软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(8):3802−3830 [doi: 10.13328/j.cnki.jos.007269] [CSTR: 32375.14.jos.007269]

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

基于原生链的跨 Rollup 机制研究*

张子龙[1,2], 贾林鹏[1], 蒋硕轩[1,2], 孙 毅[1,2]

[1](中国科学院 计算技术研究所, 北京 100190)

[2](中国科学院大学 计算机科学与技术学院, 北京 100049)

通信作者: 孙毅, E-mail: sunyi@ict.ac.cn

摘 要: Rollup 是一种新兴的区块链链下交易处理方案. 随着应用的持续发展, 不同类型 Rollup 间的互操作需求日益增长. 现有 Rollup 间互操作方案通常使用第三方服务商来协助完成, 存在着信任假设的安全风险和单点故障等问题. 基于原生链完成 Rollup 间互操作无需引入新的信任假设, 但会消耗原生链的计算与存储资源, 降低原生链的交易吞吐量, 从而严重影响跨 Rollup 性能. 基于此, 提出一种基于原生链的跨 Rollup 方案, 通过聚合交易批量处理的方式, 有效减少单笔交易的链上平均计算与存储资源开销. 具体而言, 提出基于零知识证明的交易有效性证明方案, 显著减少交易有效性验证的链上计算开销. 提出基于索引表数据压缩的交易存储方案, 降低跨 Rollup 交易的平均链上存储开销. 提出聚合规模均衡调整算法, 得到最优的聚合规模, 实现链上资源消耗与处理时延之间的平衡. 最后, 对方案进行实验验证. 实验结果表明, 所提方案在完全去信任化的前提下, 能降低链上计算开销和存储开销, 实现链上资源消耗与处理时延的平衡, 并且与现有跨 Rollup 方案相比, 所提方案的系统吞吐量也具有很好的表现.

关键词: 区块链; Rollup; 零知识证明; 多目标优化

中图法分类号: TP311

中文引用格式: 张子龙, 贾林鹏, 蒋硕轩, 孙毅. 基于原生链的跨Rollup机制研究. 软件学报, 2025, 36(8): 3802–3830. http://www.jos.org.cn/1000-9825/7269.htm

英文引用格式: Zhang ZL, Jia LP, Jiang SX, Sun Y. Research on Cross-Rollup Mechanism Based on Native Blockchain. Ruan Jian Xue Bao/Journal of Software, 2025, 36(8): 3802–3830 (in Chinese). http://www.jos.org.cn/1000-9825/7269.htm

Research on Cross-Rollup Mechanism Based on Native Blockchain

ZHANG Zi-Long[1,2], JIA Lin-Peng[1], JIANG Shuo-Xuan[1,2], SUN Yi[1,2]

[1](Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

[2](School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Rollup is an emerging off-chain transaction processing solution for blockchains. With the continuous development of applications, the need for interoperability among different types of Rollups is increasingly growing. Existing cross-Rollup interoperability solutions typically rely on third-party service providers to assist in completion, which brings about security risks such as new trust assumptions and single-point-of-failure issues. Completing interoperability among Rollups based on the native chain does not require introducing new trust assumptions, but will consume the computing and storage resources of the native chain, reduce the transaction throughput of the native chain, and thus seriously affect the performance of cross-Rollup. Based on this, this study proposes a cross-Rollup mechanism based on a native blockchain. By aggregating and processing transactions in batches, it effectively reduces the on-chain average computation and storage resource costs of individual transactions. Specifically, a transaction validity proof scheme based on zero-knowledge proof is proposed to significantly reduce the on-chain computation overhead of transaction validity verification. A transaction storage scheme based on index table data compression is proposed, reducing the average on-chain storage overhead of cross-Rollup transactions. An aggregation scale balance adjustment algorithm is proposed, which obtains the optimal aggregation scale, achieving a

* 基金项目: 国家重点研发计划 (2021YFB2700301); 国家自然科学基金 (U22B2032)
收稿时间: 2024-01-30; 修改时间: 2024-04-07, 2024-06-11; 采用时间: 2024-08-02; jos 在线出版时间: 2025-01-24
CNKI 网络首发时间: 2025-01-26

balance between on-chain resource consumption and processing latency. Finally, this study validates the proposed solution through experiments. The experimental results demonstrate that under the condition of complete trustlessness, the proposed solution reduces on-chain computing and storage overheads while achieving a balance between on-chain resource consumption and processing latency. Moreover, compared to existing cross-Rollup solutions, the proposed solution exhibits good system throughput.

Key words: blockchain; Rollup; zero-knowledge proof; multi-objective optimization

区块链系统本质上是一种分布式状态机, 相较于传统中心化系统, 其交易吞吐量存在着多个数量级的巨大差距[1], 这成为区块链技术落地的严重障碍. 因此, 如何提高吞吐量成为区块链研究中的重要问题. 现有区块链吞吐量优化方式分为链上扩容与链下扩容两类. 链上扩容方案需要对原生链进行修改, 需要链上所有共识节点共同升级, 因此, 一旦采纳链上扩容策略会发生链上硬分叉的事件, 将原生链分裂成多条互不兼容的区块链, 进而带来社区的分裂、安全风险等一系列问题. 相比较而言, 链下扩容方案仅需要链下相关扩容方参与即可, 升级成本较低, 代价更小.

Rollup 是一种链下扩容方案, 相比于其他链下扩容方案如侧链方案、链下通道方案, Rollup 解决了数据可用性的问题. 在 Rollup 系统中, 定序器先将原始交易数据完整地记录在区块链上, 仅将交易执行和状态存储放到链下进行. 链下执行节点将执行后的状态根存储在区块链上, 并使用特定的验证方案验证其有效性. 这种方式使任何节点能够自行从链上读取原始交易数据, 并执行恢复数据状态, 不会对数据进行回滚. 只要存在诚实的 Rollup 节点, 系统就能够维持正常的运行.

Rollup 技术因其具备优良的数据可用性优势而备受瞩目[2], 但从实际系统发展情况来看, 单一的 Rollup 系统无法满足所有扩容需求, 随着 Rollup 内部业务需求的不断增加和复杂度的提高, 不同 Rollup 系统之间数据和资产的交换需求也变得更为迫切, 这促进了 Rollup 互操作技术 (也称为跨 Rollup 技术) 的研发.

目前, 已经明确提出的可用于跨 Rollup 的技术方案总体来说可以分为哈希时间锁和第三方服务商两种. 哈希时间锁方案仅适用于特定的交易场景, 第三方服务商方案可以被应用于通用的合约交易. 在第三方服务商方案中, 根据服务商服务对象的不同, 可以继续分为第三方代理和第三方链下验证两种.

目前为了实现通用的 Rollup 互操作方案, 采用的是引入第三方服务商的策略, 这带来了额外的信任风险. 首先, 第三方服务商需要得到所有 Rollup 节点的信任或 Rollup 内的用户的信任才能发挥作用, 这意味着需要对服务商有相对强的信任假设. 如果所有服务商串通欺诈, 将造成损失. 其次, 引入第三方服务商导致跨 Rollup 交易产生单点故障, 整个交易过程依赖于服务商的可用性和安全性. 如果服务商遭受攻击或出现故障, 整个跨 Rollup 交易流程将受到严重影响. 最后, 第三方服务商会导致网络的去中心化程度降低, 它们对整个链下 Rollup 系统网络的控制力会相对较高. 这将影响整个 Rollup 扩容生态系统的可持续性和稳定性, 少数服务商将会控制了系统生态升级的命脉.

原生区块链 (简称: 原生链) 是 Rollup 系统的信任基础, 原生链提供的交易证明可以直接被 Rollup 系统信任, 这意味着使用原生链的方案可以避免引入额外的信任假设, 从而从根本上规避了第三方服务商带来的信任假设风险. 由于原生链交易吞吐量低下, 并且计算和存储资源有限, 直接使用链下的证明和存储策略将会使得单笔交易的平均链上开销过大, 进而导致 Rollup 互操作的交易吞吐量低下.

本文提出了一种基于原生链的 Rollup 互操作方案, 称为 NativeBridge, 在无需第三方服务商参与的前提下完成 Rollup 的互操作, 从而解决了信任假设的问题. 由于原生链的资源匮乏, 因此本文要解决的难点在于减少跨 Rollup 交易在链上资源的消耗, 提高 Rollup 互操作的交易吞吐量. 本文的核心思想是通过批量集中处理交易的方式, 有效减少单笔交易的链上平均计算与存储资源消耗, 从而提高系统的吞吐量.

本文的主要贡献如下.

(1) 基于零知识证明的交易有效性证明方案: 为降低验证发送方 Rollup 中交易的链上计算资源开销, 本方案采用了批处理的思想, 将发送方 Rollup 内交易的有效性证明进行批量生成. 同时, 本方案利用零知识证明的简洁性, 将多个交易证明聚合至 1 个零知识证明中, 从而显著减少交易有效性验证的链上计算开销. 本方案最大可实现 98% 的链上计算开销优化.

(2) 基于索引表数据压缩的交易存储方案: 为了减少跨 Rollup 交易的链上存储开销, 本研究提出了一种新型跨 Rollup 交易批量处理数据结构. 该数据结构利用少数数据字段访问频率高的特点, 通过索引表将高频长字段压缩为短索引等方式, 在保证数据完整性的基础上, 降低跨 Rollup 交易的平均链上存储开销. 通过使用以太坊实际交易数据集实验, 发现本文所提方法可以显著降低跨 Rollup 交易的平均链上存储开销, 降幅达到 66.62%.

(3) 聚合规模均衡调整算法: 在本方案中, 跨 Rollup 交易的聚合规模是交易处理性能的一个关键指标. 如果聚合规模过大, 当跨 Rollup 交易的频率较低时, 批量化交易处理将增加交易等待处理时延, 从而降低方案的可用性. 为了最大化利用本方案的优势, 本文提出了一种基于非支配排序遗传算法的聚合规模均衡调整算法, 通过实时监测交易到达率等系统参数, 实现链上资源消耗与处理时延之间的平衡.

本文第 1 节对本文工作所涉及的理论和关键技术进行介绍. 第 2 节详细阐述了基于原生链的跨 Rollup 方案的完整流程及细节. 第 3 节对聚合规模均衡调整算法进行介绍. 第 4 节设计实验方案及相关评测. 第 5 节对本文的研究内容进行了总结, 并提出了未来的可优化方向.

1 相关研究

1.1 Rollup 技术整体流程

通常来说, 一个完整的 Rollup 系统分为链上智能合约和链下系统两个部分, 如图 1 中 Rollup 的交易过程总览所示. 链上智能合约仅用于存储原始交易数据、Rollup 的默克尔树根, 以及配备对默克尔树根的有效性检验方式. 链下部分通常由定序器、执行器和验证器 3 部分组成. 一个通用的 Rollup 交易流程包括以下步骤.

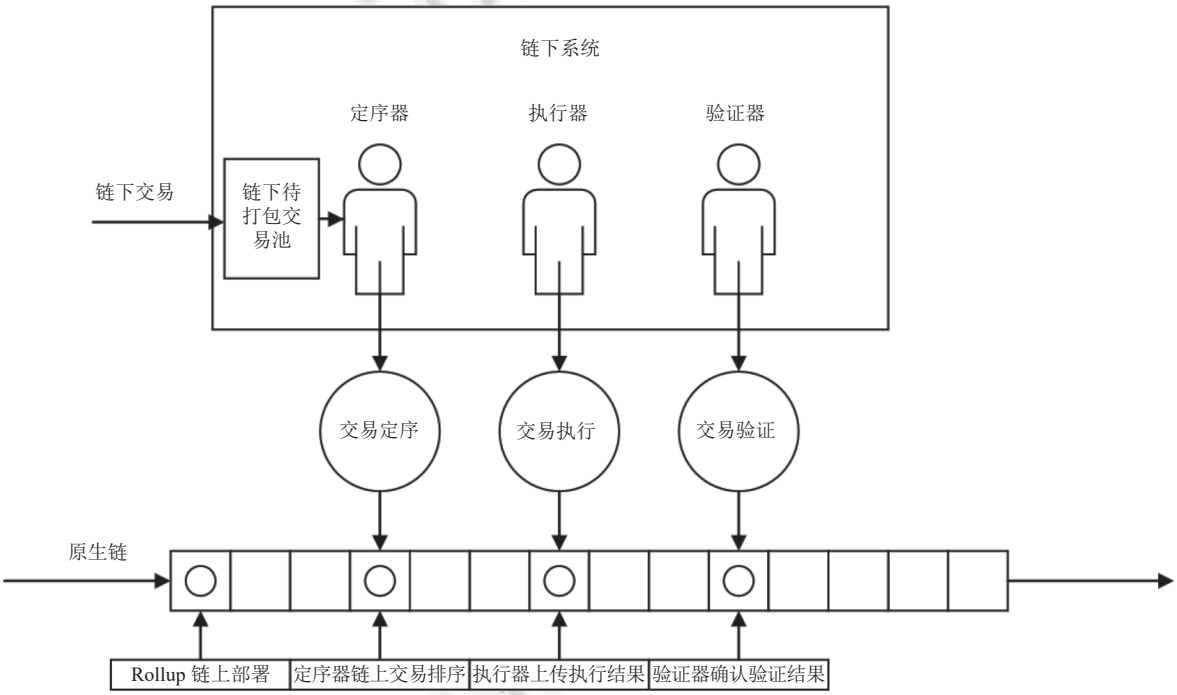


图 1 Rollup 的交易过程总览

(1) 用户发送交易后, 交易进入到链下待打包交易池中.

(2) 根据特定策略选择本轮的定序器, 定序器从链下待打包交易池中获取交易, 将交易发送到链上.

(3) 执行器通过监听交易上链信息, 当监听到新增交易后读取并执行, 将执行后的 Rollup 默克尔树根记录在链上, 以便其他验证者验证状态.

(4) 验证者通过一定的验证方式验证执行器的执行结果是否正确, 并为默克尔树根提供有效性证明以保证其可信性.

Rollup 将原始交易排序后上传至原生链, 然后通过链下执行程序完成交易, 上传默克尔树根, 如图 2 所示, 最后, 验证交易的有效性. 通过这种方式, Rollup 保证了不同节点上待执行的原始交易的一致性, 同时, 原生链的存储机制也保证了待执行原始交易的完整性和不可篡改性. 而且因为原始交易存储在原生链中, 所以每一个链下节点执行的交易都是可验证的, 交易完成之后的状态也是可验证的, 只需要对有效状态最终认可即可. 最后, 任何新的节点无需寻找合适的对象, 就可以通过自己的方式, 从原生链中读出原始的交易并执行.

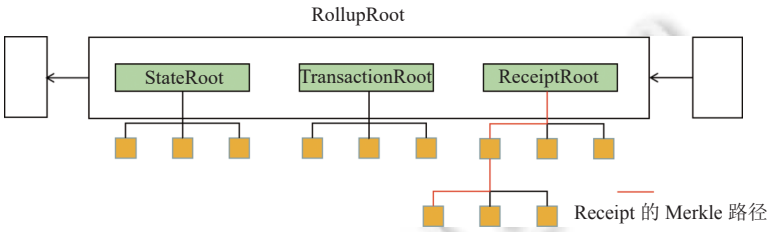


图 2 Rollup 的默克尔树根

1.2 Rollup 互操作技术

在跨 Rollup 系统中, 一笔完整跨 Rollup 交易流程由发送方 Rollup 内执行的发送方交易和接收方 Rollup 内执行的接收方交易两部分组成. 跨 Rollup 系统要解决的目标是跨 Rollup 信息传输、跨 Rollup 数据验证. 由于参与的 Rollup 系统之间缺乏直接的信任关系, 跨 Rollup 系统需要设计一种机制来保证各个参与方互相信任交易已经成功执行了, 以及没有节点在传输过程中篡改接收方交易, 避免执行错误交易给 Rollup 系统带来状态错误.

现阶段明确提出可用于 Rollup 互操作的技术方案可以分为哈希时间锁和第三方服务商的方案. 在第三方服务商中, 根据服务商服务对象的不同, 可以继续分为第三方节点代理和第三方节点链下验证两种.

(1) 哈希时间锁

哈希时间锁[3]是一种多方交易机制, 通过使用密码学哈希函数和时间限制, 可以使得两个或多个交易对手在互相不信任的情况下进行安全的交易. 虽然哈希时间锁可以用于 Rollup 互操作技术, 但它只适用于特殊的资产交换场景, 无法满足通用的合约交易需求. 同时, 其基于时间的限制也可能会造成交易的不便和不确定性. 这使得其使用场景非常受限, 不适应于更广阔的应用空间.

(2) 第三方节点代理

以 Hop[4]方案为代表的第三方代理方案, 由服务商直接为用户提供交易服务. 第三方服务商同时存在于发送方 Rollup 和接收方 Rollup 中, 并取得了用户的交易代理权. 当用户在发送方 Rollup 中发起交易后, 第三方服务商会验证交易执行成功后, 代替用户去接收方 Rollup 中执行交易. 对于接收方 Rollup 而言, 只需要确认第三方服务商获得了用户的代理权, 就可以对交易进行可信执行. 最终, 第三方节点将代理执行的结果告知用户即可. Hop 方案无需 Rollup 间的额外信任条件, 但是用户需要信任第三方服务商的诚实和有效, 以确保第三方服务 商已发送代理交易并成功执行.

(3) 第三方节点链下验证

以 Orbiter Bridge[5]为代表的第三方节点链下验证方案是指第第三方服务提供商通过 SPV (simplified payment verification) 等验证方式直接为交易提供有效性证明. 在接收方 Rollup 节点收到交易后, 只需向服务商确认原交易是否有效. 一旦服务商提供了有效的交易验证, 该交易便可被打包进接收方 Rollup 的执行交易中并执行. 在这种方案下, 接收方 Rollup 需要信任服务商之间的安全架构, 以保证至少有服务商愿意提供服务, 以及服务商未联合为恶意交易做伪证. 这种设计利用服务商之间的互相监管机制, 为实现跨 Rollup 交易提供了一定的安全性和可靠性. 这种方案也是存在信任假设的, 信任假设降低为至少存在一个诚实的服务商为整个系统提

供真实有效的工作.

1.3 区块链链上资源现状分析

以太坊 (Ethereum) 是目前链下扩容生态最好的公链, 相比于比特币等其他区块链, 以太坊提供了允许开发者构建分布式应用程序 (DApps), 更强的灵活性和可编程能力, 使得开发者可以更容易地实现自己的创意和想法. 以太坊的平均交易吞吐量仅有每秒十几笔, 其庞大的体量与羸弱的处理性能的冲突造就了以太坊拥有最丰富链下扩容市场和多样化的平台发展, 链下场景的丰富也增加了链下 Rollup 间互操作的需求. 因此, 本文选择以太坊作为原生链来研究区块链跨 Rollup 技术.

(1) 链上计算负载

对于以太坊为代表的公链而言, 链上的计算负载能力是有限的, 单位时间内能提供的计算资源受到系统设置的约束. 其约束主要来源于两个方面, 出块的间隔的限制以及单个区块的计算开销是有上限的.

由于区块的共识间隔严重受到限制, 因此在单个区块计算能力存在上限时, 无法通过加速出块的方式提升单位时间内的计算能力. 以太坊在设置出块间隔时, 考虑到其在全球范围内的部署和运行, 需要充分广播区块. 因此, 出块间隔不宜过短, 否则会增加系统分叉率, 影响整个网络的安全性.

单个区块内交易计算能力也存在上限, 不能无限制提升单个区块内的计算开销. 其原因是以太坊引入了 Gas 费的概念用于衡量链上资源消耗, 同时为区块设置了 Gas 的上限. 单个区块内所有交易的 Gas 总开销必须小于等于区块的 Gas 上限, 被称为 GasLimit, 否则交易将被拒绝. 直接调整 GasLimit 可以直接增大以太坊的吞吐量上限, 但是该方案属于对链上参数的修改是链上扩容的一部分. 本文的研究属于链下扩容策略范畴, 因此不能对以太坊本身的共识机制, 区块大小等进行调整.

(2) 链上存储负载

跨 Rollup 交易实质上是在链下执行计算, 对于链上而言, 对跨 Rollup 交易进行存储是为了让接收方 Rollup 可以获得待执行交易. 因此, 可以将同一批次上传的所有跨 Rollup 交易集合看成一个二进制文件, 在链上只做二进制文件存储相关的功能, 不需要对交易进行计算. 这种形式的存储可以采用以太坊 EIP-4844 协议提出的 blob 交易格式作为链上存储的方式. EIP-4844[6]是针对存储的升级, 它针对非链上计算的二进制文件可以提供更轻量级的存储. 但是对于 EIP-4844 协议而言, 区块能够提供的存储负载是存在上限的, 这也会影响系统整体性能.

(3) 交易访问规律

对于以太坊上的交易访问呈现了显著的长尾分布的形态, 即少数账户造成了绝大部分的状态访问. 在过去的研究中表明[7], 在分析高度在 5500000–6000000, 从 2018 年 4 月 24 日–2018 年 7 月 20 日期间的历史区块, 可以发现 72% 的交易被发往 1% 的账户地址, 66% 的交易是由 1% 的账户地址发起.

本文通过进一步研究发现, 绝大部分地址会多次出现, 仅出现一次的地址数量较少. 数据集选取自 2022 年 4 月 1 日–2022 年 9 月 25 日期间的以太坊非同质化代币 (NFT) 类交易中的样本, 为了评估交易集中的数据, 本文研究了交易地址仅出现一次的频率. 如后文图 3 所示, 仅出现一次的地址数量 (以红色柱状表示) 并未明显超过其他出现次数的地址数量. 相反, 绝大部分地址的出现次数呈现多样化. 将仅出现过一次的地址数量除以总地址数量, 其占比仅为 6.04%. 因此, 在所有地址中, 仅出现过一次的地址数量仍然处于较小的比例.

而在 Rollup 内的交易本身就是以太坊交易在链下的扩容体现, 其面临的业务场景应当是相似的. 因此, 跨 Rollup 交易访问统计规律应当会产生类似的长尾分布规律.

1.4 零知识证明技术

零知识证明算法[8]是指证明者向验证者证明某个陈述的正确性, 而不需要透露除该陈述正确外的任何信息的一种密码学算法. 零知识证明算法主要分为交互式和非交互式两类.

(1) 零知识证明介绍

交互式证明需要证明者和验证者之间进行交互, 并根据验证者的请求提供陈述的证据, 可以通过多次陈述多次证明的方式来加强安全性, 但是这种证明方式通常需要较长的时间和复杂的计算.

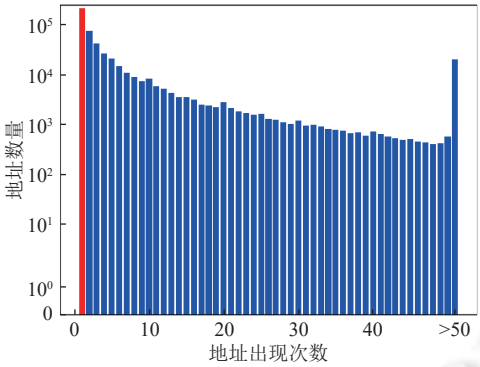


图 3 交易地址分布情况

在非交互式证明中[9], 证明者可以通过单向函数的方式生成证据, 使得验证者可以验证陈述的正确性. 这种证明方式通常对证明者的计算要求高, 而验证者的计算负载小. 在实际的使用中, 在证明者与验证者的计算能力存在较大差异的场景下, 非交互式的证明策略常常受到更多的青睐.

在非交互零知识证明中现阶段最有效的算法是 zk-SNARKs[10], 它除了具备基本的零知识特性外, 还具备简洁性的特性. 简洁性的特性指的是, 与实际计算的长度相比, 生成的零知识证据通常很小, 只需要花费更少的计算开销就可以快速证明原始计算的有效性. 利用简洁性的特性, 可以将验证者本身需要承担的大量计算开销转移到证明生成者上, 进而降低验证者所需要的计算资源. 本文采用的零知识证明算法是非交互式证明的算法, 在后续表述中如不强调说明零知识证明算法均是非交互式的.

(2) 零知识证明 Rollup (zkRollup)

ZKsync[11]是一种 zkRollup 的链下扩容方案, 它的最大特点在于采用了零知识证明来验证原始交易执行的默克尔树根的有效性. 零知识证明具有完备性特征, 即只有当原始交易与生成的默克尔树根匹配时才能生成有效的零知识证明. 因此, 验证者只需生成对应交易的有效零知识证明即可证明该批交易的有效性. 在 ZKsync 中, 执行者和验证者成为同一个节点, 通常称为 Validator. 它们在执行交易生成默克尔树根的同时, 也生成零知识证明提交到链上, 以此证明默克尔树根的有效性.

在 ZKsync 中, 还有一批巡查者 (guardians) 用于监管验证者的行为. 需要说明的是验证者无法通过造假默克尔树根来修改底层状态, 但是有可能被 DDoS 攻击, 或者由于自身主观的恶意行为而拒绝提供验证服务. 此时巡查者需要巡查鉴别验证者是否被 DDoS 攻击或者存在恶意行为故意不提供服务的行为, 通过监管巡查替换掉一些不在工作状态的验证者, 对于暂时无法提供服务的验证者使用黑名单机制在用户群体间广播确认.

对于应用场景来说, ZKsync 能够迁移一定的链上智能合约代码, 但不支持通用的 EVM 兼容的智能合约, 能满足常见的合约需求, 但合约不完全具备图灵完备性, 只能采用特殊的语言进行特殊化编译.

2 基于原生链的跨 Rollup 方案

本节研究基于原生链的跨 Rollup 方案. 首先, 本文针对单向互操作交易类型进行了设计, 该交易类型是所有后续复杂的交易类型的基础. 其次描述了整体设计思路以及基于原生链的跨 Rollup 方案的整体流程. 之后针对验证方案和数据存储方案, 分别展开论述. 对于链上的计算存储开销, 使用基于零知识证明的批量化交易验证方案, 利用零知识证明的验证简洁性的特性减少链上计算资源的开销. 对于链上存储资源的开销, 使用基于索引表数据压缩的交易存储方案, 利用少部分数据出现的高频特性, 减少单笔交易的平均链上存储开销.

2.1 系统和模型假设

本节对跨 Rollup 交易进行建模, 针对单向互操作的跨 Rollup 交易类型设计了方案, 并分析了未来可在此基础上扩展的方案.

2.1.1 系统模型

本文提出的跨 Rollup 方案 $BR(Rollup_{\text{sender}} \rightarrow Rollup_{\text{receiver}})$ 是针对以下的场景的假设. 针对一条原生链 *Blockchain*, 它具有各自独立的链下扩容方案发送方 $Rollup_{\text{sender}}$ 和接收方 $Rollup_{\text{receiver}}$, 并且智能合约 $Contract_{\text{sender}}$ 和 $Contract_{\text{receiver}}$ 分别部署在两个 Rollup 上. 一笔完整的交易会有两阶段的生命形态, 包括发送方交易 TX_{sender} 和接收方交易 TX_{receiver}. 一个通常的流程是用户将交易发送至发送方 $Rollup_{\text{sender}}$, 在 $Rollup_{\text{sender}}$ 中执行发送方交易 TX_{sender} 后产生二阶段的接收方交易 TX_{receiver} 并发送到 $Rollup_{\text{receiver}}$ 中执行. 同时, 接收方交易 TX_{receiver} 需要作为发送方交易 TX_{sender} 执行后所抛出的事件存储在交易收据树 *ReceiptTree* 中, 如图 2 所示, 交易收据树 *ReceiptTree* 是默克尔树根的一部分.

2.1.2 系统假设

本文的系统对 Rollup 做出了一些假设, 这些假设本身也是常见的 Rollup 假设. 本文的互操作方案仅限于 Rollup 方案, Rollup 是区块链 Blockchain 的一个扩容策略, 它需要满足的条件如下.

(1) 交易可读性: Rollup 内执行的原始交易 TX_{sender} 被存储在链上节点, 其他节点可以随时读取该笔交易, 且该交易必然已经被执行了, 成功与否会被记录在交易收据树中.

(2) 状态根有效性: Rollup 内执行成功后的默克尔树状态根 MerkleRoot 被上传到链上, 并由 Rollup 的自身的验证机制确保其有效性以及正确性. 该状态根存储在原生链上, 其他节点可以确认该根的结果有效性.

(3) 交易实时性: Rollup 内的节点对于所有可执行的合法交易, 会及时将交易放入待打包交易池中, 而不会因为交易费等其他因素拒绝执行交易.

(4) 交易通用性: Rollup 内能够执行的交易是具备一定的通用性, 能在智能合约中执行.

2.1.3 交易类型

需要说明的是本文实现的 Rollup 方案是最基础的单向的互操作交易类型, 后续复杂的交易类型都可以在本文之上扩展.

对于完整的交易事务性而言, 系统可以通过增加事务管理器的方式增加交易事务性. 本文的跨 Rollup 方案仅能保证在 $Rollup_{\text{sender}}$ 内执行成功的交易, $Rollup_{\text{receiver}}$ 可获得交易并执行; 在 $Rollup_{\text{sender}}$ 内执行失败的交易, $Rollup_{\text{receiver}}$ 内不会执行. 而不保证在 $Rollup_{\text{sender}}$ 执行成功, 但 $Rollup_{\text{receiver}}$ 执行失败的情况下, $Rollup_{\text{sender}}$ 内的交易将会回滚. 这种强事务性保障需求, 需要在本文的机制基础上进一步加强. 这可以选择一种完善的事务保障机制, 在 $Rollup_{\text{receiver}}$ 内执行失败后, 再发起一笔 $Rollup_{\text{sender}}$ 内的对应回滚交易来保障其事务性. 通过两笔单向跨 Rollup 交易以及对应的事务管理器来解决交易事务性的问题.

对于多 Rollup 的互操作的需求, 系统可以通过增加多交易管理器的方式增加多 Rollup 的互操作方案. 本文讨论的交易范畴仅在两个 Rollup 上展开, 但事实的业务需求会需要多个 Rollup 间的互操作. 多个 Rollup 间的交易, 可以拆分成多笔两两之间的 Rollup 交易. 通过多交易管理器管理多笔单向的 Rollup 交易来共同完成多 Rollup 间的互操作方案. 本文可以成为未来多 Rollup 间的互操作方案的基础.

2.2 整体设计思路

本文提出了一种基于原生链的跨 Rollup 方案, 称为 NativeBridge, 它所涉及的跨 Rollup 方案的主要工作分为两个部分, 第 1 部分为发送方交易 TX_{sender} 提供有效性验证, 第 2 部分为接收方 Rollup 同步接收方交易 TX_{receiver}. 而交易验证和交易同步在链下基于服务商的方案中已经有相关方案.

由于原生链每个区块的计算资源与存储资源有限, 详见第 1.3 节的分析, 将每笔跨 Rollup 的链下验证方案和存储方案直接迁移至原生链, 会消耗大量的原生链计算与存储资源. 因此, 本文提出的整体解决思路是采用批量化的处理方法, 以均摊单笔交易所需的链上资源消耗, 从而使单个区块能够承载更多交易, 提升系统吞吐量.

对于链上计算资源受限的问题, 本文采用的策略是利用 zk-SNARKs 类零知识证明所具有的简洁性证明特性, 将多笔交易的验证通过一个简单高效的零知识证明验证策略批量证明, 对于原生链只需要投入较少的计算资源就可以验证一批交易是否成功, 减少单笔交易的平均开销. 具体的方案介绍详见第 2.3 节.

根据第 1.3 节的分析, 可以得到观察到现象, 大部分的交易由少部分账户发出, 大部分账户上有多个交易. 由此可见, 事实上交易记录的地址等字符串实际上是有较高的重复性的. 因此本文提出的策略是基于索引表的思想, 提出了一种批量化跨 Rollup 交易的数据压缩方法. 整体思想是使用短索引代替长字段等形式, 有效地减少了交易大小, 缓解了链上存储资源受限的问题, 具体的方案介绍详见第 2.4 节.

接下来本文从整体上描述基于原生链的跨 Rollup 的基本流程, 分为 5 个部分: 交易发送、聚合交易、链上存储、链上验证、对端执行. 为了便于阐述, 本文只关注交易的一个方向流动, 即仅从 $Rollup_{\text{sender}}$ 到 $Rollup_{\text{receiver}}$ 的交易流动, 而相反方向的操作是对称的, 交易整体流程如图 4 所示.

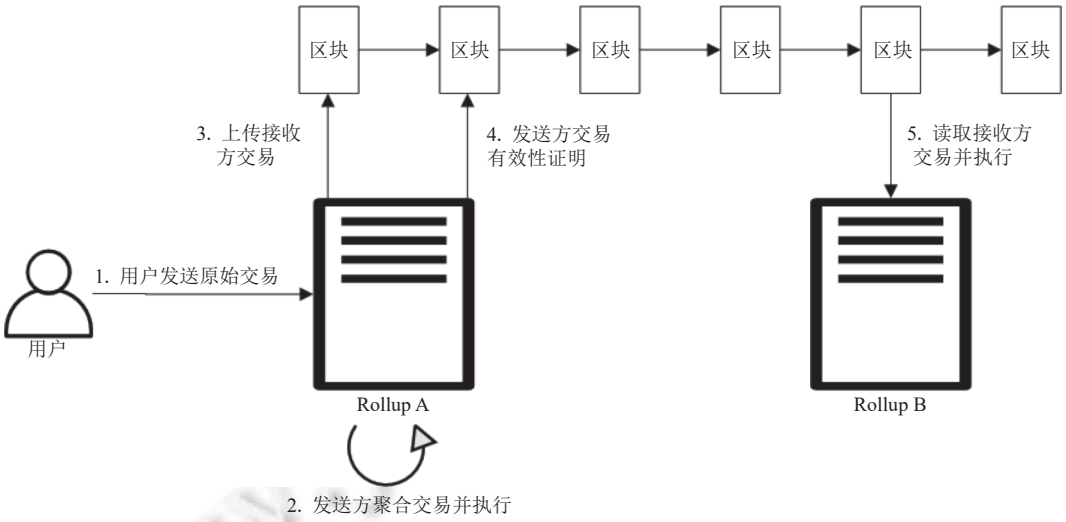


图 4 跨 Rollup 交易的整体流程图

(1) 交易发送: 用户发送单笔交易 TX_{sender} 至智能合约 $Contract_{\text{sender}}$, 通过智能合约的执行, 产生了一笔需要在 $Contract_{\text{receiver}}$ 内执行接收方交易 TX_{receiver}.

(2) 聚合交易: 发送方 $Rollup_{\text{sender}}$ 等待聚合一批交易批量处理.

(3) 链上存储: 发送方 $Rollup_{\text{sender}}$ 将接收方交易 TX_{receiver} 存储在原生链上, 以便 $Rollup_{\text{receiver}}$ 读取交易.

(4) 链上验证: 发送方 $Rollup_{\text{sender}}$ 使用零知识证明策略为 TX_{sender} 的交易有效性提供证明.

(5) 对端执行: $Rollup_{\text{receiver}}$ 从原生链读取验证通过的 TX_{receiver} 交易, 并在 $Contract_{\text{receiver}}$ 中执行 TX_{receiver};

2.3 基于零知识证明的跨 Rollup 验证方案

2.3.1 交易有效验证目标

本方案整体上要完成交易有效性验证, 同时还需要在验证的过程中优化区块链交易验证过程中的资源利用效率. 对于交易有效性的证明包括两部分.

(1) 发送方交易 TX_{sender} 是否执行成功.

(2) 存储在原生链上的接收方交易 TX_{receiver} 与发送方交易执行生成的接收方交易 TX_{receiver} 一致.

原始交易 TX_{sender} 的执行结果存储在 $Rollup_{\text{sender}}$ 的交易收据树 *ReceiptTree* 中. 证明的目标是存在一个有效的收据, 该收据能够证明 TX_{sender} 交易执行成功的状态, 并且交易抛出的事件与链上存储的交易 TX_{receiver} 一致, 以及该收据状态是链上已证实有效的交易收据树的一部分. 数学形式化表达如公式 (1) 所示, 其中 $Receipt(TX_{\text{sender}})$ 指的是 TX_{sender} 的交易收据状态, $S(TX_{\text{sender}})$ 是 TX_{sender} 的交易执行状态, 通常为 successful 或者 revert, revert 为交易失败回滚, $Event(TX_{\text{sender}})$ 是指的是 TX_{sender} 执行后抛出的事件, TX_{receiver} 为链上存储的接收方交易, *ReceiptTree* 是链上验证有效的交易收据树.

$$\begin{cases} \exists Receipt(TX_{\text{sender}}) \\ \text{s.t.} \begin{cases} S(TX_{\text{sender}}) = \text{successful} \\ Event(TX_{\text{sender}}) = TX_{\text{receiver}} \\ S(TX_{\text{sender}}) \in Receipt(TX_{\text{sender}}) \\ Event(TX_{\text{sender}}) \in Receipt(TX_{\text{sender}}) \\ Receipt(TX_{\text{sender}}) \in ReceiptTree \end{cases} \end{cases} \tag{1}$$

2.3.2 交易验证策略

本方案除了要完成上述验证目标, 还需要在验证上述目标的过程中优化区块链交易验证过程中的资源利用效率. 为此, 本文采用了批量零知识证明策略. 通过这种策略, 可以将多个交易批量验证, 从而减少链上资源的占用和交易验证的时间成本, 提高了整个系统的交易处理效率.

本方案使用的是零知识证明算法中验证开销小的简洁性. 通常而言, 零知识证明验证证明的开销随待验证函数的计算复杂度是对数增加的, 且增加效率低, 在一定范围内甚至可以近似看成是常数级别的复杂度.

在传统的交易验证过程中, 每笔交易需要分别进行 SPV 证明的生成和验证, 这会导致链上计算成本是随交易量线性增加的. 相比之下, 批量零知识证明策略可以将多个 SPV 证明合并为一个零知识证明, 合入的交易量越多, 零知识证明的总开销也不会显著增加, 从而大幅降低单笔交易平均计算成本. 此外, 在批量零知识证明策略, 由于不同交易的默克尔树的证明过程是相互独立的, 因此可以通过优化算法实现零知识证明的并行化, 提高证明的生成效率, 较好的缓解了零知识证明生成过程过长的问题.

在本方案中, 链下的 Rollup 节点为零知识证明的证明者, 它需要证明链下提供的接收方交易集合 TX_{receiver} 是真实有效的. 链上的智能合约为零知识证明的验证者, 它通过简单的计算即可验证其链下提供信息的正确性. 当链上智能合约能够证明交易的有效性后, 接收方 Rollup 则可以读取交易并执行.

本文采取的零知识策略整体属于 zk-SNARKs 类的零知识证明方案体系. 完整的 zk-SNARKs 零知识证明方案通常分为证明初始化阶段, 证明生成阶段和证明验证阶段. 在证明初始化阶段, 证明方和验证方需要提前设置相关密码学参数的公共参考串 (CRS) 的形式实现协议的非交互性. 系统参与方需要在生成证明前生成一个随机字符串, 该字符串只需要生成过程不一直被人为控制便可以保障零知识证明的有效性. 保障随机字符串的生成安全, 现已有多种研究, 可以参考文献 [12,13] 等, 定期生成和更换 CRS 保障零知识证明的公共参考串.

零知识证明生成阶段具体流程如图 5 所示, 在本方案中明文数据指的是交易收据树根集合 *ReceiptRootList*, 最终生成的接收方交易默克尔树根 $TX_{\text{receiver}}TreeRoot$ 相关信息, 具体的明文输入详见表 1. 其中交易收据树根 *Receipt-Root* 是在链上有效性验证通过的交易收据树根. 由于采用批量化验证的特点, 可能多笔交易不在同一个交易收据树根 *ReceiptRoot* 中, 因此将该批交易涉及的所有交易收据树根的集合共同形成一个交易收据树根数组 *ReceiptRoot-List*. 由于产生了多笔接收方交易, 系统将多笔接收方交易共同组织成一棵默克尔树, 其树根就是接收方交易默克尔树根 $TX_{\text{receiver}}TreeRoot$.

密文数据在本方案中指的是不公开上传的数据, 具体的密文输入详见表 2. $TX_{\text{sender}}List$ 是所有待验证的发送方交易集合; $TX_{\text{receiver}}List$ 是所有上传到链上存储的接收方交易集合; *ReceiptList* 是所有发送方交易对应的收据信息集合; 对于每一个待验证的发送方交易收据, 都会有一条 SPV 证明默克尔路径来证明该收据存在于交易收据树中, *ReceiptMerklePathList* 是所有交易收据的默克尔路径证明集合.

接下来描述具体的验证函数部分, 大致流程为:

(1) 将明文输入和密文输入后, 逐一判断输入的基本合法性;

(2) 检索每一个收据 *Receipt* 的合法性;

(3) 检索每一个收据是否存在于交易收据树中 *ReceiptTreeRootList*;

(4) 最后验证跨 Rollup 交易 TX_{receiver} 生成跨 Rollup 交易树是否为明文输入的 $TX_{\text{receiver}}TreeRoot$.

零知识证明验证策略是验证上述零知识证明生成过程有效的方案, 由零知识证明的验证者完成. 该方案需要将明文数据输入和零知识证明生成策略中的零知识证明证据输入到验证电路中, 验证电路会返回该证明证据是否

由该批明文数据生成, 详见图 6. 由于零知识证明具有完好性的特性, 即如果上述零知识生成过程不通过, 证明者无法伪造出一个有效的证明使得其通过验证方的检. 因此只要证明者提供了一份有效的证明和对应的明文输入, 那么说明证明者拥有了有效的密文输入, 且对密文输入的检查结果都是合法的. 在进行零知识证明验证时, 只需要校验明文输入的有效性, 等同于密文输入的有效性以及整个验证函数执行的有效性. 通过这种方式, 验证者可以不必获取完整的密文输入, 也不必完整的执行整个验证函数流程.

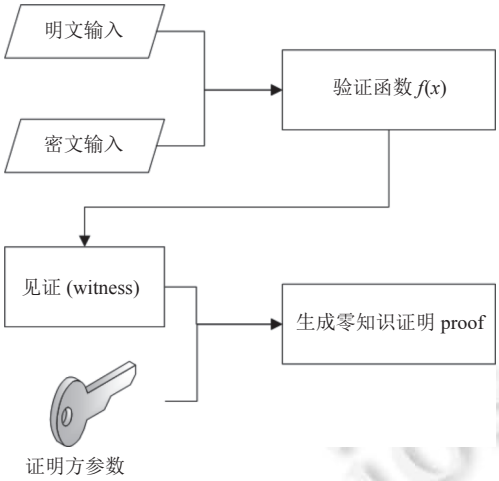


图 5 零知识证明的生成阶段

表 1 零知识证明的明文数据

明文输入字段	具体含义
ReceiptRootList	原始交易的交易收据树根列表
$TX_{\text{receiver}}TreeRoot$	该批次证明中所有接收方交易组合成默克尔树根

表 2 零知识证明的密文数据

密文输入字段	具体含义
$TX_{\text{sender}}List$	所有原始交易列表
$TX_{\text{receiver}}List$	所有跨Rollup交易列表
ReceiptList	原始交易的交易收据列表
ReceiptMerklePathList	原始交易收据的默克尔路径证明

图 6 零知识证明的验证阶段

2.3.3 零知识证明算法并行化

本文具体所采用的零知识证明算法是基于 DeVirgo[14]和 Groth16[10]的并行化零知识证明方案[15]. 选择并行化零知识证明的原因是, 跨 Rollup 交易验证中所实现的零知识证明的验证函数具有高度可并行的部分. 传统的零知识证明生成过程通常是单线程执行的, 而并行化零知识证明可以利用多线程并行化生成证明过程, 减少证明生成的时间. 通过并行化证明的方式, 可以提高单次零知识证明的生成速度, 从而使系统能够处理更大规模的证明任务. 这样可以增加单次零知识证明的交易量, 进一步降低单笔交易的平均链上计算成本, 并提高系统的性能和效率. 并行化零知识证明的验证方法如图 7 所示.

并行化证明策略的核心思想是选择一种零知识证明方式作为并行证明策略, 并选择一种零知识证明算法生成最终的零知识证明 proof, 通过证明并行化策略下的零知识证明的证据的有效性来证明整体证明的有效性. 具体而言, 首先将原始待证明函数拆分成多个可并行化的证明函数. 在本方案中, 可并行的函数部分指的是各笔证明不冲突的待验证的发送方交易, 本文通过对该部分进行并行处理, 加速零知识证明的生成过程. 在验证交易不会发生冲突时, 由于不同交易的验证之间没有依赖关系, 因此各自独立生成零知识证明的过程是可行的. 接下来, 针对分拆后的各并行化证明函数选择一种并行零知识证明算法. 各个函数独立使用并行零知识证明算法, 得到多个辅助证明模块. 最后, 在最终的零知识验证算法中, 将各并行部分的零知识证明的证据以及原始证明过程中的串行部分合并为新的最终证明函数.

从整体上看, 零知识证明的目标是证明该批交易都能通过验证函数有效性检查. 验证并行化部分的辅助模块的证据, 实际上相当于使用并行化零知识证明算法证明单笔交易的有效性. 因此, 只要每一笔交易的零知识证据的验证有效, 就相当于单笔交易原始验证函数有效. 而最终的验证函数则由所有并行化零知识证明证据的验证以及原始串行部分组成. 因此, 若最终可以生成零知识证明的证据, 那么便是证明每一笔交易的原始验证函数有效, 进而可以推导出整体证明函数的有效性.

Virgo 是目前已知的零知识证明中, 少数不需要提前进行可信的初始化设置就可以达到抗量子安全级别的零

知识证明算法, 同时它支持大规模的零知识证明计算, 在大规模场景下具有最好的生成证明的效率, 其中大规模指的是证明函数的计算复杂度规模大. DeVirgo 是 Virgo 方案的分布式版本, 它可以将原本生成证明的电路拆成多个部分可以并行进行零知识证明生成. 但直接使用 Virgo 的问题是最终证据的简洁性不够, 验证开销不小. Groth16 是另一种的零知识证明生成策略, 它的优势在于它最终生成的整体零知识证明 proof 大小是较小的, 因此它的验证函数的计算量也较小. 因此 DeVirgo 用于并行化部分的证明, 而 Groth16 用于最终部分的证明生成. 目前 DeVirgo 和 Groth16 均已开源, 可以在文献 [16,17] 中找到相应的项目源码. 文献 [15] 指出, DeVirgo+Groth16 这个算法组合被认为在可并行化的递归零知识证明中整体生成效率最高. 若不采用该方案, 选择文献 [9,13,18−22] 等方案均可达成可用目标, 但整体效率均有所下降.

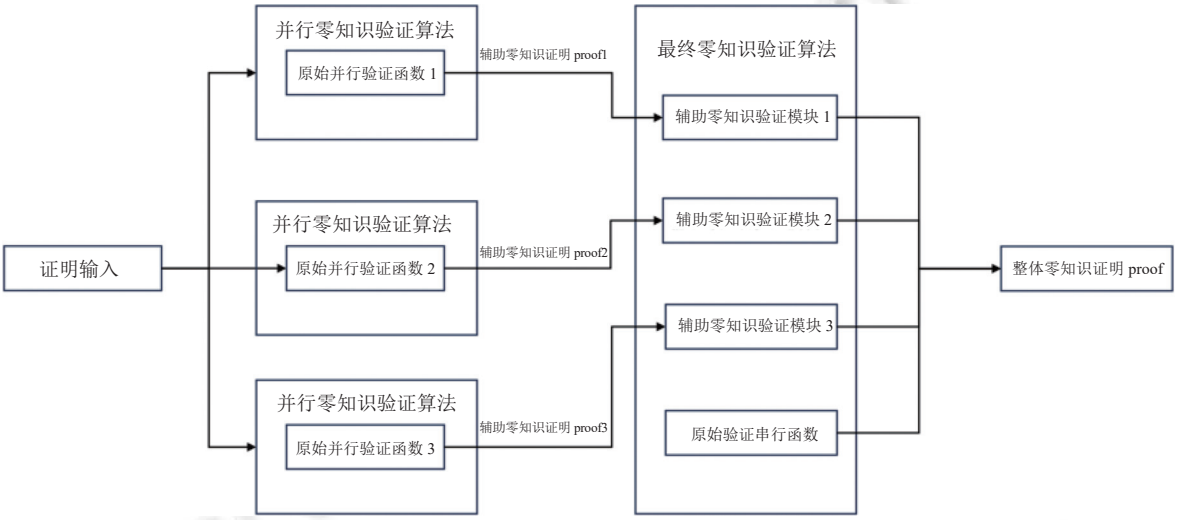


图 7 并行化零知识证明的验证方法

2.4 基于索引表数据压缩的交易存储方案

2.4.1 数据压缩策略概述

接收方 Rollup 在链上完成验证过程后, 需要执行接收方交易 TX_{receiver}. 由于采用零知识证明的方案, 链上最终只得到了零知识证明的 proof 和仅包含原始交易收据树根和接收方交易树根的明文输入数据, 而接收方 $Rollup_{\text{receiver}}$ 未获得接收方交易 TX_{receiver} 的完整原始数据. 不将整批接收方交易 TX_{receiver} 直接作为明文输入的原因是零知识证明需要进入合约层执行计算并存储, 而通常 TX_{receiver} 的交易存储量并不小, 直接在合约中存储会增加交易状态树的体积, 给链上存储带来负担.

为应对这一挑战, 本节讨论的是如何优化原生链节点的存储消耗, 减少原生链节点的负载开销, 使得系统更有效可靠. 本文提出了一套跨 Rollup 交易的存储方案. 对于交易的链上存储, 使用特殊的 EIP-4844 协议的 blob (binary large object) 方式来存储待传输的接收方交易 TX_{receiver}, 可以将无计算的二进制文件交由特殊的存储节点存储, 避免所有全节点的存储开销, 可详见第 1.3 节的描述. 对于实际要存储的数据, 引入了新的一种数据压缩策略来优化存储, 包括两个主要部分, 分别是交易数据结构压缩和索引表压缩.

(1) 交易数据结构压缩: 为了解决交易存储量大的问题, 本文采用了一种交易存储压缩的数据结构, 该技术精简了以太坊交易信息, 用最简数据结构来代替完整的交易结构. 具体而言, 本文仅保存交易信息中的必要字段, 只包含交易双方地址、交易金额及合约数据, 而不保存不必要的信息, 如交易备注、交易 Gaslimit 费用等. 同时对于批量交易, 合并相同的字段减少平均单笔交易的开销.

(2) 索引表构建: 面对交易的部分字段高频出现的特点, 本文在链上构建了一个索引表, 将较长字段及其对应

的较短索引存储在链上索引表中, 而接收方交易 $TX_{receiver}$ 内的字段在链上仅存储短索引, 而不保存完整的数据. 当接收方 Rollup 接收交易后, 可以自行读取索引表解析还原完整原文. 这样一来, 本文就能够在不占用过多链上资源的前提下, 实现便于交易的同步.

2.4.2 交易数据结构压缩

交易数据结构压缩是一种优化接收方交易 $TX_{receiver}$ 传输的方法, 其核心思路在于仅保留最简化的数据格式, 而非采用标准的以太坊交易数据结构. 这种方法能够显著减小跨 Rollup 交易数据的存储需求和传输负担, 从而提高整体的区块链性能和扩展性.

为实现交易数据结构的压缩, 本文剔除了交易中无效或冗余的交易字段. 这些字段包括但不限于交易备注, 交易 Gaslimit 上限等. 这些字段对于接收方 Rollup 而言并不会直接产生影响. 仅需要记录接收方 Rollup 需要执行的字段即可. 其次合并了相关字段, 由于同一批交易的发送方 Rollup 和接收方的 Rollup 都是固定的, 因此若每一笔交易 $TX_{receiver}$ 均保留发送方 Rollup 和接收方 Rollup 字段会产生浪费. 对于交易手续费用, 不需要每一笔交易单独记录, 而将整批交易总共计量一个总费用支出. 整体上传的批量跨 Rollup 交易数据结构如表 3 所示. 它包含整批次的交易唯一标识, 发送方 Rollup 的编号, 接收方 Rollup 的编号, 以及单笔跨 Rollup 交易的集合.

表 3 批量跨 Rollup 交易数据结构

密文输入字段	具体含义	类型
nonce	该批交易的唯一标识	bytes
$Rollup_{sender}$	发送方Rollup序号	bytes
$Rollup_{receiver}$	接收方Rollup序号	bytes
GasTotal	所有交易的总手续费开销	int
$TX_{receiver}List$	跨Rollup交易集合	$List < Transaction_{simple} >$

在批量跨 Rollup 交易数据结构中包含单笔跨 Rollup 交易数据结构, 其类型为 $Transaction_{simple}$, 单笔交易的类型结构如表 4 所示. 按照以太坊的交易格式只包含不可缺省的字段, 包括交易发送方, 交易接收方, 转移金额, 调用合约的二进制数据以及后续为字段进一步压缩所增加的压缩协议字段.

表 4 单笔跨 Rollup 交易数据结构

字段	具体含义	类型
$Address_{sender}$	发送方用户地址	Address
$Address_{receiver}$	接收方用户地址	Address
Value	发送金额	int
Calldata	调用合约二进制数据	bytes
CompressionType	数据压缩协议	bytes

2.4.3 基于索引表数据压缩

在现有实际交易的情况分析来看, 交易之间会频繁调用一些字段, 而这些字段为了保证数据的完整性通常会使用更长的存储. 例如在调用合约过程中, $Address_{sender}$ 为合约地址, 合约地址为高频出现的字符串. 一个合约地址为了防止哈希冲突, 通常具有 20 字节长度的内容. 而事实上一个 Rollup 内所涉及的所有地址若按照序号来标记, 只需要 2–4 字节即可. 4 字节可标记共计 $2^{32} - 1 = 4294967295$ 个地址, 这个数量已经远远单个 Rollup 会使用的地址总数. 此外根据第 1.3.3 节的统计结果也可以看出, 链上交易的用户地址也是具有高度重复性的.

本文提出的数据压缩方案利用部分字段使用频率高的特点, 通过部署在链上的索引表的形式, 使用短索引而替代原始字段, 整体上而言减少了链上存储的开销. 在实际链上交易存储的过程中, 交易中实际存储的仅为较短的索引, 将较长字段和较短索引对应的关系更新在链上索引表中, 后续链下节点通过从索引表还原索引替代前的交易原始信息. 索引表内的数据只需要一次更新便永久使用, 例如一个合约地址的索引关系仅需要在第一次部署的时候更新索引表内的对应关系, 之后任意次数的使用都可以直接使用索引表, 从而节省链上存储成本.

2.4.3.1 索引表组织形式

如图 8 所示, 链上索引表的组织形式具有独立的结构. 在这种结构中, 每个 Rollup 节点都拥有自己的一片独立区域空间. 在这个空间下, 包含了地址表、协议表、数据表等多张索引表. 每张索引表都由多个项组成, 每个项都包括一个 *key* 和一个 *Value*. 其中, *key* 是一个短索引, 而 *Value* 则是一个待压缩的字段.

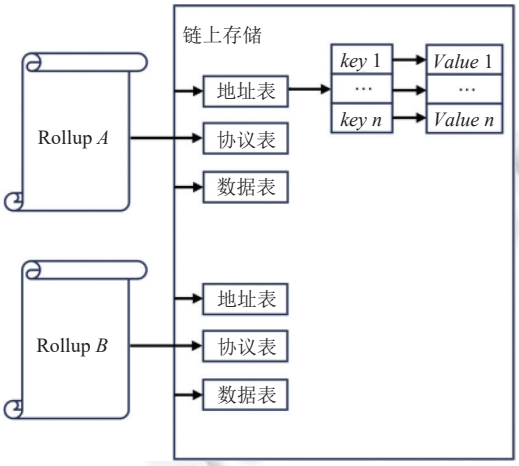


图 8 索引表链上存储示意图

地址表主要用于压缩发送方和接收方的地址信息. 通过将较长的地址长度映射为较短的短地址索引, 实现了地址信息的有效压缩. 数据表则主要应用于合约交易过程中, 将 calldata 字段中的一些较长字符串转换为短索引. 值得注意的是, 在一个 Rollup 中, 可能会部署多个合约. 由于不同合约之间的调用方式各不相同, 因此需要多张数据表来进行区分. 通常情况下, calldata 可被视为合约函数接口的调用, 而传入的数据则为函数接口调用所需的参数. 然而, 并非所有参数都需要基于索引表进行压缩. 例如, 如果参数是一个布尔值, 则无需进行压缩. 因此, 需要一个数据压缩协议来明确指示在交易调用过程中哪些参数采用了基于索引表的数据压缩, 哪些参数未使用, 以及使用了哪张索引数据表等相关信息. 协议表则负责记录这些数据压缩协议的说明. 在协议表中, *Value* 值用于存储协议说明, 而 *key* 值则是一个简单的序号. 通过这样的设计, 可以有效地组织和管理不同类型的索引表, 从而实现了地址、数据和协议等信息的高效压缩.

针对跨 Rollup 交易的场景, 索引表的更新策略需要更加精细化, 以提高整体系统的效率, 避免因为引入索引表而实际上增加了链上存储开销. 本文对索引表的更新提出一些要求, 减少索引表更新的无效开销, 避免引入索引表后链上实际开销增加的情况, 来实现更高效的索引表管理.

首先, 针对未直接涉及跨 Rollup 交易相关的内容, 无需在链上索引表中更新. 例如在地址表内, Rollup 内的地址数量会远大于发生交易所涉及的地址数量. 索引表中仅保存跨 Rollup 交易所涉及的地址信息, 而不必关心仅在 Rollup 内执行的地址. 这种做法可以有效地减少索引表的更新次数, 从而提高交易处理效率和系统性能.

其次, 本文的索引表更新时间在上传一批交易前. 在使用索引表压缩的交易在上传前, 检查是否存在未在链上存储的索引, 并及时增加该批交易所涉及的未更新索引. 这样可以避免因为未及时更新索引表而导致接收方解析交易失败或者出现异常情况, 同时, 这种方式也可以减少索引表变更的次数, 避免更新索引表的开销过大.

最后, 索引表中的索引数据应该避免冲突. 本文所使用的索引通常采用序号形式, 将增加高频字段的序号作为短索引标注. 如果不使用序号标识, 也需要选择一些不易产生碰撞的短索引生成方式. 通过在索引产生阶段不发生碰撞的方式避免索引更新异常的情况, 即使更新不成功索引缺少也能够要求重传缺失的索引. 因为若生成的短索引发生碰撞, 交易解析失败, 接收方交易将无法准确获得接收方交易, 同时对于正确的零知识证明的明文数据验证不通过, 带来系统错误.

2.4.3.2 数据压缩形式

与传统的 gzip 等数据压缩协议不同, 基于索引表方案的将长字段映射成短索引的方式, 并非通过长字段本身的重复性实现数据压缩, 而是利用该字段在未来有多次使用的需求的前提下减少开销, 这一点使得该方案具有较高的灵活性和广泛的适用性. 对于可压缩的字段, 只要可以确认该字段的相同内容在未来有可能被重复调用, 那么就可以进入索引表的压缩, 只需要保证映射后的短索引之间不发生冲突即可. 本文提出了几种可以进行有效数据压缩的协议, 并对其进行了使用场景进行了简单分析.

(1) 基于序号的地址压缩: 对于 Rollup 内的地址进行顺序编号, 然后根据这些编号构建短索引, 作为完整地址的索引. 通过这种方式, 本文能够在保持数据完整性的前提下, 显著减小地址的存储和传输成本. 这个方法的好处在于在 Rollup 内可发生交易的地址在 Rollup 节点内本身需要提前储存进入地址集合, 只有在地址集合内的地址才会被承认交易的有效性. 只需要按照进入地址集合内的顺序编号, 将序号作为压缩后的地址, 作为短索引存储在索引表中, 那么地址的映射关系必然不会产生冲突.

(2) 基于长字符串的合约内容压缩: 在智能合约 calldata 的参数中往往包含较长的字符串. 对于这类长字符串, 可以选择将长字符串的合约内容压缩通过短哈希进行压缩, 以提高存储和传输效率. 以非同质化代币 (NFT) 为例, 其铸造过程中的智能合约通常会包含一些关于 NFT 存储图片的 URL 信息. 这些 URL 具有较高的相似性, 通常是在一个基础站点下进行存储. 因此, 可以将这些 URL 的公共部分提取出来作为基础站点, 进而实现对原始字符串的压缩. 通过这种方法, 可以将合约中原本冗长的字符串进行有效压缩, 减少存储和传输过程中的数据量, 提高整体效率.

(3) 基于标准化模板的参数压缩: 在处理多参数交易的场景中可以使用该方法. 该方法的核心思想是在多个交易的参数大致相似的情况下, 利用一个基础化模板来表示这些相似的参数, 从而实现压缩数据的目标. 在此方法的实现过程中, 需要对基础化模板进行编号, 并利用序号散列标注来实现模板与原始参数之间的映射关系. 首先, 针对合约中 calldata 的多个参数, 通过对参数进行分析, 可以找到具有相似特征的参数簇. 这些参数簇可以用于构建基础化模板. 基础化模板的设计应当充分考虑参数的分布特点, 以及参数之间的相关性, 从而实现对数据特征的精确描述. 接下来, 在基础化模板的基础上, 需要对有编号的参数形态进行修改. 这意味着需要根据原始参数与基础化模板之间的差异, 对模板进行适当调整. 在模板构建和参数调整的过程中, 需要为基础化模板赋予唯一的序号, 并通过散列标注的方法, 实现模板与原始参数之间的映射关系. 序号散列标注可以采用哈希函数或其他散列算法, 将原始参数映射到相应的模板序号上. 当需要恢复原始参数时, 只需根据序号在索引表中找到对应的基础化模板, 并根据参数差异进行逆向调整, 即可实现数据的解压缩.

2.5 小 结

本节介绍了跨 Rollup 交易及其基本模型, 然后介绍了本文提出的跨 Rollup 交易的基本流程, 并采用批量化的方案来减少平均单笔交易的链上资源开销. 对于链上计算开销, 使用基于零知识证明的验证策略来验证跨 Rollup 交易的有效性. 对于链上存储开销, 使用具有基于索引表的数据压缩链上存储方案.

3 聚合规模调整算法

在第 2 节中, 本文讨论了采用批量化的方案来减少单笔交易平均的链上资源的开销, 从而提高系统的整体吞吐量. 然而, 从实际应用的角度来看, 批量化会增加交易时延, 进而给系统带来风险. 为了解决交易聚合化的问题, 本节提出一种能够动态均衡控制单次聚合规模的算法, 以便系统在时延和链上资源消耗上达到一定的平衡.

3.1 聚合化问题描述

本节所讨论的交易聚合化是指, 在发送方的 Rollup 节点接收到接收方交易 TX_{receiver} 时需要等待交易达到一个预期的聚合化规模, 然后再将达到成批的交易使用零知识证明电路中生成证明, 并将交易链上存储. 理论上, 对于零知识证明的生成, 单批次聚合规模越大, 每笔交易的平均验证计算开销越小. 因为在单批次零知识证明的过程中, 总体验证复杂度几乎不会随着聚合化规模的增加而增长. 聚合规模的增加对链上存储开销影响不大. 链上存储开销的节省主要来自于字符串的高频调用, 只要单批聚合规模不超过 blob 类交易附属大小的最大上限, 聚合规模

就不会显著影响链上存储, 但也不能超过链上存储的上限. 若仅从节省链上资源开销的角度评价, 自然是聚合规模越大, 对节约链上资源开销越好.

3.1.1 时延增加原因

在实际应用中, 本文发现聚合化方案导致交易时延的显著增加, 会极大地降低系统的可用性. 这种现象来源于两个方面.

首先, 由于交易需要批量证明, 聚合化规模的增加会导致零知识证明电路的复杂性上升. 这意味着生成零知识证明所需的时间会随之增加, 从而拖慢整批交易的处理速度; 其次, 聚合化过程需要等待一定数量的交易以形成一个完整的聚合单位. 在交易高峰期, 交易到达率较高, 因此可以较快地达到所需的聚合规模. 然而, 在非高峰期, 交易到达率较低, 系统可能需要等待较长时间才能达到预定的聚合规模.

值得进一步说明的是, 零知识电路的规模仅与聚合规模上限 $batch_{\max}$ 相关, 而生成证明的时间也仅与聚合规模上限 $batch_{\max}$ 存在直接关系. 为了确保系统性能, 只需评估合适的系统流量上限, 并根据此上限选择相应的聚合规模上限. 在交易到达率极高且接近系统预设流量上限的情况下, 实时聚合规模 $batch_{\text{now}}$ 的最优解将趋近于聚合规模上限 $batch_{\max}$, 此时对实时聚合规模 $batch_{\text{now}}$ 的选择空间极其有限. 然而, 在交易流量相对较低的情况下, 实时聚合规模 $batch_{\text{now}}$ 的调整将极大地影响交易时延, 因此应选择一个平衡解以权衡交易时延和链上开销. 实际上, 在一个系统中, 交易流量持续处于峰值的情况相对罕见, 大多数情况下, 交易到达率会波动, 但并不总是处于上限.

3.1.2 聚合化规模风险

随着交易聚合化的增加, 跨 Rollup 交易的时延会增加, 这可能会导致网络拥堵、安全风险和用户体验风险等问题. 下面将重点讨论这 3 个方面的风险.

首先是接收方 Rollup 的拥堵风险. 由于跨 Rollup 交易需要等待一定时间才能达到聚合化规模, 并在接收方 Rollup 节点内进行处理和验证, 因此可能会产生大量的交易峰值. 这会导致接收方 Rollup 内的正常交易的处理时延极大增加. 如果单批次交易规模进一步增加, 甚至会使得接收方 Rollup 节点内的代打包交易池完全被跨 Rollup 交易占据, 导致接收方 Rollup 节点无法处理正常交易.

其次是安全风险. 随着跨 Rollup 交易时延的增加, 可能会增加网络攻击和欺诈的风险. 如果攻击者能够在交易提交之前介入交易, 攻击者可能会利用这个机会进行对手的攻击, 从而损害用户的资金或其他资源.

最后是用户体验风险. 随着跨 Rollup 交易时延的增加, 可能会降低用户体验. 如果用户需要等待较长的时间才能确认交易, 可能会导致用户对应用的不信任和使用者的流失. 此外, 由于聚合化需要等待一定的时间来达到聚合化规模, 用户可能需要经常查询交易状态, 这会增加用户的操作复杂性和耗费用户的时间和精力

因此, 本文需在聚合化规模和交易时延之间找到一个平衡点, 确保基于原生链的跨 Rollup 方案的有效运行.

3.2 批量服务排队模型

跨 Rollup 交易中的聚合化问题可以被抽象成一个批量服务排队模型. 通过建立该模型, 可以对交易聚合化对系统性能的影响进行量化分析, 进而优化系统的性能, 进而利用相关算法来得到该问题的均衡解.

3.2.1 条件假设

在本模型中, 跨 Rollup 交易 (记作 TX_{receiver}) 被视作排队论模型中的顾客. 排队论模型中的服务台包括生成零知识证明电路和原生链验证零知识证明的过程. 服务结束时刻定义为零知识证明证据及完整交易信息被上传至原生链并完成验证的时刻. 对于服务接收的方式, 需要服务台一次性接收 $batch_{\text{now}}$ 笔交易同时进行, 最终该批次同时完成服务, 而非传统的一对一服务. 本节主要探讨的议题是单一批次中的实时聚合规模 $batch_{\text{now}}$ 的调整. 在每次服务台启动前, 应根据实时交易到达率来确定该规模. 在此过程中, 需要权衡链上开销成本和平均时延. 排队论模型中的总时延包括服务时长与排队等待时长之和.

需要说明的是, 尽管跨 Rollup 交易的完整生命周期理论上应当延伸至接收方 Rollup 节点读取并执行交易为止, 实际交易总时延应在该模型的平均时延基础上增加接收方 Rollup 节点的执行时间. 然而, 在实际情况中, 接收方 Rollup 节点的执行时间难以估算. 这一时间开销与跨 Rollup 交易 TX_{receiver} 的实际执行复杂度密切相关, 而事前

估算这一复杂度是具有挑战性的. 若将服务结束时刻定义为零知识证明验证完成, 零知识证明的生成时间和验证时间可以通过零知识电路规模进行推测, 零知识电路的规模仅与聚合化规模上限 $batch_{\max}$ 与待调整的实时聚合规模 $batch_{\text{now}}$ 无关. 另外需要表达的是实时聚合化规模 $batch_{\text{now}}$ 只需要小于聚合化规模上限 $batch_{\max}$ 以将其放入零知识电路计算, 而不必每次计算均达到零知识证明电路上限.

本文的模型也要提出一些基本假设和基本设置, 系统需要在正常验证流程下进行, 即不产生任何验证错误情况发生.

(1) 排队网络的模型是开环结构, 顾客接受服务后不会再次进入排队. 该现象的实际假设为跨 Rollup 交易生成证明后不会再返回待打包交易池中.

(2) 顾客是可被服务台服务的, 不会被服务台拒绝. 该现象的实际假设是处于排队过程中的跨 Rollup 交易必须是真实有效, 可以通过零知识证明检查的.

(3) 顾客到达服务台的过程是相互独立的, 且满足一定的分布规律. 该假设为不同的跨 Rollup 交易彼此之间没有依赖关系, 是相互独立的.

(4) 队列排序的规则采用特定标准的规则, 在本文的模型中采用了先进先出的规则 (FIFO). 选择 FIFO 的方案的策略来源于 Rollup 系统内及时性处理的假设, 节点应该尽快按照到达先后顺序处理待打包交易池中的交易. 若 Rollup 内重定义交易处理的顺序, 允许以优先级或其他方式处理交易, 也可以根据此更改排队论模型中的队列排序规则.

3.2.2 模型参数

本节描述后续将会出现的符号以及对应的含义, 如表 5 所示.

表 5 聚合规模均衡算法符号定义表

符号	含义
$batch_{\max}$	零知识电路中聚合规模上限
$batch_{\text{now}}$	实时的交易聚合规模
TX^{i}_{receiver}	第i笔到达的跨Rollup交易
t_i	第i笔交易到达的时刻
$t_{batch_{\text{now}}}$	第 $batch_{\text{now}}$ 笔交易到达的时刻
T_{batch}	当前批次批量打包的时刻
$g(batch_{\text{now}})$	当聚合交易规模为 $batch_{\text{now}}$ 时链上Gas计算开销
T_{proof}	零知识证明生成及链上验证总时间
g_{tol}	用户可容忍的平均计算成本上限
t_{tol}	用户可容忍的平均时延上限
$t_{\max}$	系统可容忍的平均时延上限
$g_{\max}$	系统可容忍的平均计算成本上限
α	系统偏好参数

3.2.3 问题模型

本文要解决的是问题是一个多目标优化的问题. 每一次算法运行的决策结果是当前批次批量打包的时刻 T_{batch}, 并根据分布情况可以得到实时聚合化规模 $batch_{\text{now}}$. 优化目标有两个, 其中一个是平均交易时延, 另一个是平均链上计算开销, 希望这两者的综合开销最小. 平均交易时间可以用公式 (2) 表示, 平均链上计算开销可以用公式 (3) 表示, 其中链上 Gas 计算开销 $g(batch_{\text{now}})$ 是根据链上实验的 Gas 结果拟合的函数曲线.

$$\min f_1 = \frac{\sum_{i=0}^{batch_{\text{now}}} (T_{batch} + T_{\text{proof}} - t_i)}{batch_{\text{now}}} \tag{2}$$

$$\min f_2 = \frac{g(batch_{\text{now}})}{batch_{\text{now}}} \tag{3}$$

其中约束条件如下:

$$batch_{\text{now}} \leqslant batch_{\max} \tag{4}$$

$$T_{batch} < T_{batch_{\text{now}}} + 1 \tag{5}$$

$$T_{batch} \geqslant T_{batch_{\text{now}}} \tag{6}$$

约束条件 (4) 描述了实时聚合规模在零知识电路中存在一个上限, 即实时聚合规模不得超过该上限. 约束条件 (5) 和约束条件 (6) 共同限定了实时聚合规模 $batch_{\text{now}}$ 与当前批次打包时刻 T_{batch} 之间的关系.

3.3 聚合规模调整算法

3.3.1 聚合规模均衡算法整体流程

在本研究中, 提出了一种聚合规模均衡调整算法, 旨在解决聚合时延问题以优化系统性能.

根据第 3.1.1 节的分析, 可以得知, 聚合时延问题主要在交易流量较低的情况下产生. 而当交易流量较大时, 即到达率 λ 在一定的阈值之上时, 聚合时延问题不显著的情况下, 可以考虑将多目标问题近似使用单目标问题的最优解, 从而降低算法复杂度. 在这种情况下, 用户需要对一定时延以下的交易存在时延的不敏感的特性, 即不再追求时延的最小化, 而是将系统的时延目标暂时退化成小于可容忍时延上限 t_{tol}, 但同时仍然要保障链上平均计算的开销成本应当不高于一个用户可容忍的平均计算成本上限 g_{tol}.

但是单目标问题并不是总能在容忍范围内获取到一个相对次优解, 因此在交易流量相对稀疏的情况下需要采用完整的多目标算法. 本文选择的是 NSGA-II 算法 (nondominated sorting genetic algorithm II), 是一种多目标进化优化算法, 它的设计基于遗传算法 (GA) 原理. 这种算法在优化问题中同时考虑多个目标, 并在解空间中寻找帕累托最优解. NSGA-II 算法的主要特点是采用了快速非支配排序、拥挤度算子和精英策略.

本文选择 NSGA-II 算法而非其他算法来调整跨 Rollup 交易的聚合规模, 是因为它在处理具有多个冲突目标的优化问题时能够有效地平衡各目标, 并且保持解的多样性. NSGA-II 的非支配排序、拥挤度算子和精英策略特性适用于本文的场景, 即在最小化链上资源消耗与最小化处理时延之间寻找最优平衡. 同时, 相较于其他多目标算法, NSGA-II 在处理复杂多目标优化问题时显示出更低的计算复杂度和更好的收敛性. 此外, 考虑到 NSGA-II 在多目标优化领域内的广泛应用和验证, 其稳定性和鲁棒性是本文选择它的另一原因.

本文的算法可以在满足系统性能优化要求的同时, 有效降低计算成本. 此外, 该算法充分考虑了系统偏好参数的变化, 从而在不同情况下为实际应用提供了灵活的解决方案. 算法的完整流程如图 9 所示. 具体的整体流程如下所示.

(1) 监测交易到达率和系统偏好参数变化: 首先, 本文需要实时监测诸如交易到达率和系统偏好等系统参数的变化. 交易到达率指的是单位时间内待处理的跨 Rollup 交易 TX_{receiver} 数量, 而系统参数则是在算法调整规模过程中系统初始化设置的一些参数, 例如 g_{tol}、α 等.

(2) 若交易到达率变化幅度小于预设阈值 (如 5%) 且系统偏好参数未发生变化, 则可以直接选择上一次解的结果, 无需重新计算. 这有助于节省计算资源和时间. 如果交易到达率变化幅度大于阈值或系统参数发生变化, 则需要进行下一步调整. 若仅目标函数的偏好参数 α 发生变化, 且上一次选择使用 NSGA-II 算法得到帕累托最优解集, 可直接根据偏好参数在解集中更新目标解.

(3) 单目标问题解: 如需重新计算解, 可先将多目标问题解简化为单目标问题解, 并快速求解该单目标问题的最优解. 若存在最优解, 可直接选择该解作为最终解; 若不存在最优解, 则通过 NSGA-II 算法求解多目标问题的帕累托解集.

(4) NSGA-II 解: 通过 NSGA-II 算法, 经快速非支配排序、拥挤度算子、精英策略、遗传选择交叉变异等步骤, 迭代获得帕累托最优解集.

(5) 根据目标函数的偏好参数 α 求解帕累托最优解集. 在实际应用中, 可根据具体情况适当调整目标函数的偏好参数 α, 以避免目标过于偏激.

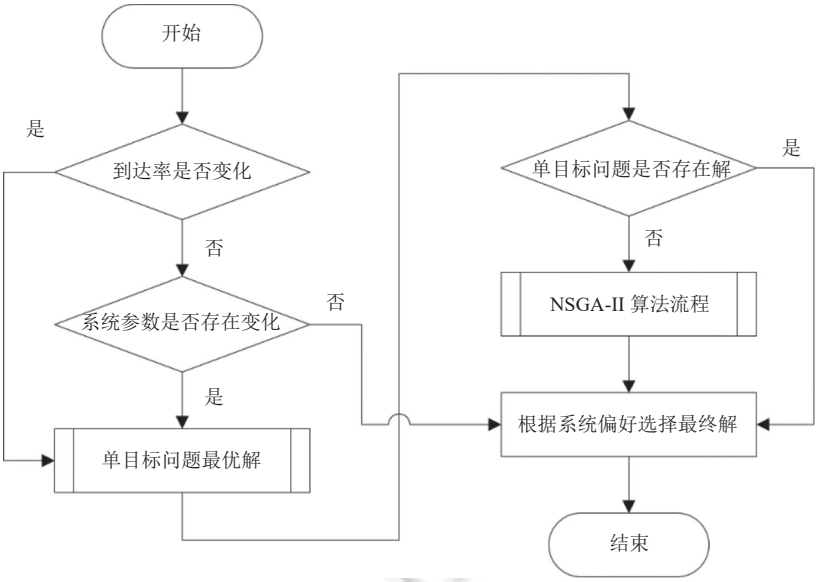


图 9 聚合规模均衡调整算法流程图

3.3.2 单目标问题算法

单目标问题算法是聚合均衡算法的一部分, 其目标是在交易量较大的情况下, 对于时延小于一定的容忍范围, 不一定要追求时延最小化的假设, 因此可以将目标退化. 如公式 (7) 所示, 优化的 f_1 函数可以退化成一个约束条件.

$$\begin{aligned} f1 &= \frac{\sum_{i=0}^{batch_{\text{now}}}(T_{batch}+T_{\text{proof}}-t_i)}{batch_{\text{now}}} \\ &\leqslant \frac{\sum_{i=0}^{batch_{\text{now}}}(T_{batch}+T_{\text{proof}})}{batch_{\text{now}}} \\ &\leqslant T_{batch}+T_{\text{proof}} \\ &\leqslant t_{\text{tol}} \end{aligned} \tag{7}$$

因此单目标问题的求解的完整约束条件和目标可以退化成下面的形式, 求解该函数也只为求 f_2 的极值, 计算复杂度相对较小.

$$\min f_2 = \frac{g(batch_{\text{now}})}{batch_{\text{now}}} \tag{8}$$

$$\text{s.t.} \begin{cases} batch_{\text{now}} \leqslant batch_{\max} \\ T_{batch} < T_{batch_{\text{now}}}+1 \\ T_{batch} \geqslant t_{batch_{\text{now}}} \\ T_{batch}+T_{\text{proof}} \leqslant t_{\text{tol}} \\ f_2 \leqslant g_{\text{tol}} \end{cases} \tag{9}$$

其中, 约束条件中的最后两个条件是相互制约的. 若当前批次批量打包的时刻 T_{batch} 存在上限, 则实时的交易聚合规模 $batch_{\text{now}}$ 也存在上限, 那么目标函数 f_2 存在下限. 若目标函数 f_2 的下限高于用户可容忍的计算压缩成本上限的时候, 该单目标问题不存在实数解.

3.3.3 非支配排序遗传算法

虽然多目标优化问题可以退化成单目标问题可以得到快速有效解, 但在交易流量较小时, 用户可容忍的条件

相对苛刻的情况下, 不存在单目标问题的实数解. 在这种情况下, 需要选择多目标优化算法来获得其帕累托最优解. 本文选择的多目标优化算法是非支配排序遗传算法 (NSGA-II).

3.3.3.1 目标函数归一化

为应用 NSGA-II 算法, 首先需要对目标函数进行归一化处理. 归一化处理的主要目的是消除目标函数之间的量纲和数值差异, 从而使得算法能够在一个统一的度量空间中对多目标问题进行有效求解. 链上成本和交易时延具有不同的单位和量纲, 数值变化范围也可能存在较大差异. 这种差异可能会导致 NSGA-II 算法在求解过程中对某些目标过度关注, 而忽略其他目标. 为解决这个问题, 本文采用归一化方法将这两个目标转换到相同的度量空间, 使得它们在求解过程中具有相近的权重. 在本问题中, 需要对链上成本和交易时延两个目标进行归一化处理.

具体来说, 本文实际使用的归一化处理方式是线性归一化. 对于目标函数 $f_i(x)$, 本文可以使用以下公式 (10) 进行归一化处理:

$$f_i^* = \begin{cases} \dfrac{f_i(x)-f_i^{\min}}{f_i^{\max}-f_i^{\min}}, & f_i(x) < f_i^{\max} \\ 1, & f_i(x) \geqslant f_i^{\max} \end{cases} \tag{10}$$

其中, $f_i^*(x)$ 表示归一化后的目标函数值, $f_i^{\min}$ 和 $f_i^{\max}$ 分别表示目标函数 $f_i(x)$ 的最小值和最大值. 更具体的, 对于等待时延, 最小值为 0, 最大值为系统可容忍的平均时延上限 $t_{\max}$. 对于平均计算成本而言, 最小值为 0, 最大值为系统可容忍的平均计算成本上限 $g_{\max}$.

3.3.3.2 快速非支配排序

快速非支配排序 (fast nondominated sorting, FNS) 是一种经典的多目标排序算法. 借鉴快速排序的思想, 通过分治策略和递归方法, 对解集进行分类和并按照支配关系判定排序. 支配关系指的是对于 x_1 和 x_2, 满足公式 (11) 则可以说明 x_1 支配 x_2, 即解 x_2 的所有目标均劣于 x_1.

$$f_1(x_1) \leqslant f_1(x_2) \text{ and } f_2(x_1) \leqslant f_2(x_2) \tag{11}$$

算法的最终结果可以将目标种群划分成为若干层, 在同一层内的解不存在支配关系, 不同层之间的节点具备支配关系. 本文最终的目标应当是最优的支配层, 也被称为帕累托最优解集, 即图 10 中的第 1 层解集.

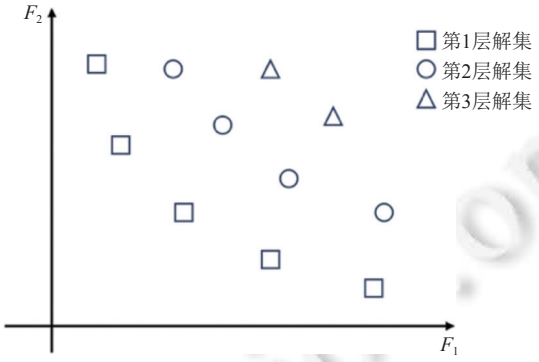


图 10 快速非支配排序结果图

3.3.3.3 拥挤度算子

拥挤度算子 (crowding distance operator, CDO) 是一种用于多目标进化优化算法的算子. 该算子通过计算个体在目标函数空间中的邻近度来度量其稀疏程度, 从而在搜索空间中保持种群的多样性. CDO 算子的主要功能是在解空间中分配适当的距离, 以使得种群中的解尽可能地分散. 对于每个目标函数, CDO 算子通过计算每个个体的左右相邻个体之间的距离, 得到个体的拥挤度值. 传统拥挤度的计算方式为:

$$CDO(i) = \sum_{j=0}^{M} \frac{f_j(x_{i+1}) - f_j(x_{i-1})}{f_j^{\max} - f_j^{\min}} \tag{12}$$

其中, $CDO(i)$ 便是当前解集中第 i 个个体的拥挤度. M 是多目标优化的目标函数个数, 在本节中为 2. $f_j(x_{i+1})$ 和 $f_j(x_{i-1})$ 分别为第 i 个粒子在目标空间上相邻的第 $i+1$ 和第 $i-1$ 个粒子在第 j 个目标函数上的取值. 需要说明的是解集中第 1 个和最后一个节点不参与拥挤度的淘汰阶段, 不需要计算.

传统的拥挤度算子存在刻画不均的情况, 若存在一部分的目标函数的稀疏度过于分散的情况下, 累加并不能直接反映出实际的系数效果, 因此需要在传统拥挤度的计算上再加入轮廓系数[23]. 使得各目标函数都应该具有较高的影响力, 这也符合本节希望均衡各目标函数的初衷. 改进后的 CDO 算子如公式 (13) 所示.

$$\begin{cases} CDO^*(i) = CDO(i) \times \sigma(i) \\ \sigma(i) = \prod_{j=1}^{M} \dfrac{f_j(x_{i+1}) - f_j(x_{i-1})}{f_j^{\max} - f_j^{\min}} \end{cases} \tag{13}$$

3.3.3.4 精英策略

精英策略 (elitist strategy) 是一种在进化算法中广泛应用的策略, 旨在确保种群中最优秀的个体得以保留并在后续进化过程中延续其优良特性. 这一策略的关键在于在每一代的进化过程中, 优秀的个体得以持续传递和传承, 从而避免因随机变异或交叉操作而损失重要的优势特征.

在本文中, 本文采用精英策略来优先选择快速非支配排序算法中排名靠前的解. 这些解在多个目标函数上均具有较优的表现, 因此保留这些解有助于提高整体种群的适应度. 同时, 本文还引入拥挤度算子 (CDO) 来对解集进行筛选, 淘汰拥挤程度较高的个体. 这样的操作有助于在解空间中维持多样性, 从而在多目标优化问题中寻找到更广泛的非劣解集.

3.3.3.5 NSGA-II 算法流程

如图 11 所示, NSGA-II 算法的主要步骤包括以下几个方面.

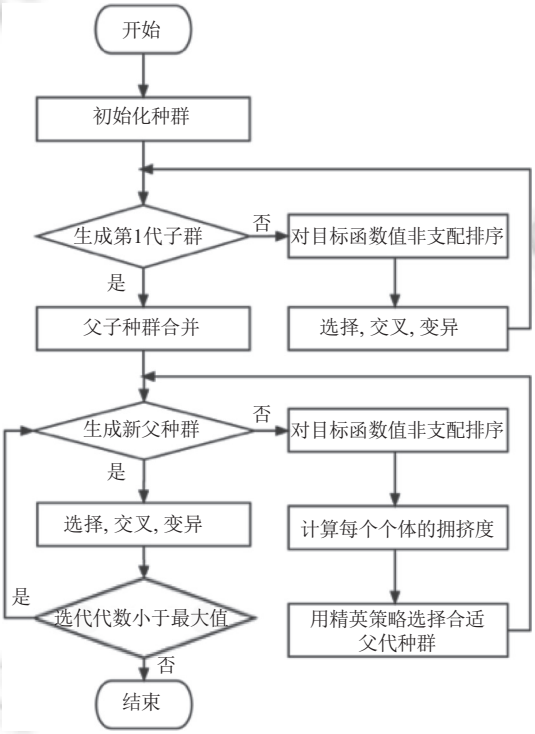


图 11 NSGA-II 算法流程图

(1) 对新生成的种群进行快速非支配排序, 按照目标函数将解分为多个非支配层次. 在每个层次内, 解之间相互不支配, 而位于较高层次的解优越于较低层次的解.

(2) 在每个非支配层次内, 计算每个解的拥挤度值, 以衡量目标空间中解的密度分布.

(3) 采用精英策略, 根据种群中个体所在的非支配层次及其拥挤度值, 优先选择支配层次高且拥挤度值较大的个体, 淘汰表现较差的解.

(4) 对选定的个体应用选择、交叉与变异等遗传算法操作, 生成新的种群.

(5) 进行逐次迭代, 直至达到预定的迭代次数或达到满意的优化效果.

NSGA-II 求解多目标优化问题的最终迭代结果是一个相对分散均衡的帕累托最优解集, 最终需要获得一个精确解来得到最终解. 理论上而言, 帕累托最优解集中的每一个解都可以作为最终结果. 系统实际上存在对目标函数的偏好需求. 接下来, 本文根据预设的偏好参数 α 设置综合评价分数. 综合评价分数是一种线性量化指标, 用于衡量解集中各解在满足多目标优化问题的各个方面的表现. 通过综合评价分数, 本文可以从帕累托最优解集中筛选出符合系统偏好的解集. 在此过程中, α 参数起到了权衡各目标函数优先级的关键作用. 综合评价分数的形式如公式 (14) 所示, 该分数越大说明其综合效果越好. 均衡的帕累托最优解集分布对系统偏好参数 α 的选择具有一定的参考价值, 可以使得偏好参数可以选择帕累托解集中相对中间的值, 有助于在选择解时避免目标函数过于偏向单一目标. 当然综合评价分数可以不仅采用线性归一化的方案, 也可以采用更复杂的模型方案.

$$score(x) = 1 - (\alpha \times f_1(x) + (1-\alpha) \times f_2(x)) \tag{14}$$

3.4 小 结

本节深入探讨了第 2 节中聚合效应所引发的新问题——聚合时延问题, 并针对该问题进行了全新的模型构建. 为了解决聚合时延问题, 本文采用了批量服务排队论模型来构建多目标优化问题. 在此基础上, 本文提出了一种聚合规模均衡调整算法, 该算法以非支配排序遗传算法 (NSGA-II) 和单目标优化解为基础, 在实现实时动态调整聚合化规模的情况下, 可以尽可能减少计算开销.

4 实验结果与分析

本节旨在对基于原生链的跨 Rollup 机制进行全面测试与评估. 首先, 本节将介绍系统测试环境. 接下来, 分别评估方案的链上计算开销和链上存储开销. 随后, 我们对聚合规模均衡调整算法进行评估, 并提供一组有效解. 最后, 对基于原生链的跨 Rollup 方案与链下跨 Rollup 方案进行整体性能评价分析.

4.1 实验环境

本文采用的测试环境是在 CentOS 7.9 的服务器上, 服务器配置情况如表 6 所示.

表 6 实验测试环境

类型	配置
CPU	Intel(R) Xeon(R) Gold 6230 CPU@2.10 GHz
CPU核数	80核
内存	256 GB
硬盘	1 TB
带宽	100 Mb/s
操作系统	CentOS 7.9

4.2 实验评价标准

在本节, 我们将设计实验评估以下几个关键方面. 首先, 针对基于原生链的跨 Rollup 方案, 本研究将对链上资源消耗进行评估. 该评估将包括两个主要部分: 一是关于验证策略计算开销的节省, 具体而言, 采用批量化零知识证明方案的证明成本开销; 二是关于存储策略的节省, 基于索引表的压缩方法节省链上存储资源量. 其次, 我们将关注聚合规模均衡调整算法的效果, 以评估该算法在不同场景下的性能表现. 最后, 将对整体采用原生链的系统与

传统链下方案的系统进行全面评价, 以便深入了解各自的优势与劣势, 为进一步的研究和应用提供有力依据.

4.2.1 链上验证计算开销的评估方案

针对链上验证计算开销的评估部分, 由于采用以太坊作为基础的原生链, 本文采用以太坊的 Gas 开销作为链上验证计算开销的衡量指标. 以太坊本身可以利用 Gas 开销来衡量执行智能合约的计算消耗, 将 Gas 开销作为评估链上验证计算开销的指标具有较高的实际意义和可行性.

为了对比基于零知识证明的批量处理方案与直接计算单笔交易的简化支付验证 (SPV) 方案在计算开销方面的差异, 本研究引入了计算效率提升比作为评估指标. 计算效率提升比旨在衡量基于零知识证明的算法相对于旧算法 (SPV 方案) 在计算效率上的优势, 进而展现. 具体来说, 该指标的计算方式如公式 (15) 所示, 从而能够直观地反映新算法在提高计算效率方面的表现. 其中公式中 Gas_{origin} 指的是 SPV 算法的计算 Gas 开销, Gas_{new} 指的是使用零知识证明后的算法计算总 Gas 开销, 包括零知识证明的消耗以及对交易明文数据的验证消耗.

$$CalRate = \frac{Gas_{\text{origin}} - Gas_{\text{new}}}{Gas_{\text{origin}}} \tag{15}$$

4.2.2 链上压缩存储开销的评估方案

针对链上传输所引发的链上存储资源消耗问题, 由于采用了 EIP-4844 协议方案, 实际上可以直接使用所需存储的字节数量来衡量链上存储资源的消耗. 为了评估压缩存储开销, 本文可以定义一个压缩效率比指标来描述压缩效率. 该指标的定义方法与计算效率提升比类似, 计算公式如公式 (16) 所示. 其中, $storage_{\text{origin}}$ 表示原始方案中链上交易的存储字节开销, 而 $storage_{\text{new}}$ 表示经过索引表压缩后的存储字节开销. 需要说明的是 $storage_{\text{new}}$ 不仅包含实际交易的存储开销, 还需要包含新增索引表字段在链上的存储开销成本.

$$StorageRate = \frac{storage_{\text{origin}} - storage_{\text{new}}}{storage_{\text{origin}}} \tag{16}$$

4.2.3 聚合均衡算法

由于聚合规模均衡算法本质上是一种快速批量服务排队模型的算法, 是一种多目标问题的求解算法的变形. 事实上如果变更 NSGA-II 算法为其他的多目标求解算法如 PESA-II 等也是可用的方案, 只是相比而言 NSGA-II 算法的帕累托解集分布更均匀, 解空间的最终结果可以反向指导偏好参数的选择. 因此实验的本质需要证明的目标是如果完全采用单目标问题求解方案, 系统会很快超过系统可容忍的平均时延上限或系统可容忍的平均计算上限, 进而使得系统的综合评价分数较低. 对比方案采用的是单目标求解的算法, 第 1 个是固定交易规模, 选取最大的 $batch_{\text{now}}$, 使得链上的计算开销降低到最低, 但该方案可能引发交易响应时延的迅速增加, 另一个对比方案是固定响应时间, 但该方案可能会使得同批聚合的规模过小, 链上的计算开销过大的问题.

4.2.4 整体系统评价

最后, 本研究对提出的基于原生链的跨 Rollup 系统进行了整体的评估, 分析了系统的整体吞吐量、时延以及与传统方案的对比. 在分析过程中, 本文特别关注了系统在去中心化和去信任化条件下的性能表现. 为了更客观地评估系统的各项性能指标, 本文在多种场景下进行了实验, 并与现有的传统方案进行了比较.

值得注意的是, 本实验的目的并非证明基于原生链的系统方案性能优于链下运营商方案, 而是探究在实现去信任化条件下, 所提出的系统能否在一定程度上达到相对可用的场景. 在确保去信任化基础上, 尽管在性能方面可能存在一定程度的权衡, 但所提出的系统能够达到现有 Rollup 技术的平均吞吐量水平下可接受的吞吐量性能.

4.3 链上计算开销

本节的链上计算开销的测试主要针对原生链生成量零知识证明策略进行验证, 以确认验证发送方 Rollup 交易的有效性的开销得到提升. 该测试主要涵盖两个方面, 即原生链的计算开销和零知识证明的计算生成时间.

4.3.1 实验数据及设计

为了衡量聚合规模对链上计算资源节省程度的影响并进行评估, 本研究将设计实验, 仅在交易数量方面构建

不同的测试用例. 在测试中, 所用交易数据为随机生成, 且均为简单的代币转账交易而不是合约交易, 这么设计的原因是合约交易复杂, 而通常两笔合约交易会有相对显著的区分. 具体设置如下: 发送方 Rollup 和接收方 Rollup 均存在一个独立的账户, 发送方 Rollup 内的账户拥有充足的资金. 接着, 发送方 Rollup 的账户会持续向接收方 Rollup 的账户进行转账, 转账金额由随机数生成器产生. 实验中, 每次测试用例的唯一区别在于交易笔数. 经过聚合处理后, 将生成跨 Rollup 交易所对应的零知识证明, 并将其发送至智能合约以进行执行. 最后, 计算智能合约的总 Gas 消耗, 以评估不同交易数量条件下, 聚合规模对链上计算资源节省程度的影响. 通过这一实验设计, 本研究旨在揭示聚合规模与链上计算资源节省之间的关系, 从而为优化跨 Rollup 方案提供有益的参考信息.

4.3.2 实验结果

对零知识证明生成证明的时间开销的开销测试结果如图 12 所示. 随着零知识电路中聚合规模上限 $batch_{max}$ 的增加, 证明生成时间呈现出上升趋势. 在批量交易数量上限为 32 的情况下, 平均证明生成时间为 18.2 s; 当上限为 256 时, 平均证明生成时间达到 54.3 s; 而在上限为 1 024 的场景下, 平均证明生成时间进一步上升至 118.6 s.

零知识证明生成时间随实时聚合规模的变化结果如图 13 所示. 在固定零知识电路中聚合规模上限 $batch_{max}$ 为 256 后, 零知识证明的证明生成时间几乎保持稳定, 不会随实际聚合规模的变化而产生剧烈变化. 总体上, 在聚合规模上限确定的情况下, 实际生成证明时间大致都在 50 s 左右上下浮动, 整体系统变化不大.

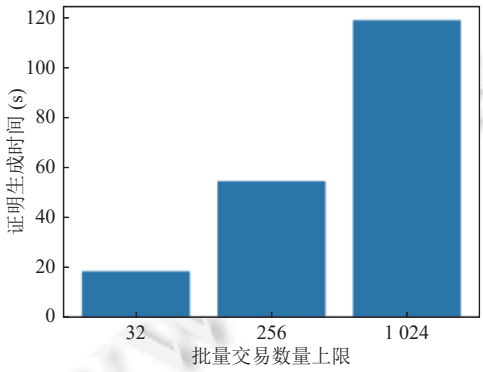


图 12 不同聚合规模上限的零知识证明生成的时间

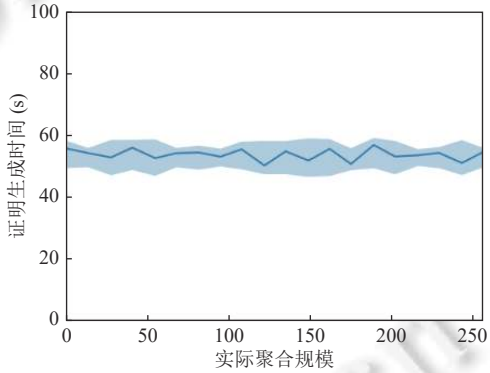


图 13 聚合规模上限固定时的零知识证明生成时间

测试链上的实际 Gas 开销的结果如表 7 所示, 测试时交易处于满负载的状态, 即实际聚合规模 $batch_{now}$ 和聚合规模上限 $batch_{max}$ 均相同, 在表中为聚合规模. 根据实验得到链上智能合约批量验证的总 Gas 开销, 最后将总 Gas 开销除以聚合规模为平均单笔交易的 Gas 开销. 通过表中数据可以得到, 对于聚合化规模越大, 所需要花费的总 Gas 开销越大, 但由于聚合规模的增长更快, 平均单笔交易的 Gas 开销实际更小.

表 7 批量零知识验证的链上 Gas 开销表

聚合规模	批量验证的总Gas开销	平均单笔交易的Gas开销
32	244 724	7 647
256	336 436	1 314
1 024	427 662	417

与本研究评估了零知识证明在聚合批量化证明中的计算效率提升, 以证实其在提高计算效率方面相较于直接 SPV 证明的优势. 测试结果如图 14 所示, 其中图例中的 *size* 为聚合规模上限 $batch_{max}$. 根据图中所展示的数据, 可以观察到以下几个现象. 首先, 零知识证明方案在提高计算效率方面需要存在一定的条件限制. 当实时聚合规模 $batch_{now}$ 较小时, 采用零知识证明的计算开销可能大于直接进行 SPV 证明, 从而对原生链带来更大的负担. 其次聚合规模上限越大, 计算效率提升比的上限越大, 最大的效率上限可以达到 98%. 最后, 在同等实时聚合规模下, 计算效率提升比会随着最大聚合规模 $batch_{now}$ 的增加而降低, 过高的最大聚合上限不一定有利.

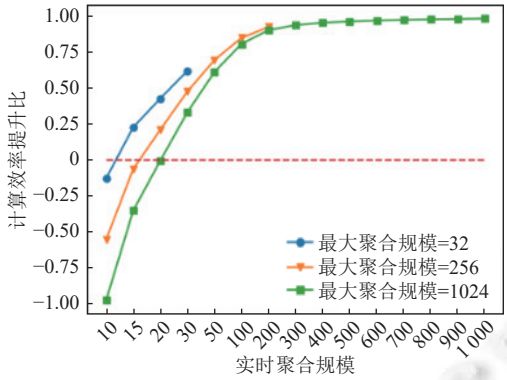


图 14 零知识证明的计算效率提升比

4.3.3 实验结果分析

通过对实验结果的分析和论证, 本文可以深入讨论零知识证明在生成时间方面的表现及其背后的原因. 根据实验数据, 零知识证明电路生成时间的现象主要受到聚合规模上限 $batch_{max}$ 的影响. 这一现象可以通过零知识证明电路的构造原理来解释. 在确定聚合规模上限 $batch_{max}$ 之后, 零知识证明电路会生成一个用于生成证明的标准 R1CS (rank-1 constraint system) 电路. 这个电路的计算规模基本上是固定的, 计算的整体复杂度取决于零知识证明的上限, 即使输入的实际聚合规模 $batch_{now}$ 发生变化, 整个电路的计算流程仍然会完整执行. 因此, 证明生成时间主要取决于聚合规模上限 $batch_{max}$, 而与实时聚合规模 $batch_{now}$ 的关系较小.

在分析零知识证明的计算效率提升比表现时, 可以观察到, 在相同交易规模的条件下, 较小的聚合规模上限 $batch_{max}$ 事实上可能导致更高的计算效率提升比, 从而在链上计算资源节省方面表现更加优越. 然而, 较小的聚合规模上限 $batch_{max}$ 也可能导致计算效率提升比的上限降低. 例如, 在聚合规模上限为 32 的情况下, 效率提升的上限仅约为 62%, 而聚合规模上限达到 1 024 的情况下, 效率提升的上限可以到达 98%. 这是由于当聚合上限 $batch_{max}$ 确定后, 验证的总开销已经近似确定了, 而聚合上限 $batch_{max}$ 越大, 总开销也会越大. 因此系统在设置聚合规模上限时, 应当充分评估当前 Rollup 的跨 Rollup 需求强度, 对未来实际使用场景中的负载状况要有一定的预估水平, 以便在实际场景中找到一个平衡点, 确保聚合规模上限 $batch_{max}$ 得到一个相对优的解.

4.4 链上存储开销

4.4.1 实验数据及设计

为了评估基于索引表的链上数据压缩策略的有效性, 本研究设计了一个实验, 采用实际有效交易数据的真实数据集进行评估. 数据集选取自 2022 年 4 月 1 日–2022 年 9 月 25 日期间的以太坊非同质化代币 (NFT) 类交易中的样本, 这些样本数据是通过爬取 NFT 交易平台而获得的. 选择 NFT 类交易中的 50 万笔交易作为数据集的原因是, 它们在 Rollup 交易场景中非常常见, 并且很可能成为跨 Rollup 交易的主要场景. 因此, 使用 NFT 合约的数据集可以较好地代表实际的跨 Rollup 交易形式. 通过使用这一真实数据集, 本研究旨在全面评估基于索引表的数据压缩策略在实际应用中的效果. 这有助于本文更准确地了解该策略对链上计算资源的节省程度, 以及在不同场景下的适用性.

与对于单笔传输验证方案中链上交易的存储的最简数据格式如后文表 8 所示, 包含完整的发送方 Rollup 序号, 接收方 Rollup 序号, 发送方用户地址, 接收方用户地址, 发送金额, 合约二进制数据等. 本文测试的对比是在使用基于索引表的数据压缩存储方案相较于传统单笔交易的最简数据存储方案的提升效率.

4.4.2 实验结果

本节测试实际应用场景中的存储压缩效率比. 通过将交易逐批存储到链上并比较实际链上存储消耗, 可以衡量交易的存储压缩效率比. 如图 15 所示, 横坐标表示随着时间推移, 逐步计算的第 k 批交易的压缩效率比, 而纵坐标则表示压缩效率比. 在实际数据集中能够达到的综合平均压缩效率比为 66.6%.

表 8 单笔跨 Rollup 交易数据结构

字段	具体含义	类型
$Rollup_{sender}$	发送方Rollup序号	bytes
$Rollup_{receiver}$	接收方Rollup序号	bytes
$Address_{sender}$	发送方用户地址	Address
$Address_{receiver}$	接收方用户地址	Address
Value	发送金额	int
Calldata	调用合约二进制数据	bytes

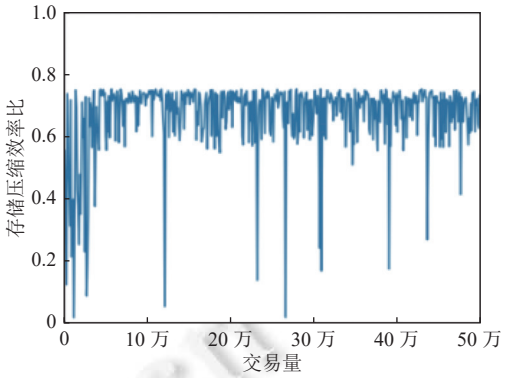


图 15 实际交易的交易存储压缩效率比

4.4.3 实验结果分析

根据上述数据, 本文发现在所有交易中, 仅出现一次的地址数量所占比例实际上较低. 这个现象的原因是, 在 NFT 交易场景中, 一个 NFT 通常会多次转手, 至少在买卖过程中产生两笔交易, 使地址出现两次. 此外, 某个地址可能持有多个 NFT, 购买两个不同的 NFT 也会导致地址的重复出现. 对于其他实际交易业务场景来说, 类似的情况同样存在. 对于单个地址而言, 仅参与一次交易的情况最终将成为少数.

关于交易存储压缩效率比的实际情况, 本文观察到存在一些批次的存储压缩效率比较低. 这一现象的原因在于, 此时这批交易中存在大量首次出现的地址. 对于首次出现的地址, 需要将其记录在索引表中, 索引表所占用的存储空间会显著降低存储压缩效率比. 但需要说明的是只要其加入了索引表, 之后再调用其便可以达到较小的存储压缩效率比了.

此外, 本文还发现存储压缩效率比实际上存在一个上限, 该上限主要由短索引和长索引的压缩比决定. 为了定量衡量单笔传输验证方案中链上交易的存储的最简数据格式, 如表 8 所示, 本文可以发现, 完整的地址需要 20 字节, Rollup 序号一般需要 4 字节, 发送金额需要 4 字节. 在忽略合约二进制数据的前提下, 原始传输方案中, 单笔交易至少需要 $4+4+20+20+4=52$ 字节的存储空间. 若采用 4 字节的方式表示压缩后的地址短索引, 并根据索引表的压缩格式 (如表 4 所示), 单笔交易仅需 $4+4+4+2=14$ 字节的存储空间, 理论存储压缩效率比的上界为 $\frac{52-14}{52}=73\%$. 实际上, 压缩后的地址短索引可以采用更短的字节表示, 例如 2 字节, 从而进一步提高存储压缩效率比. 然而, 如果仅采用 2 字节表示, 那么理论上当前 Rollup 的可用地址数量上限将变为 $2^{16}=65536$ 个, 这一数量可能偏少, 需要在实际应用中仔细考虑.

4.5 聚合规模均衡算法

4.5.1 实验数据及设计

本研究部分的实验旨在验证聚合规模均衡算法的价值和有效性. 为了评估交易到达模型, 本文假定交易的到达遵循泊松分布. 泊松分布被广泛认为是实际排队论中的一种重要概率分布形式, 具有描述独立离散事件在特定时间间隔或空间范围内发生的次数的特点. 在许多实际场景中, 例如客户到达服务台、电话呼叫进入呼叫中心或网络数据包到达服务器等情况, 泊松分布都能够为排队论提供合适的模型来分析和预测系统行为. 交易的实际构成与第 4.3 节所述相同, 即交易仅与数量相关, 且每笔交易均为相似交易.

4.5.2 实验结果

在本研究中, 采用了 NSGA-II 遗传算法对多目标问题进行求解. 如图 16 所示, 该算法的收敛效果非常显著. 经过大约 100 代的迭代过程, 本文可以观察到遗传算法已经找到了较优的解决方案. 在相似多目标问题求解方法的对比中, NSGA-II 具有较快的收敛速度. 值得注意的是, 在完成 100 代迭代的过程中, 平均计算时间仅为 1.53 s. 这意味着本文可以在很短的时间内 (秒级) 获得有效的收敛解. 这一结果表明, NSGA-II 遗传算法不仅具有较高的

求解效率, 而且在实际应用中具有很好的性能.

为了证明均衡算法的有效性, 本研究采用了一种基于固定交易笔数和固定聚合响应时间的方案, 该方案考虑了不同交易到达率 λ 和系统偏好参数 α 的场景. 实验结果如图 17 所示, 在各种到达率和偏好参数条件下, 均衡算法的综合评价得分始终保持最佳表现. 因此, 为了获得理想的响应时间和链上计算效率的平衡, 有必要采用多目标优化的均衡算法.

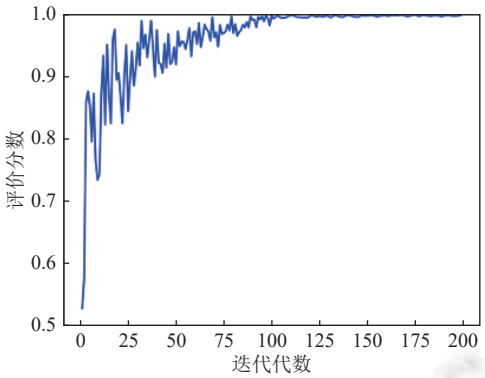


图 16 NSGA-II 遗传算法收敛速度

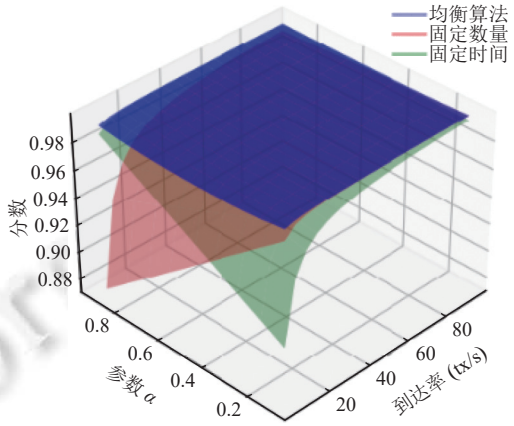


图 17 多目标优化问题综合评价分数

聚合交易响应时间的结果如图 18 所示, 此时系统的偏好参数 α 为 0.5. 从图中可以明显看出, 随着交易到达率的提高, 聚合响应时间呈现出递减的趋势. 值得注意的是, 均衡算法和固定交易数量方法在响应时间上逐渐趋于一致. 然而, 在交易到达率较低的情况下, 采用固定交易数量方法的聚合响应时间会迅速上升, 从而导致实际交易时延的显著增加. 在交易到达率只有每秒 5 笔的时候, 相较于最大聚合规模的选择, 交易时延可以减少 54.9%, 尽可能提高系统的可用性.

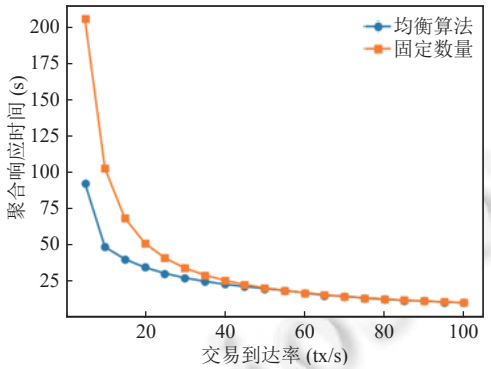


图 18 聚合交易响应时间

4.5.3 实验结果分析

对于 NSGA-II 遗传算法, 其在收敛速度和效果方面表现出色, 因此经常被用作解决多目标优化问题的核心算法. 除了本研究的实验结果之外, 文献 [23−25] 等也为 NSGA-II 算法在多目标问题中的广泛适用性提供了有力支持. 因此, 将 NSGA-II 算法作为聚合规模多目标优化问题中均衡算法的核心方法是合理的.

关于均衡算法的聚合交易响应时间结果, 本文发现在交易到达率较高的情况下, 多目标优化问题可被简化为单目标优化问题, 这将有助于减少遗传算法所需的计算开销. 若能确定交易的到达模型遵循泊松分布, 那么本文甚至可以提前根据到达率计算最优聚合响应时间, 并通过查找预先生成的表格来获得结果, 从而避免进行实时计算.

4.6 整体系统性能评价

4.6.1 实验数据及设计

本节将对比本研究提出的基于原生链的跨 Rollup 方案与现有的链下跨 Rollup 方案. 链下跨 Rollup 方案通常基于中间运营商或流动性市场参与者, 这些方案在很大程度上都存在一定程度的信任假设. 本文将具体对比包括 Hop 方案和 Orbiter Bridge 方案在内的现有方案. 此外, 还将与直接使用简化支付验证 (SPV) 证明以及链上无存储压缩方案的基于原生链的跨 Rollup 方案的最原始形态进行比较, 该方案称为原始原生链方案. 对于本文提出的方案称为聚合批量上限为 $batch_{max}$ 的原生链方案. 实验的数据集采用第 4.4 节的数据.

最终评价指标主要包括吞吐量和时延两个方面, 以更全面地了解各种方案在性能和效率方面的差异. 其中需要对吞吐量和交易时延进行一定的定义说明. 在本实验中, 所指的平均交易时延是从发送方 Rollup 内成功上链且状态根生效的时刻开始计算, 直至接收方 Rollup 有效接收并可开始执行交易的时刻为止. 系统最大吞吐量指的是在 Rollup 产生交易速度较大的前提下, 尽可能提高各方案的最大单位时间的系统吞吐量.

4.6.2 实验结果

系统的整体实验结果如表 9 所示, 结果包含几种方案的系统最大吞吐量和平均交易时延.

表 9 系统整体吞吐量性能对比

实验方案	系统最大吞吐量 (tx/s)	平均交易时延 (s)
原始原生链方案	15.2	22.6
NativeBridge-32	109.8	30.5
NativeBridge-256	302.4	72.8
NativeBridge-1024	785.7	150.9
Orbiter Bridge	1 278.6	24.6
Hop	1 446.3	14.1

4.6.3 实验结果分析

通过表中数据本文可以看出, 最原始的原生链方案相较于其他方案, 吞吐量低. 因此, 在本文之前不存在直接使用原生链作为跨 Rollup 方案的基础. 而对于本文提出的批量化交易处理的原生链方案 NativeBridge, 系统的吞吐量在随着聚合规模上限逐步增加, 并逐步达到了相对可用的水平. 本文的方案在不依赖任何链下假设, 在完全去信任化的前提下交易吞吐量最高能达到了每秒 785.7 笔.

事实上而言, 相关文献[1]指出, 现存的 Rollup 系统吞吐量性能通常在 1000–3000 tx/s 之间, ZKsync[11]的官网上公开自身的交易吞吐量上限大约在 2 000 tx/s. 而且单个 Rollup 内并不会所有交易都是跨 Rollup 交易, 相对而言本文提出的基于原生链的跨 Rollup 方案的系统吞吐量是可用的.

4.7 小 结

本节设计了实验, 综合评价了本文所设计的各个部分, 包括链上计算部分, 链上存储部分, 聚合规模均衡算法的效果, 以及系统整体的性能评价. 在链上计算部分, 使用零知识证明的验证效率较大提升了链上平均单笔交易的计算开销, 最大的计算效率提升比可以达到 98%. 在链上存储部分, 在实际数据集的测试下, 平均的链上存储压缩效率比可以达到 66.6%. 在聚合规模均衡算法中, 它相较于传统单目标的选择方案都能获得更高的综合评价分数. 该算法有助于在实际环境下, 通过调整聚合规模以平衡链上开销和系统时延. 最后整体系统最高吞吐量可以达到每秒 785.7 笔交易, 相较于正常 Rollup 的交易 1000–3000 tx/s 的吞吐量上限已经达到一个可用的系统状态.

5 总结与展望

5.1 本文工作总结

原生链是 Rollup 系统的安全性来源, 基于原生链的跨 Rollup 方案可以达到在无额外信任假设的前提下, 实现 Rollup 间的互操作. 但是原生链本身的性能存在瓶颈, 其链上计算资源和存储资源受限, 直接使用链下验证的

策略会使得系统的吞吐量低下. 本文提出了一种基于原生链的跨 Rollup 方案, 称为 NativeBridge, 该方案主要包括基于零知识证明的交易有效性证明方案、基于索引表数据压缩的交易存储方案和聚合规模均衡调整算法, 通过聚合交易批量集中处理的方式减少单笔交易的平均计算和存储开销, 增加系统的吞吐量. 本方案最大可实现 98% 的链上计算开销优化, 平均链上资源开销能降低 66.62%, 系统整体吞吐量上限最高可达到每秒 785.6 笔交易, 相较于现有 Rollup 系统的平均每秒 1000–3000 笔交易的吞吐量水平, 已经达到一个相对可用的水平.

5.2 进一步工作展望

本文设计的基于原生链的跨 Rollup 系统未来可在以下方面进一步改进. 首先, 目前采用的零知识证明策略在生成证明时间上还较长, 同时需要初始化公共参数, 存在潜在的安全风险, 因此未来可以考虑采用无公共参数初始化且证明生成时间较短的零知识证明策略如 Bulletproof[26]或 Aurora[27]. 其次, 本文设计的基于原生链的跨 Rollup 系统能够满足最基础的单向的交易互操作方案, 未来跨 Rollup 交易需要兼容更多的交易互操作形式, 包括交易的事务性、多 Rollup 间的交互形式. 可以通过选择适当的管理器并在本文提出的框架上进行扩展来满足这些需求. 此外, 本文的方案主要基于以太坊区块链及其扩容解决方案 Rollup, 但其他实现了 Rollup 扩容的区块链同样可以利用本文提出的方案实现跨 Rollup 交易.

References:

[1] Sguanci C, Spatafora R, Vergani AM. Layer 2 blockchain scaling: A survey. arXiv:2107.10881, 2021.

[2] McCorry P, Buckland C, Yee B, Song D. SoK: Validating bridges as a scaling solution for blockchains. Cryptology ePrint Archive, 2021. 1589. https://eprint.iacr.org/2021/1589

[3] Mohanty S K, Tripathy S. *n*-HTLC: Neo hashed time-lock commitment to defend against wormhole attack in payment channel networks. Computers & Security, 2021, 106: 102291. [doi: 10.1016/j.cose.2021.102291]

[4] Whinfrey C. Hop: Send tokens across rollups. 2021. https://ethresear.ch/t/hop-send-tokens-across-rollups/8581

[5] Orbiter. Orbiter Bridge. 2023. https://www.orbiter.finance/

[6] Buterin V, Feist D, Loerakker D, Kadianakis G, Garnett M, Taiwo M, Dietrichs A. EIP-4844: Shard blob transactions. 2022. https://eips.ethereum.org/EIPS/eip-4844

[7] Liu JT, Wan SG, He XB. Alias-Chain: Improving blockchain scalability via exploring content locality among transactions. In: Proc. of the 2022 IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS). Lyon: IEEE, 2022. 1228–1238. [doi: 10.1109/IPDPS53621.2022.00122]

[8] Fiege U, Fiat A, Shamir A. Zero knowledge proofs of identity. In: Proc. of the 19th Annual ACM Symp. on Theory of Computing. New York: ACM, 1987. 210–217. [doi: 10.1145/28395.28419]

[9] Chiesa A, Ojha D, Spooner N. FRACTAL: Post-quantum and transparent recursive proofs from holography. In: Proc. of the 39th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. Zagreb: Springer, 2020. 769–793. [doi: 10.1007/978-3-030-45721-1_27]

[10] Groth J. On the size of pairing-based non-interactive arguments. In: Proc. of the 35th Annual Int'l Conf. on Advances in Cryptology, Part II. Vienna: Springer, 2016. 305–326. [doi: 10.1007/978-3-662-49896-5_11]

[11] ZKsync. The elastic chain. 2023. https://zksync.io/

[12] Maller M, Bowe S, Kohlweiss M, Meiklejohn S. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 2111–2128. [doi: 10.1145/3319535.3339817]

[13] Gabizon AG, Williamson ZJ, Ciobotaru O. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive. 2019. 953. https://eprint.iacr.org/2019/953

[14] Zhang JH, Xie TC, Zhang YP, Song D. Transparent polynomial delegation and its applications to zero knowledge proof. In: Proc. of the 2020 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2020. 859–876. [doi: 10.1109/SP40000.2020.00052]

[15] Xie TC, Zhang JH, Cheng ZR, Zhang F, Zhang YP, Jia YZ, Boneh D, Song D. zkBridge: Trustless cross-chain bridges made practical. In: Proc. of the 2022 ACM SIGSAC Conf. on Computer and Communications Security. Los Angeles: ACM, 2022. 3003–3017. [doi: 10.1145/3548606.3560652]

[16] Virgo ZK reference implementation. 2023. https://github.com/yudongbei/Virgo/blob/master/README.md

[17] ZoKrates. 2023. https://zokrates.github.io/

[18] libsnark. 2023. https://github.com/scipr-lab/libsnark

[19] Ames S, Hazay C, Ishai Y, Venkitasubramaniam M. Ligero: Lightweight sublinear arguments without a trusted setup. In: Proc. of the 2017 ACM Sigsac Conf. on Computer and Communications Security. Dallas: ACM, 2017. 2087–2104. [doi: 10.1145/3133956.3134104]

[20] Ben-Sasson E, Bentov I, Horesh Y, Riabzev M. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, 2018. 046. https://eprint.iacr.org/2018/46

[21] Setty S. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Proc. of the 40th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2020. 704–737. [doi: 10.1007/978-3-030-56877-1_25]

[22] Wahby R S, Tzialla I, Shelat A, Thaler J, Walfish M. Doubly-efficient zkSNARKs without trusted setup. In: Proc. of the 2018 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2018. 926–943. [doi: 10.1109/SP.2018.00060]

[23] Wang XD, Hirsch C, Kang S, Lacor C. Multi-objective optimization of turbomachinery using improved NSGA-II and approximation model. Computer Methods in Applied Mechanics and Engineering, 2011, 200(9–12): 883–895. [doi: 10.1016/j.cma.2010.11.014]

[24] Yusoff Y, Ngadiman MS, Zain AM. Overview of NSGA-II for optimizing machining process parameters. Procedia Engineering, 2011, 15: 3978–3983. [doi: 10.1016/j.proeng.2011.08.745]

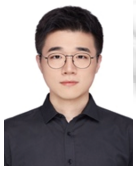
[25] Zheng WJ, Liu YF, Doerr B. A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In: Proc. of the 2022 AAAI Conf. on Artificial Intelligence. Palo Alto: AAAI, 2022. 10408–10416. [doi: 10.1609/aaai.v36i9.21283]

[26] Bünz B, Bootle J, Boneh D, Poelstra A, Wuille P, Maxwell G. Bulletproofs: Short proofs for confidential transactions and more. In: Proc. of the 2018 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2018. 315–334. [doi: 10.1109/SP.2018.00020]

[27] Ben-Sasson E, Chiesa A, Riabzev M, Spooner N, Virza M, Ward NP. Aurora: Transparent succinct arguments for R1CS. In: Proc. of the 38th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. Darmstadt: Springer, 2019. 103–128. [doi: 10.1007/978-3-030-17653-2_4]

张子龙(2000－), 男, 博士生, 主要研究领域为区块链, 数字货币.

蒋硕轩(1999－), 男, 硕士, 主要研究领域为区块链, 数字货币.



贾林鹏(1995－), 男, 博士, 助理研究员, CCF 专业会员, 主要研究领域为区块链高通量技术.

孙毅(1979－), 男, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为区块链, 社交视频.