软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(1):139−156 [doi: 10.13328/j.cnki.jos.007480] [CSTR: 32375.14.jos.007480]

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

区块链状态分片技术综述*

苏琳萱, 张 晓, 于锦阳, 何赫烺, 王锦江, 黄志杰

(西北工业大学 计算机学院, 陕西 西安 710072)

通信作者: 张晓, E-mail: zhangxiao@nwpu.edu.cn; 黄志杰, E-mail: jayzy.huang@nwpu.edu.cn

摘 要: 区块链通过全副本机制和共识协议, 在不依赖可信第三方机构的情况下, 保证了数据的安全性与一致性. 然而, 这种设计也显著限制了系统的可扩展性. 分片技术通过将区块链系统划分为多个并行工作的分片, 可有效解决上述问题. 其中, 状态分片在各分片并行的前提下, 进一步将状态数据分散存储于各分片中, 显著提高系统吞吐量与存储效率. 目前, 状态分片面临以下问题: (1) 跨分片交易涉及多个分片的协同处理, 引入额外的跨分片通信和共识开销, 需要提高其处理效率; (2) 需要合理划分状态, 降低系统跨分片交易比例并平衡各分片的工作负载; (3) 动态划分和调整状态时, 需要在不同分片间高效迁移状态数据. 系统地梳理状态分片技术的关键问题和主流实现方法, 从跨分片交易处理、状态分配和状态迁移这 3 个方面对比和分析现有成果的优缺点. 最后, 指出状态分片技术面临的问题, 为未来的研究工作提供方向.

关键词: 区块链分片技术; 状态分片; 跨分片交易; 可扩展性

中图法分类号: TP311

中文引用格式: 苏琳萱, 张晓, 于锦阳, 何赫烺, 王锦江, 黄志杰. 区块链状态分片技术综述. 软件学报, 2026, 37(1): 139–156. http://www.jos.org.cn/1000-9825/7480.htm

英文引用格式: Su LX, Zhang X, Yu JY, He HL, Wang JJ, Huang ZJ. Survey on Blockchain State Sharding Technology. Ruan Jian Xue Bao/Journal of Software, 2026, 37(1): 139–156 (in Chinese). http://www.jos.org.cn/1000-9825/7480.htm

Survey on Blockchain State Sharding Technology

SU Lin-Xuan, ZHANG Xiao, YU Jin-Yang, HE He-Lang, WANG Jin-Jiang, HUANG Zhi-Jie

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: Blockchain ensures data security and consistency without relying on trusted third-party institutions through full replication mechanisms and consensus protocols. However, this design significantly limits system scalability. Sharding addresses this limitation by partitioning the blockchain into multiple parallel working shards, with state sharding further distributing state data across these shards, significantly improving system throughput and storage efficiency. Currently, state sharding faces several challenges, including the following three problems: (1) Cross-shard transactions require collaborative processing across multiple shards, introducing additional communication and consensus overhead, necessitating improvements in processing efficiency; (2) State partitioning optimization is needed to reduce the proportion of cross-shard transactions and balance shard workloads; (3) Efficient state data migration between shards is required during dynamic state partitioning and adjustment. This study systematically reviews the key challenges and mainstream implementations of state sharding, focusing on cross-shard transaction processing, state allocation, and state migration. Finally, the study identifies current limitations of state sharding and suggests possible directions for future research.

Key words: blockchain sharding technology; state sharding; cross-shard transaction; scalability

区块链是一种去中心化的分布式账本技术, 通过密码学和共识算法等技术确保了数据的透明性、安全性以及不可篡改性等特点, 能够有效支持点对点交易与信息共享, 实现了去信任化的价值转移[1,2]. 近年来, 区块链技术从

* 基金项目: 国家重点研发计划 (2022YFB2702101); 国家自然科学基金重大集成项目 (92152301); 国家自然科学基金面上项目 (62272394)
收稿时间: 2024-11-29; 修改时间: 2025-02-25, 2025-05-02; 采用时间: 2025-05-29; jos 在线出版时间: 2025-10-29
CNKI 网络首发时间: 2025-10-31

最初的加密货币应用[3,4]逐渐扩展至供应链管理[5]、医疗健康[6]以及物联网[7]等多个领域. 然而, 传统的区块链系统要求每个节点都需要验证所有交易并存储完整的区块链数据, 导致区块链节点面临着严重的计算和存储负担, 极大限制了系统的可扩展性[8,9]. 为解决这一问题, Luu 等人[10]在 2016 年提出了 Elastico 分片协议. 随后分片技术被广泛应用于以太坊 2.0[11]、Zilliqa[12]、QuarkChain[13]和 Harmony[14]等多个项目中, 这些实际应用进一步验证了该技术在提升区块链系统可扩展性方面的前景与优势.

区块链分片技术包括网络分片、交易分片和状态分片这 3 个层次[15]. 通常, 将实现网络分片和交易分片的区块链称为部分分片系统, 同时实现网络、交易以及状态分片的区块链称为全分片系统[16]. 网络分片是区块链分片系统的基础, 它将网络节点划分为多个不相交的小型分片, 同一分片内的节点将维护相同的状态数据并就一组交易达成共识. 交易分片中每个节点仍保存完整的状态数据, 因此每个分片可独立验证任意交易, 交易则是被划分为不相交的子集, 并分配到不同的分片中进行处理, 从而实现各分片的交易处理并行化, 理论上可实现系统吞吐量随着分片数量线性扩展. 状态分片技术则是进一步将区块链系统的全局状态进行划分, 每个分片只需存储部分状态数据, 从而降低了节点的存储开销, 提高了系统的存储效率, 理想情况下状态分片可实现存储效率和系统吞吐量的线性扩展. 然而, 由于每个分片仅存储部分状态数据, 因此每个分片只能处理那些与其维护状态相关的部分交易, 这在一定程度上为系统引入了跨分片交易和负载不均衡等问题[15]. 状态分片技术涉及多个复杂的理论和技术问题, 包括状态分配、状态迁移以及跨分片交易处理等. 由于状态分片技术可有效解决区块链系统可扩展性问题, 该技术近年来引起了广泛的关注与研究.

本文将围绕区块链状态分片的关键问题与主流解决方法, 从状态分配、状态迁移以及跨分片交易处理这 3 个方面 (如图 1 所示), 对该技术的最新研究成果进行分析与综述, 论述已有解决方案的优势和不足, 并探讨该技术未来的发展方向. 本文旨在为状态分片技术的研究提供有价值的参考和帮助.

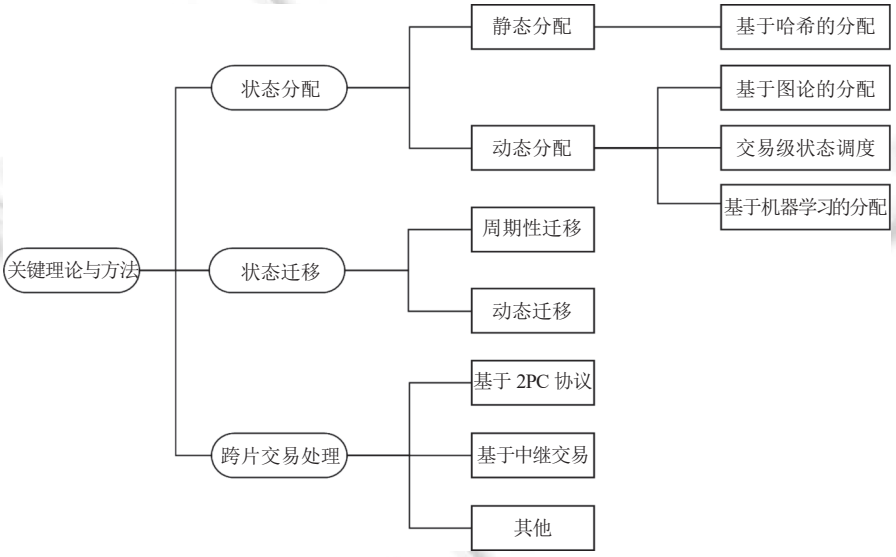

图 1 区块链状态分片技术关键理论与方法

1 背 景

1.1 区块链及其可扩展性问题

区块链本质上是一个分布式账本系统, 通过共识算法来维护数据的一致性. 在该系统中, 每个节点都保存一份完整的区块链账本副本, 并利用密码学等技术保证数据的不可篡改性, 从而在无需中心化机构的情况下, 实现了点对点的可信交易. 然而, 传统区块链系统的可扩展性严重受限: 随着节点数量的增多以及交易规模的扩大, 区块链

系统的吞吐量 (TPS) 以及存储效率无法得到有效提升. 具体表现如下.

(1) 系统吞吐量: 比特币和以太坊的吞吐量分别约为 7 和 20 TPS[17,18], 而目前的大型中心化交易处理系统的吞吐量可达上千 TPS[19].

(2) 系统存储效率: 全副本存储机制导致区块链系统存储效率极低, 系统有效数据量与总存储量之比为 1/N, 其中 N 为节点数量. 截至 2024 年 9 月 22 日, 比特币数据规模已达到 419.50 GB[20], 网络节点数量已超过 10 000 个[21], 这意味着系统需要消耗 TB 级存储空间来维护仅约 400 GB 的有效数据.

区块链可扩展性问题的根源在于其固有的机制设计. 传统区块链作为全复制系统, 要求每个节点都存储完整的区块链数据, 并参与每笔交易的验证和共识过程. 随着交易量的增长, 系统的存储开销急剧增加. 其次, 网络规模的扩大也显著增加了系统通信开销和共识复杂度, 限制了系统吞吐量.

为了提升区块链系统的可扩展性, 近年来研究人员提出了多种解决方案, 主要包括 Layer1 协议层的改进和 Layer2 的扩展技术[22]. Layer1 解决方案通过直接优化区块链的基础协议来提升系统性能, 例如优化共识算法[23,24]、增加区块大小、缩短出块时间等[25], 代表技术包括分片技术[26]和 DAG 分布式账本[27,28]. Layer2 解决方案则是在不改变主链的前提下, 将区块链部分计算和存储压力转移至链下或侧链上[29], 以提高系统可扩展性, 典型的例子包括比特币的闪电网络[30]和以太坊的 Rollups[31]. 其中, 区块链分片技术通过将交易和状态数据划分至不同分片并行处理和维护, 有效解决了区块链扩展性瓶颈. 该技术显著降低了系统的通信、计算和存储开销, 使系统在大规模网络中依然保持高效运行, 因此被视为最具前景的区块链扩容方案之一.

1.2 区块链分片技术

区块链分片技术是提升区块链系统可扩展性的重要手段. 早期的分片系统主要采用部分分片方案, 即仅对网络和交易进行分片, 但节点仍需存储完整的全局状态数据. 这种方案通过交易并行处理提升了系统吞吐量, 且由于所有节点存储完整状态, 交易验证简单, 系统协议设计也较为直观 (如图 2 所示).

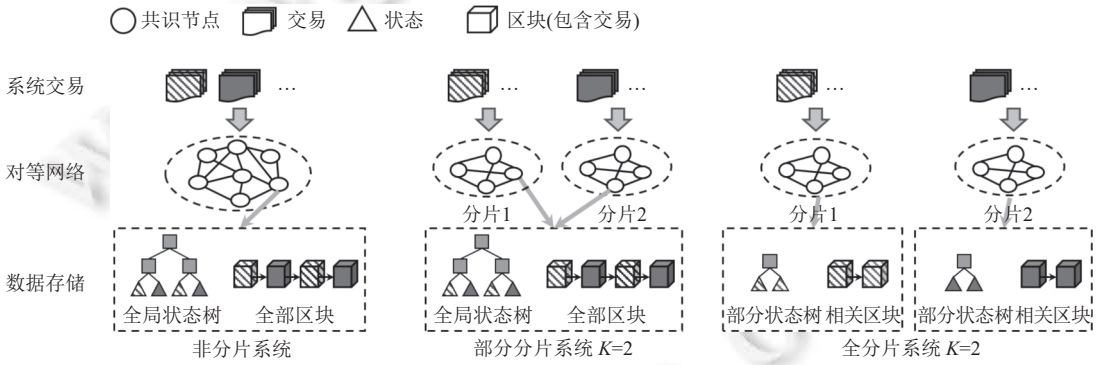

图 2 传统区块链与区块链分片系统的工作机制对比

然而, 部分分片系统存在着存储效率低下等关键问题. 随着系统吞吐量的提升, 区块链系统的数据规模急剧增加, 单个节点存储的全局状态数据规模不断上升, 导致了以下问题: (1) 全节点的存储门槛提高, 普通节点难以作为全节点加入, 影响系统的去中心化性; (2) 系统的整体存储效率未得到优化, 无法充分发挥分片架构的优势. 为解决上述问题, 大量研究提出了全分片系统, 在部分分片系统的基础上引入了状态分片技术, 其中每个分片只需存储部分状态数据. 通过这种设计, 单个节点的存储负担得到有效减轻, 存储效率得以提升, 同时系统的去中心化特性得以保持. 因此, 当前区块链分片技术的研究大多基于全分片系统.

1.2.1 全分片系统的工作机制

在区块链全分片系统中, 为了确保系统安全性和分片成员的随机性, 防止恶意节点集中在单个分片从而对其进行长期操控, 系统需要对分片进行定期重组. 因此, 系统的运行通常按纪元 (epoch) 为周期展开, 每个纪元包括配置和共识两个阶段, 具体可分为以下 6 个关键步骤[26].

(1) 建立身份: 该步骤针对无许可区块链设计. 无许可区块链中的节点缺乏可信的身份认证, 因此节点在加入区块链分片系统前需要建立身份 (通常采用 PoW 机制建立身份), 以防范潜在的女巫攻击[32]. 相比之下, 许可型区块链通常不需要执行该步骤.

(2) 组建分片: 系统根据预定的规则将已建立身份的共识节点分配至不同的分片, 并构建相应的分片网络. 为确保各分片内部的节点能够相互识别, 系统会采用特定协议, 使得同一分片中的所有节点能够相互通信并识别彼此的身份信息.

(3) 状态分配: 系统按照预定的分配规则将全局状态划分至不同分片, 该步骤通常在配置阶段完成. 此外, 为更好适应系统的实时负载, 部分方案[33,34]可能在共识阶段对状态进行动态调整.

(4) 状态迁移: 在动态划分和调整状态后, 相关状态需要在不同的分片中进行迁移. 在此过程中, 必须确保数据的一致性和完整性.

(5) 分片内共识: 在每个分片内, 所有节点需要就一组交易达成一致. 为此, 所有诚实节点必须在其所属分片内对提议的最新区块达成共识.

(6) 跨分片交易处理: 对于跨分片交易, 各相关分片必须协调一致, 以确保交易的原子提交. 这一过程通常需要通过跨分片通信来实现, 以确保不同分片间的交易能够正确处理.

身份建立、分片组建、状态分配以及状态迁移通常在配置阶段执行 (在动态适应性较强的方案中, 状态分配和状态迁移可能在共识阶段执行), 分片内共识和跨分片交易处理则在共识阶段执行.

1.2.2 区块链状态分片技术

在区块链系统中, “状态”指的是系统中所有数据在某一时刻的全局快照, 随着每笔交易的执行而变化, 反映了每个参与者在当前时刻的最新状态. 目前区块链系统中主要通过 UTXO (unspent transaction output) 和账户/余额模型两种方式保存状态. 比特币采用 UTXO[3]模型保存状态, 系统中每笔交易的输出被视为未花费的输出, 而每个输入都会引用先前未花费的输出, 这些引用的输出被标记为已花费, 一笔交易可能涉及多个输入和输出. 以太坊则采用账户/余额模型[4], 系统直接存储每个账户的状态信息, 包括账户地址、余额及智能合约代码等[35]. 用户可通过账户余额发起交易, 每笔交易包括一个发送方和一个接收方.

区块链状态分片技术的基本原理是将区块链的全局状态数据进行划分并分散存储在各个分片中, 各分片能够独立处理分片内交易, 并协同完成跨分片交易处理, 每个节点只需维护其所在分片的账本状态和相关区块数据 (如图 2 所示). 假设全局数据规模为 B, 在未分片系统和部分分片系统中, 节点需要存储完整数据, 系统存储效率为 $1/N$. 相比之下, 在全分片系统中, 假设分片数量为 K, 理论上节点仅需存储 B/K 的数据, 系统存储效率提升至 K/N, 节点存储压力降低为原来的 $1/K$. 状态分片技术主要包括状态分配、状态迁移和跨分片交易处理这 3 个关键步骤. 需要指出的是, 节点建立身份、分片组建及分片内共识这 3 个步骤与状态分片技术相互独立, 在本文中不作详细讨论.

2 状态分片技术的关键问题与方法

2.1 状态分配

状态分配将全局状态数据映射到不同的分片中维护, 在设计过程中需要考虑多个优化目标, 特别是如何降低系统跨分片交易比例及实现分片负载均衡. 我们首先对状态分配问题给出形式化定义, 并明确其中涉及的符号及其意义.

假设全分片系统包含 K 个分片, 记为集合 $S=\{s_1,s_2,\ldots,s_K\}$. 状态数据集合定义为 $D=\{d_1,d_2,\ldots,d_n\}$, n 表示系统中所有状态数据的数量. 交易集合定义为 $T=\{t_1,t_2,\ldots,t_m\}$, m 表示系统中的交易数量. 每个交易 t_i 涉及一个或多个状态数据, 即 t_i=(d_{from}, d_{to}), 通常交易会被分配至其发送状态所在分片中进行处理. 在 UTXO 模型中, 一笔交易可能会涉及多个输入和输出状态, 此时, d_{from}、d_{to} 可被表示为状态数据的集合.

状态的分配过程可以被定义为一个二元映射函数 $f: D\times S\to\{0,1\}$. $f(d_i,s_j)=1$ 表示状态数据 d_i 被分配到分片 s_j; $f(d_i,s_j)=0$ 表示状态数据 d_i 未被分配到分片 s_j. 状态分配需满足以下约束条件.

首先, 每个状态 d_i 需要被分配到一个分片 s_j, 即:

$$\sum_{j=1}^{K} f(d_i, s_j) = 1, \ \forall d_i \in D \tag{1}$$

其次, 系统中所有分片的状态数据集合的并集必须覆盖整个状态数据集合 D, 即:

$$\bigcup_{j=1}^{K} \{d_i | f(d_i, s_j) = 1\} = D \tag{2}$$

在上述约束条件下, 状态的分配包含以下两个优化目标.

● 降低跨分片交易比例: 当一笔交易的相关状态数据存储在不同的分片时, 该笔交易即为跨分片交易. 定义系统的跨分片交易的数量为 c, 即:

$$c = \sum_{t_i \in T} I\{f(d_{\text{from}}, s_i) = 1 \wedge f(d_{\text{to}}, s_j) = 1 \wedge i \neq j\} \tag{3}$$

其中, I 为指示函数, 当条件为真时返回 1. 目标是尽可能降低系统中的跨分片交易比例 c/m.

● 平衡分片工作负载: 定义分片 s_i 的工作负载 L_i 为其处理的交易数量, $\bar{L}$ 为系统的负载均值. 使用方差来衡量分片负载均衡程度, 即:

$$\frac{1}{K}\sum_{i=1}^{K}(L_i - \bar{L})^2 \tag{4}$$

设计目标为尽可能平衡各分片负载, 降低系统的负载均衡方差.

综上所述, 状态分配方法的设计目标是在满足约束条件, 即公式 (1) 和公式 (2) 的前提下, 设计一种分配方案, 能够最小化跨分片交易数量, 同时平衡各分片的工作负载.

根据状态分配的执行频率, 可将其分为静态分配和动态分配两种类型. 静态分配仅执行一次, 后续不再进行状态调整; 而动态分配则允许通过调整各分片中的维护的状态来优化分配策略, 以适应环境的变化. 静态分配通常采用哈希函数实现, 只需对输入数据 (例如账户地址) 进行哈希运算即可生成唯一的分片标识符, 计算复杂度通常为 O(1). 然而, 由于该方法缺乏灵活性, 忽略了状态间的关联与交易特征, 在降低系统跨分片交易比例和实现分片负载均衡这两个优化目标上表现不佳, 具体表现如下.

(1) 跨分片交易极高: 由于哈希函数的随机性, 静态分配方法可以视为均匀随机选择, 一笔交易跨分片的概率可以表示为 $1 - P(u, v, k)$, $v = 1$ [36,37]. 其中, u 表示交易的输入输出数量之和, v 表示需要参与验证该笔交易的分片数量, k 为系统中的分片数量. 当 v=1 时,即表示该交易的所有输入输出均位于同一分片内，此时 $1 - P(u, 1, k)$ 即可计算出该交易是跨分片交易的概率. $P(u, v, k)$ 的计算公式如下:

$$P(u,v,k) = \begin{cases} 1, & \text{if } u = v = 1 \\ \left(\dfrac{1}{k}\right)^{u-1}, & \text{if } v = 1 \\ \dfrac{k-v}{k} \cdot P(u-1, v-1, k), & \text{if } u = v \\ \dfrac{k-v}{k} \cdot P(u-1, v-1, k) + \dfrac{v}{k} \cdot P(u-1, v, k), & \text{otherwise} \end{cases} \tag{5}$$

在基于账户模型的系统中, 交易跨分片概率为 $P(k) = 1 - (1/k)$ (即 u=2). 当 $k = u = 2$ 时, 一笔交易跨分片的概率为 75%. 当 $k = 3$ 时, 交易跨分片的概率将迅速超过 95%. 随着分片数量的增加, 系统中几乎所有的交易均为跨分片交易 [10,36–39]. 与普通的片内交易相比, 跨分片交易的处理复杂性显著增加. 片内交易仅需在单个分片内完成处理, 而跨分片交易的处理需要多个相关分片间的协同处理, 这一过程不仅增加了分片间的通信频率, 还引入了额外的同步机制, 从而显著增加了计算开销和通信开销. 因此, 高比例的跨分片交易将严重限制全分片系统性能的进一步扩展.

(2) 分片负载不均: 该问题是指各分片处理的交易数量存在显著差异, 从而导致部分少数分片面临交易量过度集中的“热分片”现象 [15]. 热分片的出现不仅会增加系统平均交易确认延迟, 还会降低系统吞吐量. 这一问题在采用

账户模型的状态实现方式中尤为突出, 主要原因是现实中少数热门账户往往会发起系统中大部分交易, 当热门账户集中于同一分片时, 会导致分片的工作负载严重不均. 我们基于以太坊数据采用静态分配方法, 进行多轮次的交易分配以评估分片的负载分布. 图 3 展示了在 8 分片下, 每轮注入 50 000 条以太坊交易的结果. 该方法会在分片之间产生极度不均衡的交易分布.

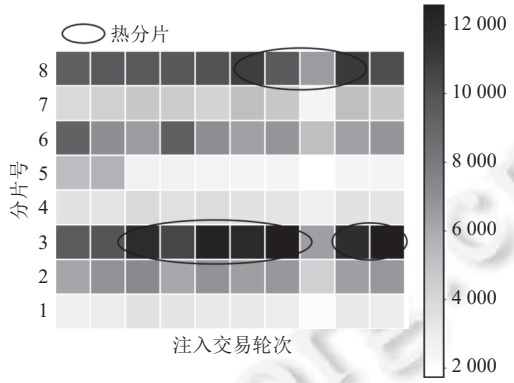

图 3 哈希分配下分片工作负载分布

为解决上述问题, 大量研究提出了动态分配方法. 相较于静态分配方法, 该方法通过分析状态之间的关联性和交易模式, 定期调整状态分配策略, 能够更加灵活地应对交易模式的变化, 有效降低系统的跨分片交易比例, 并且优化分片负载均衡, 进一步提升系统的整体性能. 动态分配的具体实现包括基于图论的分配方法、基于机器学习的分配方法以及交易级的状态调度方法. 表 1 对状态分配的各类方法的优缺点进行了总结. 此外, 动态分配过程中涉及状态存储位置的调整, 需要状态管理机制来实现状态数据的高效管理与定位. 因此, 本节还将介绍动态分配环境下的主流状态数据管理方法.

表 1 状态分配方法总结对比

分类	状态特征分析	动态适应性	优点	缺点
哈希分配	否	差	实现简单且分配高效	分配固定, 缺乏灵活性
图分配	是	中等	基于历史数据识别状态特征, 实施图划分	周期性分配, 无法应对系统突然波动
状态调度	是	强	实时调度状态, 应对系统波动能力强	存在一定的调度决策计算开销
机器学习	是	强	可预测未来交易行为, 分配准确率高	训练成本高, 算法效果依赖输入数据的质量

2.1.1 基于图论的分配方法

一些研究[15,40–45]将状态分配转化为图模型上的子图划分问题[46–48]. 在划分过程中, 通过减少不同子图之间的边数量和平衡子图权重, 实现降低系统跨分片交易比例以及平衡分片工作负载的目的. 基于图论的分配方法通常在分片配置时期执行, 系统将使用上一纪元的历史交易构建区块链状态关系图, 执行相应的子图划分算法, 从而为下一纪元生成状态分配策略. 区块链状态关系图构建方法主要包括以下两种.

(1) 账户为节点 (如图 4 所示): 该构建方法适用于采用账户模型的区块链全分片系统. 具体来说, 系统利用收集到的历史交易数据, 构建无向状态关系图 $G(V, E)$. 在该图中, V 表示账户 (节点) 集合, E 表示交易 (边) 集合. 当两个账户 v_i 和 v_j 之间存在交易时, 它们之间就会有一条对应的边 e_{ij}. 边权重 $w(e_{ij})$ 表示两个账户之间的交易数量. 节点权重 $w(v_i)$ 表示与该账户相关的交易总数. 基于状态关系图 $G(V, E)$, 系统执行子图划分算法进行状态分配.

BrokerChain[15]和 X-Shard[45]采用 METIS[46]算法将图 G 划分为 K 个不重叠的子图, 每一个子图对应一个分片. CLPA[42]、TxAllo[43]和 Estuary[44]则使用了社区检测算法[49,50], 通过识别图 G 中联系密切的节点, 将其作为社区划分子图. 这种方法确保了频繁交互的账户能够被聚集在同一分片中, 从而有效地减少跨分片交易. 此外, 图分配方法在划分过程中通过限制各子图的节点权重 $w(v_i)$ 之和, 以平衡分片间工作负载. 以 BrokerChain 为例, 在系统中

32 个分片的情况下, 可将跨分片交易比例从采用哈希分配方法时的 96% 降低至约 65%, 同时显著降低了分片间的工作负载方差.

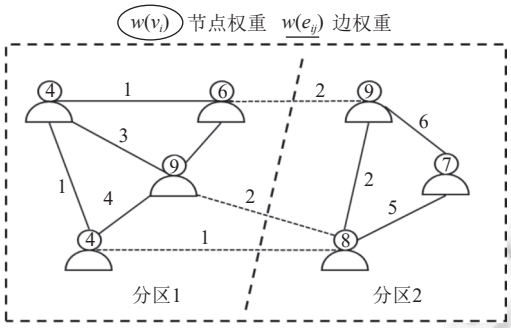

图 4 账户为节点的状态图

(2) 交易为节点 (如图 5 所示): 该构建方法适用于采用 UTXO 模型的区块链全分片系统. OptChain[40]基于 UTXO 模型提出一种以交易作为节点的状态关系图构建方法. 具体来说, 系统依然使用历史交易数据进行状态关系图 $G(V, E)$ 构建. 不同的是, $G(V, E)$ 被构建为有向图, 其中, V 是交易的集合, E 是有向边的集合. 如果交易 u 使用了交易 v 的 UTXO, 则在 E 中将存在一条从 v 到 u 的有向边 (u, v). 当图中的节点不存在出边时, 表示它们是 Coinbase 交易 (即矿工获得的区块奖励), 而没有入边的节点则表示这些交易的 UTXO 尚未被花费. 由于每笔交易只使用过去交易的 UTXO, 该图始终为有向无环图, 节点可以按照交易发生的顺序进行拓扑排序. 在完成状态关系图构建后, OptChain 引入一种时间适应评分的评估机制, 来评估将新的交易放到每个分片的适合程度, 包括衡量该交易放置到分片中而不引发跨分片交易的概率的 T2S (transaction-to-shard) 评分和评估交易放置到分片中的处理延迟的 L2S (latency-to-shard) 评分. 该方案在 32 分片下, 将系统跨分片交易比例从采用哈希分配方法下的 99% 降低至 19% 左右. 在负载平衡方面, OptChain 的最大交易队列大小约为 44 000 笔交易, 相较于 OmniLedger 的 499 000 个交易, 其表现出显著优越性, 降低了约 90% 的队列拥堵情况.

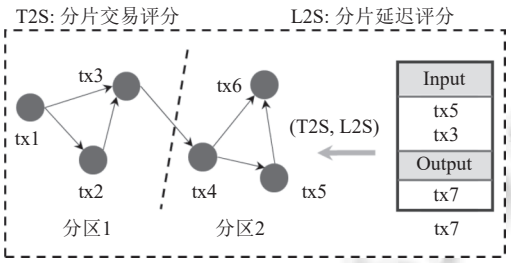

图 5 交易为节点的状态图

图分配方法基于由历史交易数据的构建的状态关系图, 在每个纪元开始时重新划分子图算法, 定期调整各分片维护的状态, 实现了更加灵活的状态分配. 然而, 图分配方法的计算开销较高, 包括图构建开销和子图划分开销: 图构建开销通常为 $O(|V|+|E|)$, 涉及遍历交易记录并构建图; 子图划分开销则依赖于所采用的图划分算法, 复杂度较高的算法可能提供更优的分配效果, 但也带来更大的计算开销, 因此需在分配开销与效果之间进行权衡. 此外, 该方法通常假设系统未来的交易行为与历史行为一致, 这使得它在应对实时的账户活动突发变化时, 动态适应性不足. 尤其是在系统需要快速响应并调整机制以应对动态环境时, 该方法无法提供及时有效的调整策略, 从而影响系统的效率[34].

2.1.2 交易级的状态调度方法

交易级的状态调度方法通常在交易达到系统后, 判断是否需要对受影响的状态数据进行调度, 以避免复杂的

跨分片交易处理, 并尽可能将状态数据重新分配至负载较低的分片, 避免分片过载的情况. 相较于基于图论的状态分配方法, 该方法在面对账户的突发行为时能够表现出更高的灵活性和适应性, 可有效避免系统短期负载不均的情况.

Shard scheduler[33]提出了一种面向账户模型的全分片系统状态调度方案. 在每笔交易到达系统后, 矿工启动状态调度程序以判断是否需要进行账户的调度. 当一笔交易被认定为跨分片交易时, 调度器将在其相关的所有分片中, 选择负载最低的分片作为存储该账户的主分片, 并计算将其他账户重新分配至主分片的成本 (计算方法与账户和分片间的交易频率相关), 如果该成本低于保留在原分片的成本, 则执行账户的重新分配. 为确保生成准确的调度策略, 调度器需要记录每个账户与分片之间的交易频率等历史信息. 相较于采用哈希分配方法的全分片系统, Shard scheduler 将 60 个分片的系统吞吐量提升为原来的 2.5 倍, 并且将平均交易确认延迟降低为原来的 28.57%.

Shard scheduler 需要分析历史数据以做出调度决策, 这可能导致决策过程的延迟. 在高交易量的情况下, 这种延迟会使得分片过载状况持续更长时间. 为了解决这一问题, LMChain[34]在调度过程中摆脱了对历史数据的依赖. LMChain 在信标链[12,14,51]的基础上, 通过将分片架构划分为事务分片 (transaction shard, TS) 和信标分片 (beacon shard, BS), 通过在 TS 过载时选择过载交易及相关状态数据转移至 BS 中进行处理, 实现了高吞吐量和低交易等待延迟. 在该系统中, BS 负责维护所有 TS 中的状态数据, 并参与负载转移协议. 然而, 该方案将所有 TS 的过载交易均转移至 BS 中进行处理, 这很可能导致 BS 成为系统的性能瓶颈.

2.1.3 基于机器学习的分配方法

机器学习在区块链分片系统中展现出应对动态环境和优化决策的巨大潜力. 机器学习在动态分片中有两大应用方向: 一是优化分片系统的各类参数以及策略, 包括系统配置参数、状态分配策略等, 以实现短期内性能提升和长期内系统的高效运行; 二是基于历史数据预测未来交易模式, 从而实现更加精准的状态分配.

(1) 使用机器学习优化系统参数

SkyChain[52]首次将深度强化学习 (deep reinforcement learning, DRL)[53–55]引入全分片系统中, 实现了对分片数量、区块大小及分片重新配置间隔等参数的动态调整和系统资源的调度. 后续基于 DRL 的区块链全分片系统[56–58]与 SkyChain 设计相似, 但均未考虑系统状态信息, 未能为状态分配策略提供有效解决方案. SPRING[59]提出了一种基于强化学习 (reinforcement learning, RL) 的状态分片框架. SPRING 将状态分配建模为马尔可夫决策过程, 设计了基于跨分片交易比例和工作负载平衡的奖励函数, 旨在指导代理最大化整体性能并实现负载均衡. 通过将历史交易数据应用于模型训练, SPRING 能够动态适应区块链的特性变化. 由于决策的高效性, SPRING 显著减少了状态分配的计算开销, 能够在分片系统的共识阶段快速执行, 与在配置阶段实施的其他状态分配方法相结合, 能够进一步降低跨分片交易比例, 提升系统性能.

(2) 使用机器学习预测未来交易模式

图分配方法的实施依赖于未来交易模式与历史交易模式相似的假设. 因此, LB-Chain[60]提出了一种基于机器学习的高效状态分配方法, 通过收集历史交易数据, 采用两层长短期记忆网络 (long short-term memory, LSTM) 模型[61]预测系统中各账户未来的交易数量. 基于预测结果, 设计了一种启发式算法, 在每个周期内动态迁移热点账户, 以实现分片间的负载均衡. 由于区块链网络中包含大量的账户, 对全部的账户进行预测显然不可行. 基于对以太坊账户的交易特征分析, 发现仅 0.02% 的热门账户发送了超过 50% 的交易, 因此, LB-Chain 选择仅针对热门账户进行预测. 该方案的交易预测值与实际值的平均误差仅 11%, 并且将重负载分片与轻负载分片的交易数量之比由哈希分配方法的 4 倍降低至 2 倍.

然而, 基于机器学习的动态分配方法需要大量历史数据进行训练, 并且该方法对于计算资源的需求较高. 此外, 该方法的性能高度依赖于输入数据的质量和准确性. 不同时间段的交易数据可能存在显著差异, 导致模型在泛化能力和适应性方面受到限制.

2.1.4 基于分片状态树的状态管理机制

以太坊的状态管理机制基于默克尔压缩前缀树 (Merkle patricia trie, MPT) 实现, 其中全局状态树 (state tree) 用于存储所有账户的状态信息. 状态树以账户地址为键, 账户状态为值, 通过 MPT 结构进行高效的组织和索引.

MPT 通过将相同的前缀合并成一条路径来减少存储空间, 避免了重复存储相同前缀的多个节点, 因此具有较小的存储开销. 同时, 由于 MPT 结合了默克尔树的特性, 使得状态数据具备不可篡改性, 即任何对状态数据的修改都会引起默克尔树根 (Merkle root) 的变化, 从而保证了状态数据的完整性和一致性. 在全局状态树中, 叶子节点存储账户的核心状态字段. 每个账户的状态由以下核心字段组成: $d_i = \{X_i|\eta,\omega,\zeta\}$, 其中, X_i 代表账户地址, η 为交易计数器 (nonce), 表示账户发送的交易次数或合约创建次数, ω 为账户余额 (balance), ζ 为合约特有字段, 包括存储根 (storageRoot) 和合约代码哈希 (codeHash) 等, 用于存储合约数据和标识合约代码.

在全分片系统中, 为支持动态状态分配, 必须设计有效的状态管理机制, 维护状态数据与各个分片之间的映射关系. BrokerChain[15]提出了基于分片状态树 (modified shard state tree, mSST) 的状态管理机制, 确保每个分片都能了解全局状态与分片的存储映射关系, 具体来说, 在 BrokerChain 中, 每个分片都会维护一个唯一的 mSST, 以记录当前状态的存储位置. mSST 的设计基于以太坊的状态树, 保证状态的高效存储与管理 (如图 6 所示). 与传统的状态树不同, BrokerChain 中的 mSST 是基于所有账户状态的存储映射构建的. 具体而言, BrokerChain 通过一个向量 φ 表示账户的存储分片. 该向量定义如下:

$$\varphi = [e_1, e_2, \ldots, e_K] \tag{6}$$

其中, $e_i = 1$ 表示账户存储在分片 i 中, 而 $e_i = 0$ 表示账户不存储在该分片. 例如, 当 $\varphi = 0010$ 时, 表示该状态数据存在分片 3 中. 为了支持该机制, BrokerChain 修改了账户状态的字段, 其数据结构定义如下:

$$d_i = \{X_i|\varphi,\eta,\omega,\zeta\} \tag{7}$$

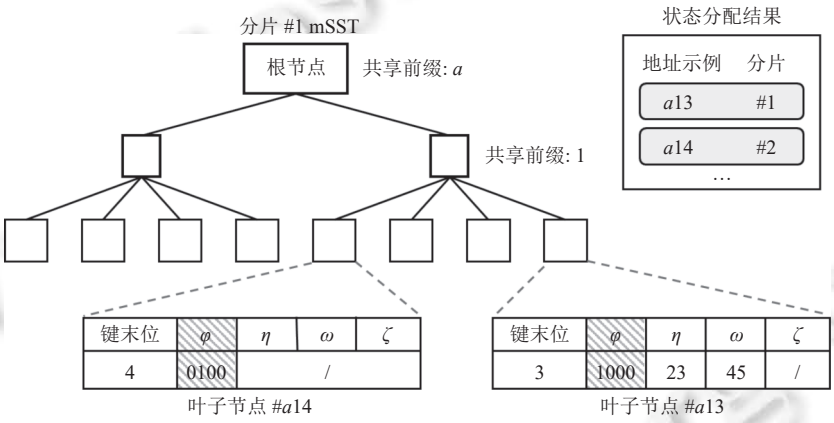

图 6 分片状态树结构

相较于以太坊的账户状态, 新增字段 φ 表示该账户的存储位置, 对于不属于本分片的账户, mSST 中仅存储该账户的分片映射 φ, 而不保留其具体的账户状态数据 (η、ω、ζ 为空).

mSST 充分利用 MPT 自带的路径压缩特性, 结合空字段存储策略, 仅存储与本分片相关的完整账户状态, 从而显著降低了全局存储开销. 此外, 每个分片均独立维护其对应的 mSST, 允许各分片高效查询状态数据的存储位置. 由于 mSST 继承了 MPT 的索引机制, 查询某个账户状态的存储分片仅需在 mSST 中进行一次路径查找. MPT 的查询复杂度通常为 O(l), 其中 l 是账户地址的长度, 即键的位数. 在以太坊中, 账户地址通常为 160 位 (20 字节), 因此, 查询的时间复杂度接近常数时间, 即 O(1).

此外, 通过在 mSST 中查询存储映射信息, BrokerChain 能够快速判断某笔交易是分片内交易还是跨分片交易: 由于分片存储映射向量 φ 的查询仅涉及固定长度的比特运算, 其查询时间复杂度为 O(1).

2.2 状态迁移

动态分配通过调整各分片所维护的状态数据, 能够有效优化系统的跨分片交易比例并实现分片负载均衡. 为满足动态分配的需求, 状态数据需在分片之间进行迁移. 在状态迁移的过程中, 需要满足以下两个要求.

(1) 状态迁移的正确性: 状态迁移必须保证数据的一致性和完整性, 确保迁移后的状态能够正确反映原始分片中的数据. 此外, 迁移机制需要具备防御恶意行为的能力[60,62], 以抵抗如交易操纵、静默攻击及重放攻击等可能影响状态正确性的安全威胁.

(2) 状态迁移的高效性: 状态迁移过程应尽可能减少对系统吞吐量和运行效率的影响. 在迁移期间, 部分交易的处理可能因状态数据的移动而受到暂停, 因此迁移机制需要优化执行效率, 减少迁移带来的额外开销.

根据状态迁移的执行时机, 现有的状态迁移机制可分为周期性迁移和动态迁移两类, 分别在系统的配置阶段和共识阶段执行. 本节将对两类状态迁移机制进行介绍, 表 2 对比了两类迁移机制的特点.

表 2 状态迁移机制对比

分类	触发时机	实现原理	优点	缺点
周期性迁移	配置阶段执行一次	生成状态块记录分配结果与状态信息, 实现状态迁移	不影响系统共识; 实现机制简单	动态适应性差; 无法根据系统波动调整状态
动态迁移	共识阶段动态执行多次	将迁移操作封装为交易, 通过交易锁定机制实现动态迁移	动态适应性强; 可应对系统波动	需要额外的锁定机制避免交易中止

2.2.1 周期性迁移

周期性迁移机制以系统的纪元为单位, 在每个纪元的配置阶段周期性执行, 系统将根据最新的状态分配结果完成状态迁移. 在 BrokerChain 中, 系统架构由一个分区分片 (P-Shard) 和多个工作分片 (M-Shard) 组成, 其中 P-Shard 负责基于图论方法执行全局状态分配, M-Shard 负责共识过程并生成包含交易的普通区块. 在配置阶段, P-Shard 依据最新的交易数据更新状态图, 对状态进行重划分, 并生成状态块 (state block, B_t)[36], 记录最新的状态分配方案及状态数据. 随后, P-Shard 将 B_t 广播至所有 M-Shard, 各分片根据 B_t 执行状态迁移, 确保各节点在后续的共识阶段能够正确验证交易并更新状态数据. 为了确保迁移的正确性, BrokerChain 采用 PBFT 共识机制保证状态块 B_t 的唯一性. 此外, 系统引入 Cuckoo 规则, 定期轮换 P-Shard 和 M-Shard 的成员, 防止节点的恶意加入或退出. 相较于传统的区块链状态同步机制 (节点需要下载并验证完整的交易历史), BrokerChain 通过状态块 B_t 直接记录了系统当前的最新状态, 使节点在同步时无需回溯所有历史交易, 仅需同步状态块中的数据即可获取当前状态, 显著减少了数据同步的规模.

为了进一步减少状态数据同步的开销, tMPT[63]基于状态块 B_t 的内部存储结构, 提出了一种修剪后的默克尔压缩前缀树 (trimmed Merkle patricia trie, tMPT), 提高配置阶段的状态数据同步效率. 在账户余额模型的区块链系统中, 默克尔压缩前缀树 (MPT) 被广泛用于存储系统中的所有账户状态信息. 然而, 相关研究表明[64], 以太坊的历史数据中仅 3% 的账户在一周内发生交易, 5% 的账户在一个月内发生交易, 这意味着大部分账户的状态数据在短时间内几乎没有变化. 然而传统的状态块记录着所有状态数据, 造成不必要的传输开销. 因此, tMPT 通过裁剪不常用的账户状态数据, 仅同步近期活跃账户的状态信息, 而非完整的账户状态数据, 进一步减少了配置阶段传输的数据量, 提高了状态迁移的效率.

周期性迁移在系统的配置阶段执行, 不会影响正常的分片共识流程. 然而, 该方法存在一定的局限性, 即在共识阶段遇到交易负载波动时, 系统无法实时调整状态分配, 可能导致分片负载失衡, 进而影响系统的整体吞吐量.

2.2.2 动态迁移

动态迁移机制在系统的共识阶段被动态触发, 可以满足状态动态调整的需求. 在查阅的相关文献中, LB-Chain[60]和另一项研究[65]提出了基于锁的动态状态迁移机制. 其基本原理是在状态迁移阶段, 对状态数据及其相关的交易进行锁定. 在此期间, 不允许对该状态数据的任何修改操作, 以避免因迁移过程中的数据不一致而导致的交易中止. 我们首先对基于锁的状态迁移问题给出形式化定义, 并明确其中涉及的符号及意义.

$s_{src}, s_{dst} \in S$ 分别表示迁移的源分片和目标分片, $d_i \in D$ 为被迁移的状态. d_i 的相关交易集合记为 $T_i = T_{in} \cup T_{out}$, 其中, T_{in} 和 T_{out} 分别表示从该状态转入和转出资金的交易集合. 定义迁移过程 $M: D \times S \times S \to D$, $M(d_i, s_{src}, s_{dst})$ 包括以下操作.

● 锁定交易集 $T_{\text{lock}} \subseteq T_i$: 在状态迁移过程中, 必须锁定 T_{lock} 中的所有交易, 避免交易被中止.

● 数据迁移 $\varphi: D \times S \times S \to D$: 此函数表示将状态数据 d_i 从 s_{src} 迁移至 s_{dst}, 即 $d_i' = \varphi(d_i, s_{\text{src}}, s_{\text{dst}})$, 迁移后 $f(d_i', s_{\text{dst}}) = 1$.

● 状态更新 $\psi: D \times S \to B$: 在 s_{dst} 上更新 d_i', 确保数据在新分片上的一致性. 函数 $\psi(d_i', s_{\text{dst}})$ 将返回布尔值, 表示更新操作是否成功.

● 释放交易集: 完成状态迁移和更新后, 解锁并处理 T_{lock} 中的交易, 恢复交易的正常执行.

LB-Chain[60]首次提出了完整的状态迁移机制, 该方案将状态迁移操作封装在交易中, 执行过程包括生成并签署交易、打包区块、达成共识和更新状态. 为防范交易操纵, LB-Chain 对迁移交易进行严格验证, 包括状态一致性、签名验证、余额一致性和分片一致性. 针对静默攻击, 客户端和 s_{dst} 可通知 s_{src} 重新发送交易. 为避免重放攻击, 引入迁移随机数, 确保交易顺序的连续性和唯一性. 此外, 为确保状态迁移期间交易的正确处理, LB-Chain 会在 s_{src} 中锁定被迁移的状态及其所有的相关交易, 即 $T_{\text{lock}}=T_i$, 并将其与状态数据一同迁移至 s_{dst} 中进行处理, 从而避免了相关交易可能面临的被中止的情况. 然而, LB-Chain 存在以下两个局限: 首先, 迁移期间所有的相关交易均被锁定, 导致这些交易需要较长时间才能得到确认, 从而影响系统的整体吞吐量和平均交易确认延迟. 其次, LB-Chain 并未针对迁移失败的情况提出有效的解决方案, 这导致该方案存在一定的系统安全漏洞.

为解决上述问题, 文献 [65] 提出了一种精细化的交易锁定机制, 以减少状态迁移阶段对相关交易的影响. 针对第 1 个问题, 该方案在状态由 s_{src} 迁移至 s_{dst} 期间, 仅锁定从该状态转出资金的相关交易, 即 $T_{\text{lock}}=T_{\text{out}}$, 依然允许向该状态转入资金的相关交易 T_{in} 的处理. 当状态数据完成迁移后, s_{dst} 向 s_{src} 发送通知, s_{src} 将该状态相关的最新状态 d_i' 和未完成交易发送给 s_{dst}. 在此过程中, 即使状态处于迁移过程, 仍会保持更新. 该方案减少了被锁定的交易数量, 确保向该状态转入资金的交易的正常处理, 不会因为迁移而中断, 将交易的完成时间降低至 LB-Chain 的 30% 左右. 针对第 2 个问题, 该方案为两类迁移失败情况提供了解决方案. 第 1 类是迁移超时, 发生在 s_{dst} 未能成功验证状态 d_i', 从而未回复 s_{src}, 导致 s_{src} 在迁移窗口结束后认为迁移失败. 为避免此类失败, 设置超时查询机制, s_{src} 需要在迁移窗口内主动向 s_{dst} 发送查询请求, 以确认状态迁移结果, 并在失败时解锁状态数据及相关交易. 第 2 类是迁移过程中的消息传播延迟, s_{dst} 成功验证迁移但 s_{src} 未及时接收到该验证消息, 导致 s_{src} 误认为迁移失败并解锁账户, 导致该状态数据在两个分片中出现重复. 为避免此类失败, 设置状态确认机制, s_{src} 在迁移窗口到期后不会直接解锁该状态数据, 而是继续向 s_{dst} 查询, 直到确认迁移成功后再执行解锁操作.

2.3 跨分片交易处理

在区块链全分片系统中, 节点只需维护部分区块链状态数据, 跨分片交易的处理需要在多个分片之间进行数据验证和状态一致更新. 为确保系统的正确性, 跨分片交易必须满足 ACID 属性[66], 即原子性、一致性、隔离性和持久性. 其中, 原子性要求交易在所有相关分片中要么全部成功提交, 要么全部中止, 以避免部分提交的情况; 一致性则要求在交易执行后, 各分片的状态保持一致, 以防止数据冲突. 现有的跨分片交易处理方法主要包括基于两阶段提交 (two-phase commit protocol, 2PC) 和基于中继交易的方案. 表 3 总结了这两种方案的优缺点. 此外, 本节还将讨论若干新型的跨分片交易处理方法.

表 3 跨分片交易处理方案对比

分类	实现原理	优点	缺点
2PC	基于两阶段提交协议, 通过协调者管理跨分片交易处理	确保跨分片交易的强一致性, 适用于复杂事务处理	依赖协调者, 交易的提交需要锁定相关状态
中继交易	设计中继交易, 保证交易的“最终原子性”	实现无锁异步处理, 支持交易并行处理	适用于基于账户模型的全分片系统

2.3.1 基于两阶段提交 (2PC) 的方案

两阶段提交协议 (2PC)[67]被广泛应用于分布式系统, 能够确保分布式事务中数据的强一致性[68]. 该协议分为准备阶段和提交阶段. 在基于 2PC 协议的方案中, 存在一个协调者负责管理多个分片, 确保交易的一致性提交. 在准备阶段, 协调者从输入分片收集验证交易输入有效性的证明消息. 在提交阶段, 协调者将收集到的证明消息发送至所有相关分片. 根据验证结果, 跨分片交易将在每个分片中被一致地提交或中止: 如果所有输入均有效, 则交易

有效并最终被提交; 否则, 交易无效并被中止. 在处理跨分片交易的过程中, 交易的输入状态将暂时被锁定, 直到交易最终提交或中止, 以避免输入的双重花费.

Chainspace[38]引入了一种基于分片驱动的两阶段提交机制 S-BAC (sharded Byzantine atomic commit) 处理跨分片交易, 并结合并行处理与拜占庭容错优化, 提高了跨分片交易的处理效率. OmniLedger[36]则采用基于客户端驱动的两阶段提交机制 (如图 7 所示). 然而, 上述由客户端驱动的 2PC 协议依赖于客户端检索各阶段的证明消息, 在一定程度上违背了轻客户端的原则[37]. 此外, 2PC 协议的锁定机制导致与被锁定状态相关的其他交易暂时无法被处理, 并且存在一定的加锁/解锁开销.

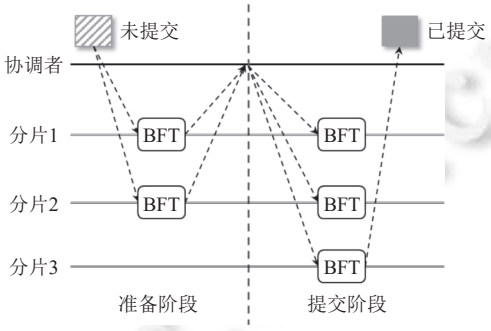

图 7 客户端驱动的 2PC 协议

2.3.2 基于中继交易的方案

基于中继交易的跨分片交易处理方案最初在 Monoxide[39]中提出, 通常适用于基于账户/余额模型的区块链系统. 该方案将跨分片交易的处理分为提款和存款两个操作并强调交易的“最终原子性”, 即对于跨分片交易, 系统首先执行提款操作, 随后与其他交易并行处理, 最后在适当时机执行存款操作. 这意味着, 一旦提款操作被确认, 存款操作必将在后续某个时刻得到执行. 具体而言, 跨分片交易首先在输入分片中进行验证并执行提款操作, 以确保资金从源账户中扣除. 随后, 输入分片生成中继交易 (relay transaction), 并将其转发至相应的输出分片. 输出分片验证接收到的中继交易的有效性, 并执行存款操作, 确保资金正确存入目标账户 (如图 8 所示). 该方案通过使用中继交易进行简单的消息交换, 实现无锁异步的跨分片交易处理.

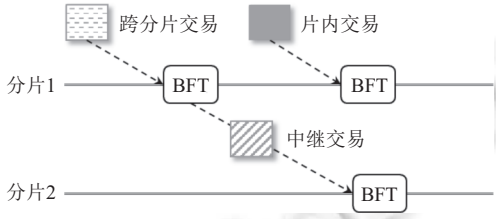

图 8 基于中继交易的跨片交易处理

2.3.3 其 他

在 RapidChain[37]协议中, 分片领导者根据交易的输入和输出将跨分片交易拆分为多个子交易, 并利用这些子交易将相关的 UTXO 转移至同一分片进行处理. 为降低跨分片消息传播的开销, RapidChain 采用基于 Kademlia 的路由算法, 将子交易传递给相应的分片, 确保分片间的高效通信. 然而, 该方法也存在一些缺陷. 首先, 由于交易拆分导致的系统中交易数量增加, 从而导致额外的通信开销. 其次, 分片领导者的角色可能引发安全问题, 尤其在领导者存在恶意行为的情况下. 最后, 对路由算法的依赖可能成为系统的潜在瓶颈.

BrokerChain[15]通过引入“中介账户 (broker account)”作为跨分片交易的中介, 协调多个分片之间的交易执行, 以确保交易的顺利完成和状态的一致性. 这些中介账户需具备足够的代币, 并通过质押资产来获得成为中介账户

的资格. BrokerChain 将跨分片交易的处理分为两个阶段, 分别在源分片和目标分片中执行, 通过锁定代币和随机数机制来防止重放攻击, 并确保交易的原子性. 然而, 该机制的实际应用依赖于中介账户的资产质押, 在缺乏有效激励机制的情况下, 可能难以吸引足够数量的用户担任中介账户, 从而影响该机制的高效性与可扩展性.

X-Shard[45]提出了一种乐观策略来处理多输入多输出的跨分片交易. 该方案中, 跨分片交易初步被乐观地认为有效, 只有在被提交到目标分片时才进行有效性验证. X-Shard 引入了由分片管理的网关账户来处理跨分片交易, 每笔交易被分解为多个内部分片的子交易, 子交易首先在输入分片中将代币从用户账户转移至网关账户, 最终由网关账户将资金转移至接收账户. 各输入分片可以乐观并行处理子交易, 无需等待其他分片的结果. 对于验证失败的跨分片交易, 系统将回滚以确保交易的原子性. 此外, X-Shard 设计了一种基于阈值签名的跨分片提交协议, 将跨分片交易的通信开销从 $O(n^2)$ 降低至 $O(n)$. 然而, 该方案通过网关账户集中处理跨分片交易, 与 BrokerChain 面临着相同的缺乏有效激励机制的问题.

3 状态分片现有方案对比与分析

区块链状态分片技术的研究围绕状态分配、状态迁移和跨分片交易处理这 3 大关键问题展开, 其目标是通过优化这些关键环节, 有效缩小系统实际性能与理论预期之间的差距. 表 4 对近年来相关研究成果进行了梳理并对比, 总结了各方案在解决上述关键问题方面的技术特点.

表 4 状态分片技术对比总结

年份	分片协议	状态模型	状态分配			跨片交易处理	状态迁移
			方法	分片数量/跨片交易比例	负载分布		
2016	Elastico[10]	UTXO	—	—	—	—	无需迁移
2018	OmniLedger[36]	UTXO	哈希分配	32/99.80%*	—	2PC	无需迁移
2019	Monoxide[39]	账户模型	哈希分配	32/96.20%*	—	中继交易	无需迁移
2019	OptChain[40]	UTXO	图分配	32/21.70%†	对比OmniLedger, 降低约90%交易队列拥堵	其他#	—
2021	Shard scheduler[33]	账户模型	状态调度	10/60%–70%†	60分片时, 延迟比哈希分配低3.5倍	2PC	—
2022	BrokerChain[15]	账户模型	图分配	32/65.70%†	对比Monoxide, 具有更小的分片工作负载方差	其他	周期性迁移
2023	LB-Chain[60]	账户模型	机器学习	—	对比Monoxide, 将最大最小负载比由4倍降至2倍	中继交易	动态迁移

注: †表示从原文献中获得的数据; *表示使用真实交易并按照论文设计的方法测试得到的数据; —表示协议设计中并未考虑该属性或是未明确给出该项属性的相关数据; #表示OptChain为部署在客户端的轻量化状态配方案, 可与不同的跨分片交易处理方案结合

状态分片技术所涵盖的 3 个关键问题密切相关, 并且在其研究过程中相互促进. 最初, Elastico[10]协议并未采用状态分片技术, 节点仍面临巨大的存储压力. 为解决此问题, 后续协议引入状态分片技术. 在状态分片技术的早期研究中, 大量工作集中于设计跨分片交易的处理方法, 研究重点在于如何在分片间协同的前提下, 保证交易处理的原子性和一致性. 然而, 为简化系统实现, 大多数协议采用静态分配方法, 这导致系统存在着高跨分片交易比例和热分片的现象, 显著限制了系统性能. 随着这些问题的暴露, 动态分配方法成为研究热点. 通过动态调整分片维护的状态数据, 降低了跨分片交易比例, 同时保证了分片间的工作负载均衡, 显著改善了系统性能表现. 在此基础上, 跨分片交易处理方法的研究逐步从保证交易的 ACID 属性转向追求更高效的方案, 并与不同状态分配方法结合, 进一步提高处理效率. 此外, 动态分配方法的研究也促使状态迁移机制的研究需求逐渐显现. 状态迁移机制需要在动态调整状态分配过程中, 确保分片间状态数据的快速、安全迁移.

从表 4 中可以看出, 现有研究多集中于状态分片中某一关键问题. 为提升系统性能, 通常引入多重技术进行优化, 但这一过程往往忽略了分片系统的完整性与简洁性, 为理论的实际应用带来了诸多挑战. 此外, 不同的解决方案通常依赖于特定的应用场景假设. 例如, 图分配方法通常假设账户未来的交易特征符合历史趋势, 而状态调度方法则以账户行为存在突发波动为前提. 这些假设导致不同方案在设计过程中的侧重点和适用范围各异. 因此, 在研

究过程中必须充分考虑这些假设的现实合理性及其适用范围, 以确保研究能够在实际系统中有效应用.

4 研究展望

状态分片技术对于解决区块链扩展性问题以及实现更高效、更可持续的区块链系统具有重要的现实意义, 未来可从以下 4 个方面展开研究.

(1) 状态分配频率与系统性能之间的权衡

动态分配方法的执行需消耗额外的网络和计算资源. 例如, 图分配方法需要消耗额外的资源开销用于构建和划分子图. 频繁的状态分配在某些情况下可能为系统带来负面影响. 因此, 未来可深入研究状态分配执行频率与系统性能间的关系, 通过量化分析状态分配频率对系统性能的影响, 建立相应的数学模型, 为动态分配策略的优化提供理论依据.

(2) 多目标优化的状态分片技术研究

在状态分片技术的研究中存在多个优化目标, 包括降低跨分片交易比例、平衡分片负载以及减少状态迁移的数据量. 然而, 这些目标之间往往存在相互制约关系, 难以在同一系统中同时实现最优. 因此, 未来可研究多目标优化的状态分片技术, 探索上述优化目标之间的平衡点. 例如, 可采用权重法为目标分配权重, 将其合并为一个综合目标进行优化, 即最大化系统吞吐量. 其中, 跨分片交易比例的权重取决系统中跨分片交易的处理方法, 负载均衡的权重取决于各分片处理能力, 而状态迁移数据量的权重则取决于具体的迁移策略.

(3) 区块链分片系统的激励机制设计

状态分片的高效执行依赖于矿工、账户等各方参与者诚实且积极地响应系统需求. 在此过程中, 激励机制需要确保参与者的积极性并防范潜在的恶意行为, 以维护系统的安全性和可持续发展. 以 BrokerChain[15]和 X-Shard[45]为例, 两者均引入中间账户协调处理跨分片交易, 这要求设计有效的激励机制, 确保中间账户愿意积极诚实地提供服务. 针对这一问题, 文献 [69] 开发了一个去中心化金融 (DeFi) 应用[70]BrokerFi, 并设计了 Broker2Earn 协议[71], 用于激励中间账户在跨分片交易中的积极参与. 这一工作不仅在理论上构建了针对分片环境的激励机制框架, 同时也实现了系统原型, 展示了该机制在实际应用场景中的可行性. 尽管上述研究为分片系统中的激励机制设计提供了有益尝试, 但从整体来看, 当前学术界在该方向的研究仍处于起步阶段, 现有成果在复杂分片系统中的适用性和有效性尚未得到充分验证. 因此, 未来应重点研究如何根据贡献在不同类型的参与者之间合理分配奖励, 以最大化系统性能.

(4) 区块链分片技术的落地应用探索

分片技术作为提升区块链系统扩展能力的重要手段, 已应用在多种场景中. 但是由于应用特征的差异, 该技术在不同应用场景中的可行性与适配性仍需进一步深入研究. 在动态异构的物联网场景中, 状态分片可以根据设备的计算存储能力自适应地调整分片结构, 实现对资源受限节点的高效适配与任务分配[72]; 在高并发的去中心化金融场景中, 状态分片可以显著降低存储开销与状态读写访问冲突, 提高系统吞吐能力与响应速度[73]. 然而, 不同的应用场景在节点规模、数据存储需求与状态数据特征等方面存在显著差异. 例如, 医疗数据共享场景中节点数量较少, 数据更新频率较低, 且对吞吐量的需求通常较低; 而供应链溯源场景中节点众多且异构, 数据更新频繁, 吞吐量的需求较高. 因此, 未来研究应聚焦于结合实际场景需求, 选择合适的分片策略, 并对其进行针对性优化, 以实现更高效的系统资源利用与整体性能提升.

5 总 结

本文首先介绍了区块链分片技术的背景知识, 结合区块链全分片系统的工作机制, 梳理了区块链状态分片技术的 3 个核心问题, 即状态分配、状态迁移以及跨分片交易处理. 随后, 围绕上述关键问题, 对其现有主流实现方法以及研究现状进行了分析和综述, 总结并对比其优势与不足. 接下来, 从上述 3 个角度对近年来区块链状态分片技术最新的研究成果进行总结与对比. 最后, 探讨了状态分片技术未来可能面临的问题以及未来潜在的发展方向,

以期为后续研究提供有价值的参考和借鉴.

References

[1] Yuan Y, Wang FY. Blockchain: The state of the art and future trends. Acta Automatica Sinica, 2016, 42(4): 481–494 (in Chinese with English abstract). [doi: 10.16383/j.aas.2016.c160158]

[2] Li JJ, Qin R, Ding WW, Wang G, Wang T, Wang FY. A new framework for Web3-powered decentralized autonomous organizations and operations. Acta Automatica Sinica, 2023, 49(5): 985–998 (in Chinese with English abstract). [doi: 10.16383/j.aas.c220753]

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

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

[5] Korpela K, Hallikas J, Dahlberg T. Digital supply chain transformation toward blockchain integration. In: Proc. of the 2017 Hawaii Int'l Conf. on System Sciences (HICSS). Big Island, 2017. 4182–4191.

[6] Mettler M. Blockchain technology in healthcare: The revolution starts here. In: Proc. of the 18th IEEE Int'l Conf. on e-Health Networking, Applications and Services (HealthCom). Munich: IEEE, 2016. 1–3. [doi: 10.1109/HealthCom.2016.7749510]

[7] Shi JS, Li R. Survey of blockchain access control in Internet of Things. Ruan Jian Xue Bao/Journal of Software, 2019, 30(6): 1632–1648 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5740.htm [doi: 10.13328/j.cnki.jos.005740]

[8] Croman K, Decker C, Eyal I, Gencer AE, Juels A, Kosba A, Miller A, Saxena P, Shi E, Sirer EG, Song D, Wattenhofer R. On scaling decentralized blockchains: A position paper. In: Proc. of the 2016 Int'l Workshops on Financial Cryptography and Data Security. Christ Church: Springer, 2016. 106–125. [doi: 10.1007/978-3-662-53357-4_8]

[9] 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]

[10] 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]

[11] Munro A. Ethereum 2.0 roadmap, timeline and implications. 2021. https://www.finder.com.au/news/ethereum-2-0-roadmap-timeline-and-implications

[12] The Zilliqa Team. The Zilliqa project: A secure, scalable blockchain platform. 2018. https://zilliqa.com/

[13] QuarkChain. Building the super world computer. 2024. https://quarkchain.io/

[14] Harmony Team. Harmony: Technical whitepaper. 2024. https://www.allcryptowhitepapers.com/harmony-whitepaper/

[15] Huang HW, Peng XW, Zhan JZ, Zhang SY, Lin Y, Zhen 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]

[16] Li SZ, Yu MC, Yang CS, Avestimehr AS, Kannan S, Viswanath P. PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously. IEEE Trans. on Information Forensics and Security, 2021, 16: 249–261. [doi: 10.1109/TIFS.2020.3009610]

[17] BitcoinWiki. Bitcoin scalability problem. 2017. https://bitcoinwiki.org/wiki/bitcoin-scalability-problem

[18] Chernet HF, Jilledi SK. A next-generation smart contract and decentralized blockchain platform: A case study on ethiopia. Journal of Business Analytics and DataVisualization, 2020, 1(1): 28–34.

[19] Visa. Benefits of accepting Visa. 2014. https://usa.visa.com/content_library/modal/benefits-accepting-visa.html

[20] Bitinfocharts. Cryptocurrency statistics. 2024. https://bitinfocharts.com/

[21] Bitnodes. Reachable bitcoin nodes—Bitnodes. 2023. https://bitnodes.io/

[22] 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]

[23] Yuan Y, Ni XC, Zeng S, Wang FY. Blockchain consensus algorithms: The state of the art and future trends. Acta Automatica Sinica, 2018, 44(11): 2011–2022 (in Chinese with English abstract). [doi: 10.16383/j.aas.2018.c180268]

[24] Yang X, Zhang N, Lou WJ, Hou YT. A survey of distributed consensus protocols for blockchain networks. IEEE Communications Surveys & Tutorials, 2020, 22(2): 1432–1465. [doi: 10.1109/COMST.2020.2969706]

[25] Göbel J, Krzesinski AE. Increased block size and Bitcoin blockchain dynamics. In: Proc. of the 27th Int'l Telecommunication Networks and Applications Conf. (ITNAC). Melbourne: IEEE, 2017. 1–6. [doi: 10.1109/ATNAC.2017.8215367]

[26] Wang G, Shi ZJ, Nixon M, Han S. SoK: Sharding on blockchain. In: Proc. of the 1st ACM Conf. on Advances in Financial Technologies. Zurich: ACM, 2019. 41–61. [doi: 10.1145/3318041.3355457]

[27] Wang Q, Yu JS, Chen SP, Xiang Y. SoK: DAG-based blockchain systems. ACM Computing Surveys, 2023, 55(12): 261. [doi: 10.1145/3576899]

[28] 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]

[29] Li F, Li ZR, Li H. Research on the progress in cross-chain technology of blockchains. Ruan Jian Xue Bao/Journal of Software, 2019, 30(6): 1649–1660 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5741.htm [doi: 10.13328/j.cnki.jos.005741]

[30] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments (DRAFT Version 0.5.9.2). 2016. https://nakamotoinstitute.org/library/lightning-network/

[31] Thibault LT, SarryT, Hafid AS. Blockchain scaling using rollups: A comprehensive survey. IEEE Access, 2022, 10: 93039–93054. [doi: 10.1109/ACCESS.2022.3200051]

[32] Gervais A, Karame GO, Wüst K, Glykantzis V, Ritzdorf H, Capkun S. On the security and performance of proof of work blockchains. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 3–16. [doi: 10.1145/2976749.2978341]

[33] 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]

[34] Hu DC, Wang JR, Liu XL, Li Q, Li KQ. LMChain: An efficient load-migratable beacon-based sharding blockchain system. IEEE Trans. on Computers, 2024, 73(9): 2178–2191. [doi: 10.1109/TC.2024.3404057]

[35] Wang S, Ouyang LW, Yuan Y, Ni XC, Han X, Wang FY. Blockchain-enabled smart contracts: Architecture, applications, and future trends. IEEE Trans. on Systems, Man, and Cybernetics: Systems, 2019, 49(11): 2266–2277. [doi: 10.1109/TSMC.2019.2895123]

[36] 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]

[37] 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]

[38] 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, 2018.

[39] 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 (NSDI 2019). Boston: USENIX, 2019. 95–112.

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

[41] 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]

[42] Li CL, Huang HW, Zhao YT, Peng XW, Yang RJ, Zheng ZB, Guo S. Achieving scalability and load balance across blockchain shards for state sharding. In: Proc. of the 41st Int'l Symp. on Reliable Distributed Systems (SRDS). Vienna: IEEE, 2022. 284–294. [doi: 10.1109/SRDS55811.2022.00034]

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

[44] Jia LP, Liu YX, Wang KY, Sun Y. Estuary: A low cross-shard blockchain sharding protocol based on state splitting. IEEE Trans. on Parallel and Distributed Systems, 2024, 35(3): 405–420. [doi: 10.1109/TPDS.2024.3351632]

[45] Xu J, Ming YL, Wu ZH, Wang C, Jia XH. X-Shard: Optimistic cross-shard transaction processing for sharding-based blockchains. IEEE Trans. on Parallel and Distributed Systems, 2024, 35(4): 548–559. [doi: 10.1109/TPDS.2024.3361180]

[46] Karypis G, Kumar V. METIS: Unstructured graph partitioning and sparse matrix ordering system version 2.0. METIS, 1995.

[47] Meyerhenke H, Sanders P, Schulz C. Parallel graph partitioning for complex networks. IEEE Trans. on Parallel and Distributed Systems, 2017, 28(9): 2625–2638. [doi: 10.1109/TPDS.2017.2671868]

[48] Li H, Liu YN, Yuan H, Yang SQ, Yun JP, Qiao SJ, Huang JB, Cui JT. Research on dynamic graph partitioning algorithms: A survey. Ruan Jian Xue Bao/Journal of Software, 2022, 34(2): 539–564 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6705.htm [doi: 10.13328/j.cnki.jos.006705]

[49] Ugander J, Backstrom L. Balanced label propagation for partitioning massive graphs. In: Proc. of the 6th ACM Int'l Conf. on Web Search and Data Mining. Rome: ACM, 2013. 507–516. [doi: 10.1145/2433396.2433461]

[50] 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]

[51] Skidanov A, Polosukhin I. Nightshade: Near protocol sharding design. 2019. https://nearprotocol.com/downloads/Nightshade.pdf

[52] Zhang JT, Hong ZC, Qiu XY, Zhan YF, Guo S, Chen WH. SkyChain: A deep reinforcement learning-empowered dynamic blockchain sharding system. In: Proc. of the 49th Int'l Conf. on Parallel Processing. Edmonton: ACM, 2020. 3. [doi: 10.1145/3404397.3404460]

[53] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge: MIT Press, 1998.

[54] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436–444. [doi: 10.1038/nature14539]

[55] Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 2017, 34(6): 26–38. [doi: 10.1109/MSP.2017.2743240]

[56] Yang ZX, Yang RZ, Yu FR, Li M, Zhang YH, Teng YL. Sharded blockchain for collaborative computing in the Internet of Things: Combined of dynamic clustering and deep reinforcement learning approach. IEEE Internet of Things Journal, 2022, 9(17): 16494–16509. [doi: 10.1109/JIOT.2022.3152188]

[57] Yuan SJ, Li J, Liang JH, Zhu YX, Yu X, Chen JP, Wu CT. Sharding for blockchain based mobile edge computing system: A deep reinforcement learning approach. In: Proc. of the 2021 IEEE Global Communications Conf. (GLOBECOM). Madrid: IEEE, 2021. 1–6. [doi: 10.1109/GLOBECOM46510.2021.9685883]

[58] Yun J, Goh Y, Chung JM. DQN-based optimization framework for secure sharded blockchain systems. IEEE Internet of Things Journal, 2021, 8(2): 708–722. [doi: 10.1109/JIOT.2020.3006896]

[59] Li PZ, Song MX, Xing MZ, Xiao Z, Ding QY, Guan SJ, Long JY. SPRING: Improving the throughput of sharding blockchain via deep reinforcement learning based state placement. In: Proc. of the 2024 ACM on Web Conf. Singapore: ACM, 2024. 2836–2846. [doi: 10.1145/3589334.3645386]

[60] 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]

[61] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural Computation, 2000, 12(10): 2451–2471. [doi: 10.1162/089976600300015015]

[62] Tian GH, Hu YH, Chen XF. Research progress on attack and defense techniques in block-chain system. Ruan Jian Xue Bao/Journal of Software, 2021, 32(5): 1495–1525 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6213.htm [doi: 10.13328/j.cnki.jos.006213]

[63] Huang HW, Zhao YT, Zheng ZB. tMPT: Reconfiguration across blockchain shards via trimmed Merkle patricia trie. In: Proc. of the 31st IEEE/ACM Int'l Symp. on Quality of Service (IWQoS). Orlando: IEEE, 2023. 1–10. [doi: 10.1109/IWQoS57198.2023.10188757]

[64] Kim JY, Lee J, Koo Y, Park S, Moon SM. Ethanos: Efficient bootstrapping for full nodes on account-based blockchain. In: Proc. of the 16th European Conf. on Computer Systems. ACM, 2021. 99–113. [doi: 10.1145/3447786.3456231]

[65] Huang HW, Lin Y, Zheng ZB. Account migration across blockchain shards using fine-tuned lock mechanism. In: Proc. of the 2024 IEEE Conf. on Computer Communications. Vancouver: IEEE, 2024. 271–280. [doi: 10.1109/INFOCOM52122.2024.10621244]

[66] Haerder T, Reuter A. Principles of transaction-oriented database recovery. ACM Computing Surveys (CSUR), 1983, 15(4): 287–317. [doi: 10.1145/289.291]

[67] Bernstein PA, Hadzilacos V, Goodman N. Concurrency Control and Recovery in Database Systems. Reading: Addison-Wesley, 1987.

[68] Zhu T, Guo JW, Zhou H, Zhou X, Zhou AY. Consistency and availability in distributed database systems. Ruan Jian Xue Bao/Journal of Software, 2018, 29(1): 131–149 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5433.htm [doi: 10.13328/j.cnki.jos.005433]

[69] Zheng J, Chen QD, Su CH, Huang HW. BrokerFi: A DeFi dapp built upon broker-based blockchain. In: Proc. of the 29th IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS). IEEE, 2023. 1817–1825. [doi: 10.1109/ICPADS60453.2023.00251]

[70] Jensen JR, von Wachter V, Ross O. An introduction to decentralized finance (DeFi). Complex Systems Informatics and Modeling Quarterly, 2021, 26(26): 46–54. [doi: 10.7250/csimq.2021-26.03]

[71] Chen QD, Huang HW, Yin ZK, Ye G, Yang QL. Broker2Earn: Towards maximizing broker revenue and system liquidity for sharded blockchains. In: Proc. of the 2024 IEEE Conf. on Computer Communications. IEEE, 2024. 251–260. [doi: 10.1109/INFOCOM52122.2024.10621431]

[72] Cai T, Chen WH, Zhang JT, Zheng ZB. SmartChain: A dynamic and self-adaptive sharding framework for IoT blockchain. IEEE Trans. on Services Computing, 2024, 17(2): 674–688. [doi: 10.1109/TSC.2024.3376242]

[73] Zhang JT, Chen WH, Hong ZC, Xiao G, Du LL, Zheng ZB. Efficient execution of arbitrarily complex cross-shard contracts for blockchain sharding. IEEE Trans. on Computers, 2024, 73(5): 1190–1205. [doi: 10.1109/TC.2024.3365929]

附中文参考文献

[1] 袁勇, 王飞跃. 区块链技术发展现状与展望. 自动化学报, 2016, 42(4): 481–494. [doi: 10.16383/j.aas.2016.c160158]
[2] 李娟娟, 秦蕊, 丁文文, 王戈, 王坛, 王飞跃. 基于 Web3 的去中心化自治组织与运营新框架. 自动化学报, 2023, 49(5): 985–998. [doi: 10.16383/j.aas.c220753]
[7] 史锦山, 李茹. 物联网下的区块链访问控制综述. 软件学报, 2019, 30(6): 1632–1648. http://www.jos.org.cn/1000-9825/5740.htm [doi: 10.13328/j.cnki.jos.005740]
[23] 袁勇, 倪晓春, 曾帅, 等. 区块链共识算法的发展现状与展望. 自动化学报, 2018, 44(11): 2011–2022. [doi: 10.16383/j.aas.2018.c180268]
[28] 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]
[29] 李芳, 李卓然, 赵赫. 区块链跨链技术进展研究. 软件学报, 2019, 30(6): 1649–1660. http://www.jos.org.cn/1000-9825/5741.htm [doi: 10.13328/j.cnki.jos.005741]
[48] 李贺, 刘延娜, 袁航, 杨舒琪, 韵晋鹏, 乔少杰, 黄健斌, 崔江涛. 动态图划分算法研究综述. 软件学报, 2023, 34(2): 539–564. http://www.jos.org.cn/1000-9825/6705.htm [doi: 10.13328/j.cnki.jos.006705]
[62] 田国华, 胡云瀚, 陈晓峰. 区块链系统攻击与防御技术研究进展. 软件学报, 2021, 32(5): 1495–1525. http://www.jos.org.cn/1000-9825/6213.htm [doi: 10.13328/j.cnki.jos.006213]
[68] 朱涛, 郭进伟, 周欢, 周烜, 周傲英. 分布式数据库中一致性与可用性的关系. 软件学报, 2018, 29(1): 131–149. http://www.jos.org.cn/1000-9825/5433.htm [doi: 10.13328/j.cnki.jos.005433]

作者简介

苏琳萱, 硕士生, 主要研究领域为区块链, 分布式存储系统.
张晓, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为分布式存储系统, 云计算与云存储系统, 系统评测与仿真, 区块链.
于锦阳, 硕士生, 主要研究领域为区块链, 分布式存储系统.
何赫娘, 硕士生, 主要研究领域为区块链, 分布式系统.
王锦江, 博士生, 主要研究领域为云计算, 多级内存管理与性能优化, 分布式存储.
黄志杰, 博士, 副教授, CCF 专业会员, 主要研究领域为编码理论, 存储系统, 区块链.