the 39th Int'l Conf. on Distributed Computing Systems (ICDCS). Dallas: IEEE, 2019. 525–535. [doi: 10.1109/ICDCS.2019.00059]

[67] Król M, Ascigil O, Rene S, Sonnino A, Al-Bassam M, Rivière E. Shard Scheduler: Object placement and migration in sharded account-based blockchains. In: Proc. of the 3rd ACM Conf. on Advances in Financial Technologies. Arlington: ACM, 2021. 43–56. [doi: 10.1145/3479722.3480989]

[68] Huang HW, Peng XW, Zhan JZ, Zhang SY, Lin Y, Zheng ZB, Guo S. BrokerChain: A cross-shard blockchain protocol for account/balance-based state sharding. In: Proc. of the 2022 IEEE Conf. on Computer Communications. London: IEEE, 2022. 1968–1977. [doi: 10.1109/INFOCOM48880.2022.9796859]

[69] Karypis G, Kumar V. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Minneapolis: University of Minnesota, 1997.

[70] Zhang YZ, Pan SR, Yu JS. TxAllo: Dynamic transaction allocation in sharded blockchain systems. In: Proc of the 39th Int'l Conf. on Data Engineering (ICDE). Anaheim: IEEE, 2023. 721–733. [doi: 10.1109/ICDE55515.2023.00390]

[71] Jin D, Yu ZZ, Jiao PF, Pan SR, He DX, Wu J, Yu PS, Zhang WX. A survey of community detection approaches: From statistical modeling to deep learning. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(2): 1149–1170. [doi: 10.1109/TKDE.2021.3104155]
[72] Lancichinetti A, Fortunato S. Community detection algorithms: A comparative analysis. Physical Review E, 2009, 80(5): 056117. [doi: 10.1103/PhysRevE.80.056117]
[73] Li MZ, Wang W, Zhang J. LB-Chain: Load-balanced and low-latency blockchain sharding via account migration. IEEE Trans. on Parallel and Distributed Systems, 2023, 34(10): 2797–2810. [doi: 10.1109/TPDS.2023.3238343]
[74] Dayan N, Weiss T, Dashevsky S, Pan M, Bortnikov E. Spooky: Granulating LSM-tree compactions correctly. Proc. of the VLDB Endowment, 2022, 15(11): 3071–3084. [doi: 10.14778/3551793.355185]
[75] Li JH, Ren YJ, Han SJ, Lee PPC. Enhancing LSM-tree key-value stores for read-modify-writes via key-delta separation. In: Proc. of the 40th Int'l Conf. on Data Engineering (ICDE). Utrecht: IEEE, 2024. 4938–4950. [doi: 10.1109/ICDE60146.2024.00375]
[76] Qi XD. S-Store: A scalable data store towards permissioned blockchain sharding. In: Proc. of the 2022 IEEE Conf. on Computer Communications. London: IEEE, 2022. 1978–1987. [doi: 10.1109/INFOCOM48880.2022.9796800]
[77] Protocol Labs. Filecoin: A decentralized storage network. 2017. https://filecoin.io/filecoin.pdf
[78] Storj Labs. Storj: A decentralized cloud storage network framework. 2018. https://static.storj.io/storjv3.pdf
[79] Benet J. IPFS-content addressed, versioned, P2P file system. arXiv:1407.3561, 2014.
[80] Hellings J, Sadoghi M. ByShard: Sharding in a byzantine environment. The VLDB Journal, 2023, 32(6): 1343–1367. [doi: 10.1007/s00778-023-00794-0]
[81] Hong ZC, Guo S, Li P, Chen WH. Pyramid: A layered sharding blockchain system. In: Proc. of the 2021 IEEE Conf. on Computer Communications. Vancouver: IEEE, 2021. 1–10. [doi: 10.1109/INFOCOM42981.2021.9488747]
[82] Ren K, Li D, Abadi DJ. SLOG: Serializable, low-latency, geo-replicated transactions. Proc. of the VLDB Endowment, 2019, 12(11): 1747–1761. [doi: 10.14778/3342263.3342647]
[83] Thomson A, Diamond T, Weng SC, Ren K, Shao P, Abadi DJ. Calvin: Fast distributed transactions for partitioned database systems. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. Scottsdale: ACM, 2012. 1–12. [doi: 10.1145/2213836.2213838]
[84] Abadi DJ, Faleiro JM. An overview of deterministic database systems. Communications of the ACM, 2018, 61(9): 78–88. [doi: 10.1145/3181853]
[85] Zheng PL, Xu QQ, Zheng ZB, Zhou ZY, Yan Y, Zhang H. Meepo: Sharded consortium blockchain. In: Proc. of the 37th Int'l Conf. on Data Engineering (ICDE). Chania: IEEE, 2021. 1847–1852. [doi: 10.1109/ICDE51399.2021.00165]
[86] Que QF, Chen ZH, Zhang Z, Yang YQ, Zhou AY. A coordinator-free cross-shard transaction execution for sharded permissioned blockchains. Journal of Computer Research and Development, 2023, 60(11): 2469–2488 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.202330294]
[87] Hong ZC, Guo S, Zhou EY, Zhang JT, Chen WH, Liang JW, Zhang J, Zomaya A. Prophet: Conflict-free sharding blockchain via Byzantine-tolerant deterministic ordering. In: Proc. of the 2023 IEEE Conf. on Computer Communications. New York: IEEE, 2023. 1–10. [doi: 10.1109/INFOCOM53939.2023.10228939]
[88] Rahnama S, Gupta S, Sogani R, Krishnan D, Sadoghi M. RingBFT: Resilient consensus over sharded ring topology. In: Proc. of the 25th Int'l Conf. on Extending Database Technology. Edinburgh: EDBT, 2022. 298–311.
[89] Hyperchain. BitXHub whitepaper v2.0. 2022. (in Chinese) https://upload.hyperchain.cn/BitXHub%E7%99%BD%E7%9A%AE%E4%B9%A6.pdf

附中文参考文献:

[1] 董昊文, 张超, 李国良, 冯建华. 云原生数据库综述. 软件学报, 2024, 35(2): 899–926. http://www.jos.org.cn/1000-9825/6952.htm [doi: 10.13328/j.cnki.jos.006952]
[5] 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, 2018, 41(5): 969–988. [doi: 10.11897/SP.J.1016.2018.00969]
[6] 邵奇峰, 张召, 朱燕超, 周傲英. 企业级区块链技术综述. 软件学报, 2019, 30(9): 2571–2592. http://www.jos.org.cn/1000-9825/5775.htm [doi: 10.13328/j.cnki.jos.005775]
[23] 佟兴, 张召, 金澈清, 周傲英. 面向端边云协同架构的区块链技术综述. 计算机学报, 2021, 44(12): 2345–2366. [doi: 10.11897/SP.J.1016.2021.02345]
[25] 黄华威, 孔伟, 彭肖文, 郑子彬. 区块链分片技术综述. 计算机工程, 2022, 48(6): 1–10. [doi: 10.19678/j.issn.1000-3428.0063887]
[26] 谭朋柳, 徐滕, 涂若欣. 区块链分片技术研究综述. 计算机科学, 2024, 51(11): 307–320.

[52] 张国潮, 王瑞锦. 基于门限秘密共享的区块链分片存储模型. 计算机应用, 2019, 39(9): 2617–2622. [doi: 10.11772/j.issn.1001-9081.2019030406]
[86] 阙琦峰, 陈之豪, 张召, 杨艳琴, 周傲英. 面向分片许可链的无协调者跨片交易处理. 计算机研究与发展, 2023, 60(11): 2469–2488. [doi: 10.7544/issn1000-1239.202330294]
[89] 趣链科技. BitXHub 白皮书 v2.0. 2022. https://upload.hyperchain.cn/BitXHub%E7%99%BD%E7%9A%AE%E4%B9%A6.pdf


唐海波(1993－), 男, 博士生, CCF 学生会员, 主要研究领域为分布式数据库, 区块链, Web 数据挖掘.


金澈清(1977－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为基于位置的服务, 数据质量, 不确定性数据管理, 区块链.


张焕(2000－), 男, 硕士生, 主要研究领域为分布式数据库, 区块链.


周傲英(1965－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为 Web 数据管理, 数据密集型计算, 内存集群计算, 分布式事务处理, 大数据基准测试和性能优化.


张召(1977－), 女, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为分布式数据库, 区块链.