软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(7):3151−3183 [doi: 10.13328/j.cnki.jos.007366] [CSTR: 32375.14.jos.007366]

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

区块链测试基准综述*

张 孝[1], 秦春玲[1,2], 王文收[3], 刘 昊[1], 陈晋川[1], 杜小勇[1]

[1](中国人民大学 信息学院, 北京 100872)

[2](腾讯科技 (北京) 有限公司, 北京 100193)

[3](资金集中收付管理中心, 北京 100036)

通信作者: 陈晋川, E-mail: jcchen@ruc.edu.cn

摘 要: 近年来, 区块链技术已经广泛应用到数据要素流通、金融、物流、政务以及司法等领域. 随之也出现了若干区块链测试基准, 以评测不同区块链系统的性能. 然而, 现有区块链测试基准内容差异较大, 缺少统一的区块链测试基准框架来规范基准的内容, 也缺失统一的指标体系来明确区块链系统在性能和安全性方面应具备的能力. 从数据库发展历程来看, 统一的、可重复的、公平的测试基准规范可以更好地引导行业发展. 区块链本身也是一种特殊的分布式数据库管理系统, 应该借鉴数据库技术在发展过程中积累的宝贵经验. 参考数据库测试基准的内容, 针对区块链系统所特有的去中心化、不可篡改、可信等特点, 提出一个区块链测试基准参考框架 UFBCB. 该参考框架定义了区块链测试基准的 5 个核心要素: 应用模型、数据模型、负载、指标和执行规则, 并阐明 5 个要素相互之间的关系, 为区块链性能评测提供了一个统一的参考标准. 提出一个全面衡量区块链系统各项关键能力的测试指标体系, 包含性能、能耗、扩展性和安全性这 4 个方面的指标, 基本覆盖了已有的区块链测试指标. 在此基础上, 将 UFBCB 框架与现有区块链测试基准进行详尽的对比分析, 指出目前区块链测试基准存在的普遍问题. 最后, 对区块链测试基准未来的发展方向进行了讨论.

关键词: 区块链; 测试基准; 性能评估; 可扩展性; 安全性

中图法分类号: TP311

中文引用格式: 张孝, 秦春玲, 王文收, 刘昊, 陈晋川, 杜小勇. 区块链测试基准综述. 软件学报, 2025, 36(7): 3151–3183. http://www.jos.org.cn/1000-9825/7366.htm

英文引用格式: Zhang X, Qin CL, Wang WS, Liu H, Chen JC, Du XY. Survey on Blockchain Benchmarks. Ruan Jian Xue Bao/Journal of Software, 2025, 36(7): 3151–3183 (in Chinese). http://www.jos.org.cn/1000-9825/7366.htm

Survey on Blockchain Benchmarks

ZHANG Xiao[1], QIN Chun-Ling[1,2], WANG Wen-Shou[3], LIU Hao[1], CHEN Jin-Chuan[1], DU Xiao-Yong[1]

[1](School of Information, Renmin University of China, Beijing 100872, China)

[2](Tencent Technology (Beijing) Co. Ltd., Beijing 100193, China)

[3](Centralized Fund Collection and Payment Management Center, Beijing 100036, China)

Abstract: In recent years, blockchain technology has undergone rapid development and found widespread application in areas such as data circulation, finance, logistics, government affairs, and judiciary. Alongside this growth, various benchmarks have emerged to evaluate the performance of different blockchain systems. However, these benchmarks exhibit significant differences in scope and methodology, lacking a unified framework to standardize their design and evaluation processes. In addition, there is no consistent metric system to explicitly define the performance and security capabilities required of blockchain systems. Drawing from the development history of databases, a unified, repeatable, and fair benchmarking standard can provide valuable guidance for industry advancement. As a specialized form of

* 基金项目: 国家重点研发计划 (2022YFB2702102)
收稿时间: 2024-06-18; 修改时间: 2024-09-17; 采用时间: 2024-11-24; jos 在线出版时间: 2025-04-25
CNKI 网络首发时间: 2025-04-27

distributed database management system, blockchain can benefit from the extensive experience gained through the evolution of database technology. Inspired by database benchmarks and considering the unique characteristics of blockchain systems such as decentralization, immutability, and trustworthiness, a blockchain benchmarking reference framework, UFBCB, has been proposed. This framework identifies five core elements essential to blockchain benchmarks: application model, data model, workload, metrics, and execution rules, and clarifies the interrelationship among these elements. It provides a unified standard for evaluating blockchain performance. Moreover, a comprehensive metric system encompassing performance, energy efficiency, scalability, and security is introduced, aiming to capture the critical capabilities of blockchain systems comprehensively. Based on this framework, a detailed comparative analysis of existing blockchain benchmarks is conducted, highlighting common issues and deficiencies. Finally, this study discusses potential directions for the future development of blockchain benchmarking.

Key words: blockchain; benchmark; performance evaluation; scalability; security

1 引 言

区块链系统是一种提供通用编程接口、去中心化的分布式数据库[1]. 2008 年加密数字货币系统比特币[2]的出现标志着区块链的诞生, 这一时期区块链主要充当比特币的账本[3], 记录交易, 并支持简单的智能合约 (smart contract), 这也是区块链常常被称为“分布式账本”的由来. 2014 年出现的以太坊[4]旨在全面支持图灵完备的智能合约, 为去中心化应用程序 (DApp) 提供基础, 成为通用计算基础设施 (“The world’s computer”)[5]. 区块链的应用范围从加密数字货币、DApp 逐渐扩展到金融[6,7]、供应链[8,9]、物联网[10−13]、医疗[14,15]、政府服务、司法、能源[16,17]等领域.

区块链利用块链式数据结构来存储日志, 通过分布式共识协议来生成和更新状态, 并使用密码学技术保证消息的可信性. 区块链架构从上至下可分为应用层、执行层、数据层、共识层和网络层[18]. 根据对参与节点身份的要求, 可以将区块链分为公有链和许可链两大类[19−21]. 公有链也称为无许可链 (permissionless blockchain), 任何节点都可以在不经授权的情况下参与或退出区块链网络. 典型代表是比特币和以太坊. 许可链 (permissioned blockchain) 又可以细分为联盟链 (consortium blockchain) 和私有链 (private blockchain), 共同特点是节点需要先得到授权, 典型代表是 Hyperledger Fabric (https://www.hyperledger.org/) 和 FISCO BCOS (http://www.fisco-bcos.org/).

去中心化以及高容错是所有区块链系统的核心设计目标, 而实现这两个目标通常都需要在一定程度上牺牲系统性能[22], 例如比特币系统平均每秒只能处理 7 笔交易, 以太坊系统经过扩容升级之后也只能达到每秒数百笔交易[23], 和传统中心化系统性能相距甚远. 针对这一问题, 近年来涌现出一批新型区块链系统, 采用离链[24]、侧链[25]、跨链[26]、事务并发执行[27]、分片[28,29]和有向无环图 (DAG)[30,31]等技术来提升性能.

层出不穷的区块链系统给上层应用带来新的问题. 例如, 如何衡量不同区块链系统的性能[32]? 如何根据应用场景特点选择合适的区块链系统[33]? 此外, 不同区块链系统在不同方面表现各异. 例如, 采用工作量证明 (PoW) 协议的区块链需要传输的消息数量较少, 适合规模很大的区块链网络, 但吞吐能力较低; 采用实用拜占庭容错 (practical Byzantine fault tolerance, PBFT) 协议的区块链可以支持较高的吞吐量, 但由于其消息传输量随网络节点数量增加快速上升, 严重限制了此类区块链系统的网络规模; 采用分片、侧链以及有向无环图 (DAG) 等技术的区块链系统虽然提升了吞吐效率, 但在安全性方面都有不同程度的削弱. 另一方面, 不同的应用场景对区块链的性能和安全性也有着不同的要求. 例如, 加密数字货币和金融领域最看重安全性, 大部分去中心化应用需要高吞吐率和低延迟, 而司法和物流则更看重数据的不可篡改性.

从数据库行业的发展过程来看, 上述问题应该通过建立测试基准和相应的评测体系来解决. 数据库的测试基准是一套用于比较和评估不同数据库系统性能的规范, 它通常包括以下几个要素: 应用模型、数据模型、负载、指标和执行规则. 应用模型是指基准所模拟的特定应用场景. 数据模型是指代表业务特征的数据模式和数据集, 其中, 数据模式描述了数据库中的表结构、表之间的关联关系和完整性约束, 数据集是根据真实应用场景生成的测试数据. 负载是一组模拟真实业务场景或特定使用情况的操作集合, 包括查询、插入、更新和删除等各种操作, 以及操作的类型、比例和分布情况. 指标是一组用来评估数据库性能的标准, 比如吞吐率、响应时间等. 执行规则设

定了基准测试的运行方式, 以确保正确地获取测试指标, 包括实施测试所需的软硬件系统配置、攻击配置以及测试流程等. 部分测试基准还提供了生成测试数据集和负载, 并执行测试用例的可执行程序或源码, 这通常被称为测试工具.

目前, 数据库测试基准种类繁多, 其中由事务性能管理委员会 (TPC) 发布的系列基准得到业界广泛认可, 包括 TPC-C[34]、TPC-B[35]、TPC-E[36]、TPC-H[37]和 TPC-DS[38]. 具体内容将在第 2 节中介绍. 数据库测试基准为不同数据库系统提供了一个公平的性能评价标准, 确保了比较的可比性和评估的可重复性. 同时, 它也推动了数据库技术的创新和市场的良性竞争, 帮助用户做出更明智的数据库选择.

随着区块链技术的不断发展和应用领域的扩大, 越来越需要对不同区块链系统进行评测以匹配各种应用场景和性能要求. 学术界和工业界对此都展开了积极的研究和探索. 新加坡国立大学的 Dinh 等人[39]提出了首个用于分析许可链的测试基准 BlockBench. BlockBench 使用真实和合成的工作负载, 从吞吐率、延迟、可扩展性和容错性这 4 个维度来评测许可链系统. BlockBench 还设计并实现了一个开源的工具 (https://github.com/ooibc88/BlockBench), 并利用该工具测试了以太坊、Hyperledger Fabric 以及 Parity 的性能. 然而, BlockBench 没有针对许可链的应用场景设计应用模型和数据模型, 其测试指标也相对简单. Hyperledger 社区提出了一个专门针对 Hyperledger Fabric 的测试基准 Hyperledger Caliper[40], 它通过运行用户自定义的工作负载来分析智能合约的吞吐量、延迟和资源消耗. 但 Caliper 并未定义应用模型、数据模型和负载, 其指标也未考虑到系统的安全性. 中国信息通信研究院研发的区块链性能评测平台 Trusted Bench[41]也沿用了 Caliper 架构.

除此之外, 其他具有一定影响力的区块链测试基准包括 Diablo[42]、Gromit[43]、BCadvisor[44]、BCTMark[45]、DAGBENCH[46]、xBCBench[47]、BlockMeter[48]、AlphaBlock[49]等.

基于这些区块链测试基准, 用户可以在相同的测试环境下分析和比较不同区块链系统的性能, 这在一定程度上解决了区块链系统评测的问题. 然而, 对比测试基准的几个关键要素, 我们发现上述区块链测试基准仍需要完善.

● 应用模型: 目前, 大多数区块链测试基准并未定义基于区块链真实业务场景的应用模型. Diablo 中定义了交易所、视频分享、游戏等 5 个去中心化的应用程序. BBSF 中使用了去中心化代币交易所、NFT 市场、NFT 铸造、体育博彩网站这 4 个工作场景. 然而, 这些研究并未深入分析这些场景是否适合区块链应用. 而其他大多数的研究则没有明确定义应用模型.

● 数据模型: 大多数区块链测试基准并未讨论数据集的生成方式[27,40,43,45]. 在 BlockBench 中, 研究者使用了 3 个流行的以太坊智能合约 EtherId、Doubler 和 WavesPresale, 并借鉴了数据库基准的数据模型, 如 YCSD 和 SmallBank. Klenik 等人[50]将 TPC-C 转换为智能合约的模型. 部分测试基准利用智能合约预加载了不同数量的交易作为数据集[39,46]. 然而, 目前尚未看到有专门针对许可链应用的数据集和数据集产生工具.

● 负载: 当前的测试基准中, 负载主要采用数据库负载或简单的合成负载, 这并不能真实反映区块链应用的特性. 例如, BlockBench 中使用的 YCSD 和 SmallBank 负载是数据库的基准负载, 而在 Hyperledger Caliper 中并未提供预定义的负载, 用户需要自行配置.

● 指标: 目前测试基准缺少了对测试指标的统一定义. 例如, 在定义交易吞吐量和延迟的公式时, 一些文献考虑了 N 个节点的平均值[51], 但大多数文献并未考虑这一点. 在计算资源利用率时, 部分文献考虑了 RPC 请求消耗的资源[44], 但大多数文献并未考虑. 此外, 大多数测试基准中也缺乏对安全性和容错性指标的定义.

● 执行规则: 目前的测试基准中, 往往缺乏对测试环境预设的详细描述, 例如区块的大小、区块生成的时间间隔、链上存量区块的数量 (即区块的高度)、数据存储的格式、并发度等. 这使得评估结果难以复现, 从而无法验证其准确性.

作为一种特殊的分布式数据库, 区块链的测试基准可以充分借鉴数据库测试基准的内容. 但值得注意的是, 由于区块链系统自身特点, 其测试基准与数据库测试基准也应该存在一定差异. 首先, 从数据集角度来看, 传统数据库测试基准通常包含数据集, 其中数据规模和数据分布是影响性能的关键参数. 在区块链系统中, 主要瓶颈在于数据上链的速度, 在此过程中已有数据的规模或分布对系统性能影响相对较小. 很多区块链测试基准也就不关心数据的规模. 然而, 随着区块链应用的发展, 已经出现了一些查询类应用场景, 这些场景下性能受到数据集影响较大.

因此, 区块链测试基准需要根据应用需求对数据集的规模和分布进行适当的配置. 其次, 从负载角度来看, 数据库测试用例往往通过数据库事务, 比如 SQL 语句来定义操作, 而区块链系统常常采用智能合约. 最后, 由于区块链系统通常在弱信任或无信任的环境中运行, 安全性和容错性是区块链测试基准必须考虑的指标. 因此, 区块链测试基准还应定义攻击模式, 因为安全性和容错性的评估需要针对具体的攻击, 例如评估在 DDoS 或女巫攻击下系统性能的影响程度.

目前, 针对区块链测试基准的综述分析工作还相对较少. Touloupou 等人[52]从支持区块链类型、指标、测试工具等方面对区块链测试基准进行了比较分析, 并提出了一个区块链基准测试工具. Fan 等人[18]从支持的区块链类型、工作负载、指标等方面对 BlockBench、Caliper 和 DAGBENCH 进行了比较. Chacko 等人[53]提出了测试基准应考虑的 4 个要素: 系统配置、参数调整、工作负载和指标. 他们利用这 4 个要素对 BlockBench、Caliper、Diablo、Gromit 和 BCTMark 进行了案例研究, 评估了其局限性. 其中系统配置和参数调整属于本文所提框架中的执行规则. 这 3 项研究均未讨论测试基准中的应用模型, 事实上很多现有的区块链测试基准要么采用简单的比特币转账交易场景, 要么直接沿用数据库测试基准的应用模型 (如 SmallBank 和 YCSD), 没有充分考虑区块链应用的特殊性. 此外, 这 3 项工作也未关注测试基准中数据模型的设置. 测试工具是测试基准的具体实现, 本研究重点关注测试基准的内容, 不对测试工具进行分析.

表 1 对本文工作和现有的区块链测试基准综述工作进行了对比. 表中的“✓”代表包含相应维度的分析, “—”代表不包含 (下同). 可以看到, 本文工作增加了应用模型、数据模型和执行规则这 3 个维度的讨论. 此外, 本文工作还将针对区块链应用的特殊性, 对现有的指标进行了详细的梳理和分析.

表 1 区块链基准综述研究

综述	区块链类型	应用模型	数据模型	负载	指标	执行规则
[18]	✓	—	—	✓	✓	—
[52]	✓	—	—	—	✓	—
[53]	✓	—	—	✓	✓	✓
本文	✓	✓	✓	✓	✓	✓

本文的主要贡献如下.

- 提出了一个区块链测试基准参考框架 UFBCB. 该框架结合了数据库的测试基准理论和区块链系统的特性, 明确了区块链测试基准所需考虑的关键要素及其相互之间的关联性.
- 设计了一个区块链度量指标体系. 通过对现有区块链评价指标的全面梳理, 我们从多个维度和层次对这些指标的定义及其计算公式进行了分类和总结, 构建了一个覆盖性能和安全性的全面的度量指标体系.
- 分析评估现有的区块链测试基准. 根据我们提出的区块链测试基准参考框架 UFBCB, 对现行的区块链测试基准进行了深入的对比分析. 比较了它们的主要特性、优缺点以及适用场景, 为未来的区块链测试基准的设计提供了参考和指导.

本文第 2 节介绍背景知识, 包括区块链技术和测试基准的基础概念, 并对常用的数据库测试基准进行分析. 第 3 节总结并分类目前用于评估区块链系统的各种指标. 第 4 节提出一个包含 5 个关键要素的区块链测试基准参考框架, 并对每一个要素进行深入的讨论. 第 5 节基于这些基本要素, 对现有的基准进行详细的比较和分析. 第 6 节对本研究进行总结和回顾.

2 背景知识

本节主要介绍区块链的基本知识, 并详细分析常见的数据库测试基准, 为深入探讨区块链测试基准奠定基础.

2.1 区块链

区块链本义是由“区块”组成的链, 每个区块保存了前一区块的哈希值, 用以验证其内容的正确性和完整性. 逻辑上多个区块就构成了一个单向链表. 由于经常被用来记录加密数字货币系统中的转账交易, 区块链也常被称为

分布式账本. 从计算机科学的角度, 区块链实质是一个特殊的分布式数据库, 采用共识协议来保证各个节点上数据的一致性, 能够容忍少量恶意节点, 并支持运行用户定义的程序 (智能合约).

本节将介绍区块链的背景知识, 包括核心数据结构、事务处理过程、共识协议和智能合约, 最后将介绍区块链所面临的安全威胁.

2.1.1 区块链数据结构

本节将对区块链的基础数据结构进行概述, 包括交易、区块以及状态数据库. 交易沿用了比特币系统的提法, 本质上是对状态数据库进行的一系列操作. 交易以区块为单位批量处理, 每个区块一旦在网络中达成共识, 所有的节点便会执行该区块内的所有交易, 并更新本地状态数据库. 这一过程确保了整个网络中的状态保持同步和一致.

交易是一个带有数字签名的字符串, 表示了区块链中一个事务, 描述了对状态数据库的一个或一组操作, 如资金转移、记录资产状态等. 数据库中用户提交 SQL 语句来执行操作, 而在区块链系统中, 用户则是通过调用智能合约来实现相应的功能. 区块链一个交易的内容通常是对智能合约的部署和调用.

图 1 展示了比特币的区块结构. 区块由区块头和区块体组成, 区块头包含了区块链版本号、前一个区块的哈希值、区块中所有交易数据的 Merkle 根哈希、时间戳等信息. 不同的区块链可能有不同的字段, 例如比特币的区块头还包含随机数 (Nonce) 和难度目标 (Bit), 以太坊和 Hyperledger Fabric 的区块头还存储了状态数据库的 Merkle 根 (State root). 区块的哈希值是通过对区块头进行哈希运算得到的, 可作为区块的唯一标识符. 每个区块保存前一个区块的哈希值, 以便于快速验证区块内容.

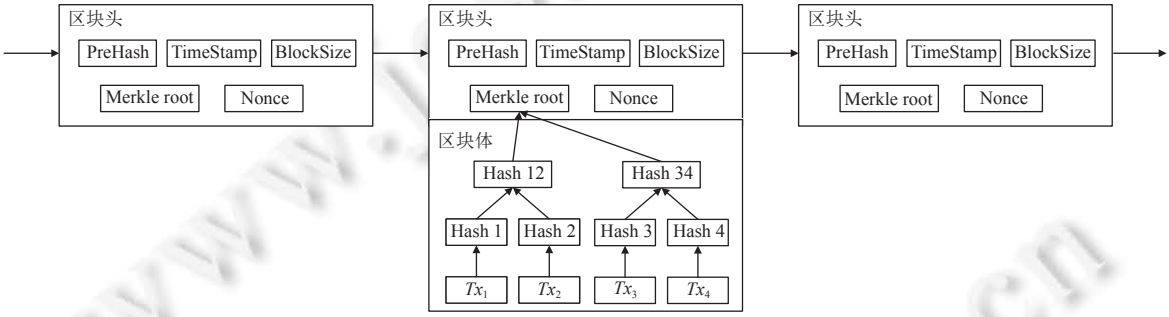

图 1 比特币区块结构

区块体是由多个交易组成, 根据这些交易的内容计算哈希值, 并根据它们在区块中的顺序进行串接运算, 即可得到在区块头中保存的 Merkle 根哈希. 如图 1 所示, 对区块体中所有交易信息, 按预设的 Hash 算法计算各交易的 Hash 值, 然后相邻的两个 Hash 值进行字符串拼接, 再进行哈希运算, 生成一个新的 Hash 值. 不断重复此过程, 最终形成一个根节点, 即 Merkle 根哈希. 为了克服传统 Merkle 树存储空间消耗和查询性能低下的缺点, 以太坊引入改进后的 Merkle Patricia 树 (MPT), 融合了 Merkle 树和前缀树的优点. 以太坊中的交易树、收据树和状态树都采用 MPT 结构. Hyperledger Fabric 为了减少添加数据的代价, 结合 Merkle 树和 Hash 桶两种数据结构的优点, 设计了 Merkle Bucket 树. Hyperledger Fabric 中的交易树和状态树都采用 Merkle Bucket 的结构. 这些改进的 Merkle 树及其变体在区块链中起到保证数据完整性和验证数据有效性的重要作用, 并提高了存储效率和查询性能.

区块链状态数据库保存了区块链所支撑的应用程序中各变量的当前状态, 比如加密数字货币系统中各账户的余额、拍卖程序中各用户的出价、物流溯源中各个货物的当前位置等. 比特币采用未使用的交易输出 (unspent tansaction output, UTXO) 模型来记录状态, 这与比特币的交易结构密切相关. 每执行一个交易, 就将消耗一个或多个 UTXO, 也会产生一个或多个新的 UTXO. 通过遍历 UTXO 集合, 可以计算出每个账户的余额. 以太坊采用账户模型来存储状态数据, 每个账户包括了余额、合约代码、存储数据等信息. 以太坊中所有的账户 (Key) 以及它们的状态 (Value) 构成了一棵 MPT, 树根 (状态 Merkle 根) 存储在区块头部. MPT 中包括叶子节点、分支节点和扩展节点这 3 种, 扩展节点包含了共同的 Key 前缀, 分支节点是一个长度 17 的 List, 前 16 个元素是 16 进制字符, 最

后一个元素代表了一个数值. 叶子节点是一个 Key-Value 键值对, 存储了账户状态数据. 以太坊将 MPT 转化为一系列的 K-V 对存储在 LevelDB 中. Hyperledger Fabric 中所有账户及其状态存储在 Merkle Bucket 树中. Merkle Bucket 由底层的 Hash 表和上层的 Merkle 树组成. 哈希表由一系列哈希桶 (Bucket) 组成, 状态数据被散列到每一个桶中, 并按序排列, 通过计算每一个桶的所有数据, 获得一个哈希值, 该哈希值作为上层 Merkle 树节点的叶子节点, 通过不断地迭代, 最终获得整棵树的根节点, 存储在区块头部. Hyperledger Fabric 将状态数据持久化存储到 LevelDB 或 CouchDB[54]中.

2.1.2 区块链事务处理过程

如前所述, 区块链系统中, 事务是以区块为单位批量处理. 为了保证各节点状态一致, 在事务处理过程中还需要通过共识来进行协调. 因此, 共识是事务处理过程的一部分. 如图 2 所示, 区块链的事务处理过程可以分为以下几个步骤.

- 交易发起: 客户端发起一笔交易, 该交易发送至区块链网络中的多个节点.
- 交易验证: 收到交易请求的网络节点需要对交易进行验证. 验证过程包括检查交易的格式、签名、权限和合法性等, 确保其符合预定义的规则和约束. 验证通过的交易被添加到节点的交易池中, 验证不通过的交易将被丢弃.
- 区块打包: 根据共识协议, 选取出块节点, 该节点从本地交易池中选择一部分交易, 打包成一个新的区块.
- 共识: 出块节点将生成的区块广播到网络中的其他节点. 网络中所有节点遵循某个共识协议对接收到的新区块进行一系列验证, 包括检查区块的哈希值、事务顺序、合法性等.
- 执行提交: 区块通过验证后将被添加到区块链中, 所有的节点都会执行区块中的内容并更新自己的本地状态数据库.

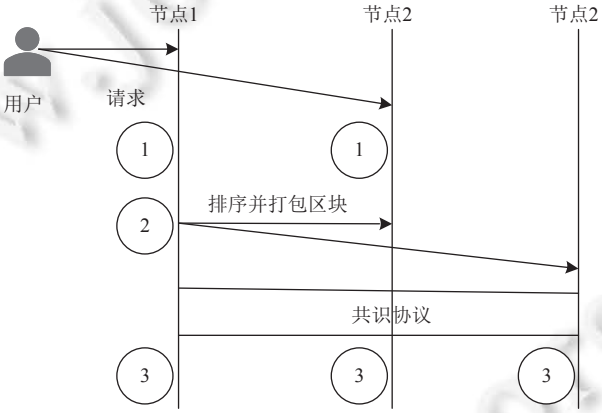

图 2 区块链工作机制

2.1.3 区块链共识协议

共识协议保障了区块链系统多个节点的一致性, 也是整个系统可信性和安全性的基石. 区块链的数据结构, 比如保存前一个区块的 Hash 以及本区块交易的 Merkle 根, 只能起到快速验证区块内容的作用. 区块链的防篡改能力根源在于共识协议.

一般将共识协议分为 PoX 系列和 BFT 系列, 前者常用于公链系统, 后者则广泛被许可链系统采用. 比特币系统采用了工作量证明 (proof of work, PoW) 机制, 由此衍生出系列 PoX 算法. PoX 系列算法核心在于通过某种机制来获得记账权, 一旦确定由某个节点发布账本 (即新的区块), 其他所有节点只需要验证区块内容的合法性, 不用进行投票. 由于区块发布之后没有投票确认环节, PoX 协议存在链分叉的问题, 即同一个高度可能存在多个区块. 1993 年, 美国科学家 Dwork 等人[55]首次提出了工作量证明这一思想, 主要用于应对垃圾邮件问题. PoW 的核心思想是通过分布式节点基于各自的计算机算力来解决一个复杂的数学问题, 从而获得下一个区块的记账权. 由于所选取的数学问题在理论上只能通过穷举来解决, 得到问题的解意味着付出了相应的工作量. PoW 安全性很高, 但

存在着耗费大量资源的问题. 2011 年, 一位名为 Quantum Mechanic 的网友在 Bitcointalk 论坛首次提出了权益证明 (proof of stake, PoS) 机制 (https://en.bitcoin.it/wiki/ProofofStake). PoS 是 PoW 的变种, 通过引入币龄这一概念, 在一定程度上解决 PoW 耗费大量算力的问题. 各个分布式节点的币龄越大, 获得记账权的可能性就越大. PoS 不再需要投入巨量资源来求解无实际意义的数学难题, 但币龄决定记账权也在一定程度上削弱了区块链去中心化特质. 目前以太坊系统已经从 PoW 全面转向 PoS.

另一类重要的共识协议称为 BFT (Byzantine fault tolerance) 协议. 此类协议的特点是一般不需要竞争记账权, 但是区块发布之后, 所有节点要通过多轮投票来决定是否接受. 与 PoX 不同的是, 区块一经上链就不会被分叉. BFT 协议的核心思想来源于 Paxos 协议[56], 通过多轮投票的方式达成共识. Paxos 采用简单多数的方式来确定投票结果, 也不考虑故意作恶的情况, 因此不能容忍拜占庭类错误. 尽管如此, Paxos 仍然是分布式系统的经典协议, 由 Paxos 衍生出很多协议, 如 Raft[57], 仍然被广泛用于分布式数据库系统、DNS 等应用中. 值得一提的是, 不少区块链系统出于对性能和安全的综合考虑也采用 Raft[57]. 1999 年, Castro 等人[58]提出了实用拜占庭容错 (PBFT) 算法, 通过调整投票确认机制以及引入 View Change 实现了容忍不超过 (N–1)/3 的拜占庭节点, 其中, N 为全部节点数. PBFT 是目前许可链主流的共识协议.

2.1.4 智能合约

智能合约由跨领域学者 Nick Szabo[59]在 1996 年提出, 含义是“一种可以实现合同条款的计算机程序”, 其执行过程可以不依赖于任何中间媒介. 智能合约的概念被提出之后, 技术层面无法实现不依赖第三方的执行合约程序, 因此一直未能真正地落地. 直到区块链技术的出现, 提供了一个去中心的分布式执行环境, 满足了智能合约的要求, 才使得此项技术得以真正地实现.

在区块链社区, 通常认为智能合约是运行在区块链系统之上的一段计算机代码, 由共识机制确保代码的正确执行[60]. 但多方参与的计算机程序, 在实际中往往意味着多方共同遵守的协议或规范, 仍然是广义“合约”的范畴. 目前, 智能合约没有统一的编程语言和运行模型. 比特币的智能合约采用了简单的脚本语言, 表达能力受限. 以太坊智能合约采用了图灵完备 Solidity 语言. 在 Hyperledger Fabric 中将智能合约称为链码 (chaincode), 对区块链的读写操作都需要经过链码来完成. 无论采用哪种语言, 智能合约都可视为一个运行在区块链平台上的状态机. 智能合约实现好合约内容, 当用户部署或调用交易时修改智能合约的值和状态, 智能合约生成区块, 对链上数据进行修改, 链上数据的修改会更新智能合约的世界状态. 而所有外部的数据源都应当来自可信的预言机 (oracle machine).

2.1.5 区块链安全威胁

根据被攻击的模块, 可以将区块链安全威胁分为攻击网络、攻击共识以及攻击智能合约这 3 类, 可能造成系统性能下降甚至崩溃、链分叉、网络分区以及用户财产损失[61,62]. 测试基准主要考虑针对网络或共识层面的攻击. 针对智能合约代码漏洞的攻击, 比如算数溢出、可重入漏洞等[63,64], 通常采用漏洞扫描工具来检测, 因此并不包含在测试基准中.

常见针对网络的攻击主要包括 DNS 劫持 (DNS hijack)[65]、日蚀攻击 (eclipse attack)[66]和 DDoS 攻击 (DDoS attack)[67,68]等.

(1) DNS 劫持: 新的节点加入区块链网络时, 依赖特定的 DNS 服务来发现已有的成员节点, 从而与区块链网络其他节点取得联系. 如果 DNS 服务器被劫持, 新的节点可能得到错误的成员节点地址, 也就不能正常参与共识.

(2) 日蚀攻击: 如果一个节点 A 的邻居全部为恶意节点, 这些恶意节点可以屏蔽 A 与网络其他节点的通信. 这样对于网络中其他节点来说, 节点 A 也就成为拜占庭节点. 日蚀攻击可以增大区块链网络中拜占庭节点的比例, 降低系统性能, 甚至造成系统崩溃.

(3) DDoS 攻击: 即分布式拒绝服务攻击, 这是常见的网络攻击方式. 通过利用合理的请求消耗节点资源, 使得节点不能对外正常服务 (比如参与共识). 针对区块链网络的 DDoS 攻击可能造成系统性能下降.

常见针对共识的攻击主要包括女巫攻击 (sybil attack)[69,70]、自私挖矿 (selfish mining)[71]、分叉攻击 (fork attack)[72,73]和提案延迟[74]等. 其中, 前 3 类攻击主要是针对公有链.

(1) 女巫攻击: 攻击者通过创建或控制多个账户来操纵共识协议的结果.

(2) 自私挖矿: 攻击者有策略地广播自己的区块, 以获取额外收益. 例如节点在成功获得区块后并不立即广播, 而是在发现网络中产生新区块再广播, 人为地造成分叉竞赛.

(3) 分叉攻击: 攻击者试图在区块链网络中引入分叉来篡改账本内容, 比如重复花费相同的数字货币.

(4) 提案延迟: 攻击者可以故意拖延共识的进程, 比如恶意延迟发送提案、延迟发送投票消息或发起错误的 Viewchange 投票等.

2.2 数据库测试基准

测试基准被广泛应用于软硬件产品的比较和方法评估, 文献 [75] 中将测试基准定义为“根据特定特征 (如性能、可靠性或安全性) 对竞争系统或组件进行竞争性评估和比较的标准工具”. 理想的测试基准具有 5 个特性: 相关性、可重复性、公平性、可验证性、经济性[76]. 测试基准必须以实验的可重复性为目标, 即详细描述各种部署条件和设置的能力, 以便其他人详细描述、扩展和重现它们.

数据库测试基准是指一套用于评测、比较在相同配置下不同数据库系统性能的规范, 使用基准所生成的性能指标值能够客观、全面地评测各个数据库系统的性能差距[77]. 根据应用场景的不同, 数据库的测试基准大致可分为在线事务处理 (OLTP) 和在线分析处理 (OLAP). 针对 OLTP 应用的基准包括威斯康星基准 (Wisconsin benchmark)[78,79]、AS3AP[80]、TPC-C[34]、TPC-E[36]和 OLTP-Bench[81]等. 针对 OLAP 应用的基包括 SetQuery[82]、TPC-H[37]、TPC-DS[83]和 SSB[84]等. 近年来随着 HTAP 数据库的发展, 出现了同时支持 OLTP+OLAP 的测试基准, 例如 CH-Benchmark[85]和 HATrick[86]等. 下面我们分别进行简要介绍.

2.2.1 面向 OLTP 的数据库测试基准

威斯康星基准是由威斯康星大学的 Bitton 等人[78]共同完成的测试基准. 该基准的数据模型包含 3 个表: ONEKTUP、TENKTUP1 和 TENKTUP2. 每张表都包含 13 个整数属性和 3 个字符串属性. 数据集设置如下: ONEKTUP 包含 1 000 条记录, TENKTUP1 和 TENKTUP2 各包含 10 000 条记录, 每条记录总长 208 字节. 数据均匀分布. 负载包含了 32 条 SQL 语句, 涵盖了 5 种基本操作, 包括: (1) 不同选择率下的选择操作; (2) 不同重复率下的投影操作; (3) 连接操作; (4) 简单聚集操作与聚集函数; (5) 插入、删除、更新等操作. 测试指标为执行 32 条 SQL 的总时长.

TPC-C 应用最为广泛, 其应用模型是一个大型商品批发销售公司的销售过程. 数据模型包含了仓库、区域、顾客、订单等 9 个表. TPC-C 定义了数据的分布和倾斜度, 并提供了一个数据集生成工具, 以确保数据的一致性. TPC-C 的负载包括 5 种类型的事务操作, 即订单管理、库存管理、支付等, 并明确了每种事务的比例和执行顺序. 例如, 订单管理和支付事务分别占据了 45% 和 43% 的负载. TPC-C 的主要性能指标是每分钟的新订单事务数 (transactions per minute, *TpmC*) 和性价比 (Price/Performance, 通常以 *Price*/*TpmC* 表示), 以评估系统的性能和成本效益. TPC-C 的执行规则明确了测试流程, 包括了数据导入、负载生成和执行、性能监控、环境清理等. 另外要求测试至少持续 2 h, 以保证测试的公平性和准确性.

TPC-E 是 TPC 组织在 2007 年 2 月推出的以证券交易应用为核心的基准. 它的数据模型比 TPC-C 复杂, 包含了 33 个表, 188 个字段, 表的类型分为 4 类: 与客户 (customer) 相关的 9 张表, 与经纪公司 (broker) 相关的 9 个表, 与市场 (market) 相关的 11 张表, 与维度 (dimension) 相关的 4 张表. 负载包括了交易查询、交易执行、交易结果更新、市场观察、证券信息等共 12 种事务. 测试指标包括性能指标 (*TpsE*, 每秒可以处理多少交易) 和性价比 (美元/*TpsE*). 执行规则要求每秒执行一次数据维护事务, 并在每次测试开始时执行一次交易清理事务.

2.2.2 面向 OLAP 的数据库测试基准

TPC-H 是一个基于商务采购应用的测试基准, 用于评估数据库管理系统在决策支持系统 (DSS) 场景下的性能. 其数据模型包含 Part、Region、Nation、Customer 等 8 张表, 代表参与商业领域中采购和订购的实体. 数据集由标度因子 (scale factor) 决定, 总的数据规模从 1 GB 到 100 TB 不等. 负载包括了 22 个复杂查询语句 (QUERY) 和 2 个更新数据语句 (RF1, RF2). 查询语句涵盖了各种复杂的决策支持查询, 例如销售分析、库存管理、供应链分析等. 指标为每小时内执行的查询数 (*QphH@size*). 其中, *QphH* 表示每小时查询数量, *Size* 表示数据集的规模.

执行规则要求测试过程中先执行 RF1, 然后执行 22 个 QUERY, 最后执行 RF2.

TPC-DS 是 TPC 在 2012 年发布的一个面向商品零售场景的测试基准. 其数据模型采用雪花模型, 包含 Store_Sales、Store_Returns、Catalog_Sales 等 7 个事实表和 Store、Call_Center、Catalog_Page、Web_Site、Web_Page 等 17 个维度表, 每个表包含 18 个字段. 数据集同样由标度因子决定, 具有 1T–100T 的多个不同规模. 负载包括 99 个随机可替换的 SQL 查询和数据维护任务 (DM). 查询语句包含多 Join 和聚合运算, 分析的数据量大, 旨在回答真实场景下的商业问题. 指标主要包括每小时内执行完毕的查询数 *QphDS@SF*、单位性能指标上的价格以及系统开放的可用时间. 执行规则要求基准测试按照如下的顺序进行: Load Test (数据加载测试)、Power Test (性能测试)、Throughput Test 1 (吞吐量测试 1)、Data Maintenance Test 1 (数据维护测试 1)、Throughput Test 2 (吞吐量测试 2)、Data Maintenance Test 2 (数据维护测试 2). 其中, Power Test 是单线程执行 99 个查询, 而 Throughput Test 是多线程并行执行查询, 相当于是压测.

2.2.3 同时支持 OLTP+OLAP 的数据库测试基准

CH-Benchmark 用于评估 HTAP 数据库在处理混合事务和分析查询时的性能. 它融合了 TPC-C 和 TPC-H 两种基准. 数据模型修改了 TPC-C 中的 9 个表和 TPC-H 中的 8 个表, 合并成了 12 个表, 并统一了数据集的伸缩模型. 负载包含了 TPC-C 中的 5 个事务以及 TPC-H 的 22 条查询, 删掉了更新语句 (RF1, RF2). 指标包括每分钟事务 (*TpmC*) 和每小时已完成查询 (queries per hour, *QphH*) 的指标. 基准测试的执行规则要求先仅测试 OLTP 或 OLAP, 然后混合执行 (OLTP+OLAP).

测试基准在数据库领域起到了至关重要的作用. 它提供了标准化的测试环境和指标, 可以客观地评估和比较不同数据库系统在各种应用场景下的性能. 通过这些基准, 开发者和系统管理员能够更深入地理解数据库系统的性能特性, 从而根据具体的应用需求进行优化和调整. 通过不断改进和完善测试基准, 可以促进数据库系统的发展和创新, 进而推动数据库技术的进步.

3 指 标

区块链度量指标是评估区块链系统质量、性能和可扩展性的基础, 主要分为 4 类, 如表 2 所示: (1) 性能指标, 包括吞吐率、延迟等; (2) 资源效率指标, 包括利用 CPU、内存、硬盘 IO、网络传输、功耗的效率; (3) 安全性指标, 指区块链部分节点出现拜占庭错误时系统性能受影响情况; (4) 扩展性指标, 指区块链系统在增加负载或参与节点时, 性能的变化情况. 这些指标为区块链系统在不同方面的表现提供了量化的衡量, 帮助我们了解系统的运行情况、性能瓶颈和潜在问题.

表 2 区块链评估指标

分组	指标	描述
性能指标	吞吐率	衡量区块链系统处理交易能力的指标, 通常以每秒处理的交易数 (TPS) 为单位
	延迟	衡量从提交交易到交易被确认所需的时间, 反映了区块链系统的响应速度
资源效率指标	CPU 效率	衡量区块链系统对计算资源利用效率的指标
	内存效率	评估区块链系统对内存资源的使用效率
	磁盘 IO 效率	衡量区块链系统对存储资源利用效率的指标
	网络传输效率	评估区块链系统对网络资源的使用效率
	功耗效率	评估区块链系统对能源的使用效率
安全性指标	孤儿块率	衡量区块链系统受到攻击时网络分叉率
	性能下降率	衡量区块链系统受到攻击时性能下降的程度
扩展性指标	吞吐率扩展指标	衡量区块链系统在增加负载或参与节点数量时, 吞吐率增加与节点数量增加的比值
	延迟扩展指标	衡量区块链系统在增加负载或参与节点数量时, 延迟时间增加与节点数量增加的比值

上述指标并不是完全独立的. 从指标定义可以看出, 性能指标定义了吞吐率和延迟, 不依赖于其他指标, 是整个指标体系的基础. 其余指标中, 只有孤儿块率和性能指标没有直接关联. 资源效率指标关注每单位吞吐率所消耗

的资源, 安全性指标中性能下降率评估在遭受安全攻击时对吞吐率和延迟的影响, 扩展性指标则考察节点数量或负载增加对吞吐率和延迟的影响. 因此可以认为资源效率、安全性和扩展性这 3 类指标都是从不同维度来考察被测系统要达到一定性能所付出的代价.

为便于量化分析, 我们为每个指标明确了相应的计算公式. 各指标公式中使用的符号和定义如表 3 所示.

表 3 符号及其意义

分组	符号	意义
通用符号	Tx	交易
	T	时间
	u	区块链中的一个节点
	N	区块链系统节点数量
	NB	区块的数量
指标符号	TPS	吞吐率
	TL	延迟
	TPC	CPU 效率
	TPM	内存效率
	$TPIO$	磁盘 IO 效率
	TPN	网络传输效率
	TPP	功耗效率
	DP	防御性能
	EP	扩展性能
系统资源符号	F	单个 CPU 核的频率
	U	CPU 使用率
	P	节点瞬时功率
	MEM	区块链程序使用的内存
	$DISKRW$	从磁盘读取和写入的数据量
	$TRANS$	网络中传输的数据量

3.1 性能指标

区块链的性能指标包括吞吐率 (transaction per second, *TPS*) 和延迟 (transaction latency, *TL*).

3.1.1 吞吐率

吞吐率 (*TPS*) 是区块链系统每秒能够成功执行的交易数量. 在一段时间 (ΔT) 内, 节点 u 的吞吐率 TPS_u 可以通过公式 (1) 计算.

$$TPS_u = \frac{|Tx \text{ in } \Delta T|}{|\Delta T|} \tag{1}$$

N 个节点在时间段ΔT 内的平均吞吐率为:

$$\overline{TPS} = \frac{\sum_u TPS_u}{N} \tag{2}$$

3.1.2 延 迟

延迟 (*TL*) 是指从交易首次发送到网络, 到交易被确认的时间间隔. 它反映区块链系统处理交易的效率. 单个交易 Tx 的延迟是指用户提交 Tx 的时刻到区块链网络确认 Tx 的时刻, 记为 TL_{Tx} 区块链网络在时间段ΔT 内的平均延迟可通过公式 (3) 计算.

$$\overline{TL} = \frac{\sum_{Tx \text{ in } \Delta T} TL_{Tx}}{|\Delta T|} \tag{3}$$

3.2 资源效率指标

资源效率指标衡量区块链系统消耗单位资源所完成的交易数量, 体现区块链系统资源利用的效率. 常见的资

源效率指标包括 CPU 效率、内存效率、磁盘 IO 效率、网络传输效率和功耗效率. 文献 [87] 中首次定义了区块链资源效率的计算公式, BCadvisor[44]基准中也采用了类似的公式, 在第 5.4.1 节中有详细的说明.

3.2.1 CPU 效率

CPU 效率 (*TPC*) 是指 CPU 单次有效脉冲所完成的交易数量. 它反映 CPU 在执行任务时的利用率和性能. 在 ΔT 时间段内, 节点 u 的 CPU 效率 TPC_u 可以通过公式 (4) 计算.

$$TPC_u = \frac{|Tx \text{ in } \Delta T|}{\sum_{t\in\Delta T}|F * U(t)|} \tag{4}$$

其中, F 是 CPU 的频率, $U(t)$ 是区块链程序在时刻 t 的 CPU 使用率. 考虑到网络中 CPU 的整体使用率, 可以取平均值:

$$\overline{TPC} = \frac{\sum_u TPC_u}{N} \tag{5}$$

3.2.2 内存效率

内存效率 (*TPM*) 是指区块链在使用内存资源方面的效率和性能. 它衡量区块链在处理数据时对内存的有效利用程度. 在 ΔT 时间段内, 节点 u 的平均内存效率 TPM_u 可以通过公式 (6) 计算.

$$TPM_u = \frac{|Tx \text{ in } \Delta T|}{\sum_{t\in\Delta T}|MEM(t)|} \tag{6}$$

其中, $MEM(t)$ 是在 t 时刻, 区块链程序使用的内存, 包括实际内存和虚拟内存. 当考虑整个网络时, 可以通过以下方式取平均值:

$$\overline{TPM} = \frac{\sum_u TPM_u}{N} \tag{7}$$

3.2.3 磁盘 IO 效率

磁盘 IO 效率 (*TPIO*) 是指区块链在进行磁盘 IO 操作时的性能和效能, 它衡量了区块链在读取和写入数据到磁盘时的速度和资源利用情况. 在 ΔT 时间段内, 节点 u 的磁盘 IO 效率 $TPIO_u$ 可以通过公式 (8) 计算.

$$TPIO_u = \frac{|Tx \text{ in } \Delta T|}{\sum_{t\in\Delta T}|DISKRW(t)|} \tag{8}$$

其中, $DISKRW(t)$ 表示在 t 时刻进行磁盘读取和写入的数据量. 为了对所有节点的磁盘进行概述, 通过以下方式取平均值:

$$\overline{TPIO} = \frac{\sum_u TPIO_u}{N} \tag{9}$$

3.2.4 网络传输效率

网络传输效率 (*TPN*) 是指区块链在进行网络数据传输时的性能和效能, 它衡量了区块链在通过网络发送和接收数据时的速度和资源利用情况. 在 ΔT 时间段内, 节点 u 的平均网络传输效率 TPN_u 可以通过公式 (10) 计算.

$$TPN_u = \frac{|Tx \text{ in } \Delta T|}{\sum_{t\in\Delta T}|TRANS(t)|} \tag{10}$$

其中, $TRANS(t)$ 表示在 t 时刻网络中传输的数据量. 为了考虑整个网络流量, 可以通过以下方式取平均值:

$$\overline{TPN} = \frac{\sum_u TPN_u}{N} \tag{11}$$

3.2.5 功耗效率

功耗效率 (*TPP*) 是指区块链节点在功耗方面的效率, 反映了区块链系统节点在执行其任务时所消耗的电能效率. 在 ΔT 时间段内, 节点 u 的功耗效率 TPP_u 可以通过公式 (12) 计算.

$$TPP_u = \frac{|Tx \text{ in } \Delta T|}{\sum_{t\in\Delta T}|P(t)|} \tag{12}$$

其中, $P(t)$ 是区块链程序在时刻 t 的瞬时功率. 当考虑整个网络时, 可以取平均值:

$$\overline{TPP}=\frac{\sum_u TPP_u}{N} \tag{13}$$

3.3 安全性指标

区块链常见的安全攻击模式包括 51% 攻击、女巫攻击、日蚀攻击、网络分区攻击等, 这些攻击行为通常对区块链造成两种后果: 一是网络分叉, 引发数据不一致、双重花费、数据篡改、资金损失等风险; 二是吞吐率下降, 引发交易延迟、业务中断等风险. 例如女巫攻击容易导致网络不稳定、分区行为, 从而使区块链分叉, 出现双重支付的风险. DDoS 攻击通过 TCP 或 Ping 洪水攻击耗尽区块链系统资源, 使其无法提供正常的服务, 降低系统的吞吐率. BlockBench[39]中首次提到了安全性指标的定义, 本研究在此基础上总结出使用孤儿块率和性能下降率两个指标来量化区块链的安全性, 这些指标可以帮助评估区块链系统在面对攻击时的稳定性和可靠性.

3.3.1 孤儿块率

在区块链中, 孤儿块是指被挖掘出来, 但未被接受的区块. 孤儿块与主链分离, 形成分支. 孤儿块率 (DP_F) 定义为区块链网络中的孤儿块数与主链中总块数的比值. 比值越低, 系统的安全性就越高, 越不容易受到攻击. 该指标可以通过公式 (14) 计算.

$$DP_F=\frac{|B_{\text{forks}}|}{|B_{\text{main}}|} \tag{14}$$

其中, $|B_{\text{main}}|$ 表示主链的长度 (即区块数量), 而 $|B_{\text{forks}}|$ 表示网络中所有孤儿块总数.

3.3.2 性能下降率

性能下降率 (DP) 定义为受到攻击时系统的性能与系统正常运行时的性能的比值, 可具体分为吞吐率比率和延迟比率两个指标.

吞吐率比率 (DP_{TPS}) 量化为受到攻击时的吞吐率与系统正常运行时吞吐率的比值. 比值越高, 系统的抗攻击能力越强. 该指标可以通过公式 (15) 计算.

$$DP_{TPS}=\frac{TPS_{\text{attack}}}{TPS_{\text{normal}}} \tag{15}$$

其中, TPS_{attack} 代表遭受恶意攻击时的吞吐率, TPS_{normal} 代表正常系统的吞吐率, DP_{TPS} 通过受攻击时系统的吞吐率与正常吞吐率的比值来衡量受到攻击时系统的安全性.

延迟比率 DP_{TL} 定义为受到攻击时的事务延迟与系统正常运行时的延迟的比例, 比值越低, 系统的抗攻击能力越强. 该指标可以通过公式 (16) 计算.

$$DP_{TL}=\frac{TL_{\text{attack}}}{TL_{\text{normal}}} \tag{16}$$

其中, TL_{attack} 代表遭受恶意攻击时的延迟, TL_{normal} 代表正常系统的延迟, DP_{TL} 通过受攻击时系统的延迟与正常延迟的比值来衡量受到攻击时系统的安全性.

3.4 扩展性指标

扩展性指标是用来衡量区块链随着节点和工作负载的增加, 系统性能随资源变化的情况, 可具体分为吞吐率扩展指标和延迟扩展指标两种. 在节点和工作负载增加的情况下, 系统如果能够保持较高的吞吐率, 并且延迟保持在可接受的范围内, 则认为系统具有较好的扩展性, 文献 [39] 中首次对区块链的扩展性能进行了评估. 文献 [39,46,88] 中提到扩展性是区块链的关键指标, 通过观察增加节点数时吞吐率和延迟的变化来进行衡量. 用户可以通过扩展性指标了解不同的区块链在相同的节点规模下的性能表现. 针对扩展性指标的测试也能展示区块链系统的性能拐点, 比如当节点规模增大到一定数值时系统性能出现急剧下降的情况[39].

3.4.1 吞吐率扩展指标

吞吐率扩展指标 (EP_{TPS}) 量化为系统扩展过程中增加的吞吐率与增加的节点数量之间的比值, 比值越高, 代表

系统的可扩展性越强. 该指标可以通过公式 (17) 计算.

$$EP_{TPS}=\frac{\Delta TPS}{\Delta N} \tag{17}$$

其中, ΔTPS 代表系统扩展过程增加的吞吐率, ΔN 代表区块链系统在扩展过程中增加的节点数量.EP_{TPS} 通过对比吞吐率的变化与节点数量的变化来衡量系统的扩展性, 该值越大, 系统的扩展性越好.

3.4.2 延迟扩展指标

延迟扩展指标 (EP_{TL}) 量化为系统扩展过程中增加的延迟时间与增加的节点数量之间的比值. 比值越低, 代表系统的可扩展性越强. 该指标可以通过公式 (18) 计算.

$$EP_{TL}=\frac{\Delta TL}{\Delta N} \tag{18}$$

其中, ΔTL 代表系统扩展过程增加的延迟时间, ΔN 代表区块链系统在扩展过程中增加的节点数量.EP_{TL} 通过对比延迟的变化与节点数量的变化来衡量系统的扩展性, 该值越小, 系统的扩展性越好.

3.5 小 结

本节主要针对区块链中总体性能、资源消耗、安全性和扩展性的指标做了说明和定义, 通过这些指标可以对区块链系统进行全面详尽的衡量, 在测试基准中可根据负载的应用场景选择不同的指标. 公有链和许可链在目标用户和应用场景上存在显著差异, 并且采用不同的共识协议, 因此它们的评测指标体系也有所不同. 对于公有链, 特别是采用 PoW 共识协议的区块链系统, 评测指标通常会包括功耗. 对于许可链, 评测指标中较少考虑资源效率.

4 区块链测试基准参考框架——UFBCB

本节将介绍本文提出的区块链测试基准参考框架 UFBCB (unified refrence framework of blockchain benchmark). 如图 3 所示, 该框架主要由 5 个要素组成: 应用模型、数据模型、负载、指标和执行规则. 其中, 数据模型是根据应用模型生成的, 负载则是参照应用模型结构进行设计的. 在实施测试过程中, 根据执行规则, 导入数据集, 执行负载测试, 对被测试系统进行评估, 从而得到相应的测试指标数据. 基于该框架, 可以针对不同类别的区块链系统设计测试基准. UFBCB 并非具体的测试基准, 而是对区块链测试基准的抽象. 类似 OSI 参考模型, UFBCB 描述了区块链测试基准的组成要素和结构. 现有的区块链测试基准都可以看作是这个参考框架的实例. UFBCB 具有很高的灵活性和扩展性, 可以适应不同的区块链应用场景.

图 3 测试基准参考框架 UFBCB

本节将采用一个基于区块链的供应链管理例子来介绍各个测试要素.

供应链管理覆盖了产品从原材料阶段到最终交付给消费者的全过程. 在这个过程中, 区块链和智能合约的应用显著提升了供应链管理的透明度, 增强了安全性, 降低了成本, 增强了多方互信. 沃尔玛与 IBM 联合推出的“从农场到餐桌”的食品溯源和真实性验证方案, 很好地展示了区块链技术在食品供应链中的实际应用. 供应链管理涉及供应商、制造商、分销商、零售商、仓储和物流等各个参与方, 主要的操作包括采购管理、生产管理、库存管理和订单处理等关键步骤.

4.1 应用模型

应用模型是指对特定区块链业务场景的描述. 应用模型是数据模型、负载、指标和执行规则的基础, 旨在模拟现实世界中的应用, 以便评估区块链系统在类似条件下的性能. 以供应链场景为例, 应用模型包含供应商、客户、订单、产品、库存和发货等实体, 以及订单处理、库存管理、产品配送、退货处理等业务逻辑, 可模拟从原材料采购到产品制造, 再到产品零售的整个业务流程.

4.2 数据模型

区块链基准的数据模型是用于描述和组织区块链系统中数据的结构和关系的一种模型. 如图 4 所示, 数据模型包括模式和数据集, 其中数据模式涵盖了区块链系统中的各种实体 (例如, 区块、交易、智能合约等) 的相互关系, 定义了数据的存储和组织方式, 规定了如何通过交易和智能合约来访问和修改这些数据. 数据集是指用于基准测试的数据集合, 它设定了基准测试的初始条件. 统一的数据集便于不同的区块链之间进行公平的评估. 本节将针对区块链基准中的数据模式和数据集的设计进行分析.

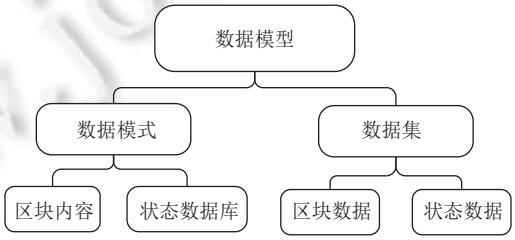

图 4 数据模型的组成

4.2.1 数据模式

区块链数据主要分为区块数据和状态数据, 区块数据包括了区块头的元数据和交易序列数据, 数据无法被篡改. 状态数据记录了区块链上的世界状态, 状态数据可以被创建、更新和删除. 区块数据通常以文件形式存储, 而状态数据则被持久化到数据库中. 因此区块链测试基准中的数据模型主要包括两部分: 区块内容的设置和状态数据库的设置.

(1) 区块内容: 根据第 2 节中的介绍, 区块是一组交易的集合, 包含了区块头和交易列表. 不同的区块字段可能影响区块的大小和事务处理的复杂度, 从而影响区块链的性能, 因此基准中需要说明对这部分字段的设置. 大多数的区块链中, 区块头字段是由区块链协议规定的, 遵循特定的标准. 因此基准中一般不需要考虑区块头的设置.

交易字段的内容一般包括发送者地址、接收者地址、交易金额、签名、时间戳等基本信息. 此外, 大多数的区块链中定义了一个数据字段用于存储自定义数据. 例如, Hyperledger Fabric 的交易数据字段 (payload) 可以自定义与业务相关的数据, 如调用的智能合约名、合约的方法名、传入的参数列表、交易类型、交易 ID、交易的时间戳等.

另外, 每个区块中包含的交易数量对区块上链的速度有重大的影响, 因此区块链基准中需要对上链数据和区块的包含交易数量做明确的要求. 以供应链应用场景为例, 上链的数据至少应包括如下信息. ① 区块元数据: 区块版本、前一个区块的哈希值、Merkle 树根哈希、区块高度等, 用于确保区块链的完整性和一致性. ② 权限信息: 发送者身份和签名、背书人身份和签名, 用于验证交易的合法性和安全性. ③ 交易数据: 商品的订单信息、生产

信息、物流信息、质检报告、交易的时间戳等, 用于记录交易的具体操作和上链的业务数据, 实现溯源和透明度. ④ 结果数据: 智能合约执行状态代码、执行结果、错误信息等. 这些数据用于记录智能合约的执行情况和结果.

图 5 是一个简单的智能合约片段, 实现了供应链应用中订单创建的功能. 其中定义了一个名为 Order 的结构体, 用于存储订单 ID、产品名称、产品数量、买家地址、卖家地址等信息. 函数 CreateOrder 用于创建订单, 接收产品名称、数量和卖家地址作为参数, 并将订单存储在 Orders 映射中, 实现订单的存储和检索. 当调用智能合约的方法时, 函数的参数 (订单 ID、产品名称、数量等) 会被记录到区块链中. 此外, 交易的发起者信息、交易的时间戳、元数据信息等也会被记录到区块链中.

```
contract SupplyChain {
    struct Order {
        uint256 orderId ;
        string product;
        uint256 quantity;
        address buyer;
        address seller;
        bool is Created ;
        string status;
        unit256 timestamp;
    }
    uint256 private orderldCounter;
    function createOrder (string memory _product, uint256 _quantity, address _seller) public {
        require (bytes (_product). length >0 && _quantity >0 && _seller !=
        address (0), "Invalid parameters");
        orderldCounter++;
        orders [orderIdCounter]=Order (orderIdCounter, _product, _quantity,
        msg.sender, _seller, true, "Created", block .timestamp);
        emit OrderCreated(orderldCounter,_product, _quantity, _seller,
        block .timestamp);
    }
}
```

图 5　智能合约举例

(2) 状态数据库: 区块链的状态数据库是一种特殊的数据结构, 记录了区块链网络中所有账户的状态信息, 并随着交易的执行而不断变化. 状态数据库的设置包括了数据模式的设置.

数据模式是对数据库中数据的组织、管理和存储方式的描述, 它定义了数据的基本结构、属性、关系和约束, 是设计和构建状态数据库的基础. 状态数据库中的数据模式设置对某些应用场景的性能有较大的影响. 因此, 在测试基准中需要明确说明数据模式.

下面我们以供应链应用为例说明数据模式的设置. 在供应链应用场景中, 状态数据库包含了各种实体的信息, 如订单信息、产品信息、供应商信息、物流信息、仓库信息等. 表 4 和表 5 分别展示了产品和订单实体中属性的设置, 其中, productId 和 orderId 可以设置主键索引, 用来检索、更新或删除实体信息.

表 4　供应链中产品实体的属性设置

字段名	类型	描述
productId	String	产品的唯一标识符
name	String	产品名称
description	String	产品描述
manufacturer	String	制造商
price	Number	产品价格

表 5　供应链中产品订单的属性设置

字段名	类型	描述
orderId	String	订单的唯一标识符
productId	String	对应的产品标识符
amount	Number	订单金额
orderDate	Date	下单日期
status	String	订单状态

4.2.2 数据集

在测试基准中, 数据集是指用于测试和评估系统性能的数据集合, 用于模拟实际应用场景中的数据, 按照应用模型定义的数据结构存储. 为了模拟真实应用场景, 基准应提供与真实数据在数据量、数据分布、数据类型和数据关联等方面高度相似的数据集. 数据集的选择和设计应根据具体的测试目标和应用场景进行定制, 以确保测试结果的可靠性和实用性. 区块链的数据集包括区块数据和状态数据. 每个区块分别由一组交易组成. 状态数据库的值应等于在初始状态 (如空的数据库) 上面顺序执行每个区块的结果. 以供应链应用为例, 区块数据包括交易信息、时间戳、区块头等数据, 其中, 交易信息包括订单、物流、支付等操作. 状态数据包括订单状态 (如已下单、已发货、已交付)、物流状态 (如在途、已签收)、产品信息 (如产品的属性、规格、批次号、生产日期) 等.

4.3 负 载

负载是指测试时对被测系统施加的一系列交易. 区块链的工作负载是一组模拟真实场景或特定使用情况的交易集合, 旨在评估系统在不同业务需求下的性能表现. 与数据库不同, 区块链的交易是对特定智能合约的调用, 因此区块链的负载应说明相应智能合约的逻辑. 负载是测试基准的核心内容, 区块链测试基准应明确定义负载使用的合约, 包括每个合约的对外接口, 如变量和函数, 其中函数部分需列出每个函数的参数、返回类型、处理过程、发出的事件以及调用的外部合约[88]. 下面将从负载类型和负载特征对基准的负载设计进行描述, 如图 6 所示.

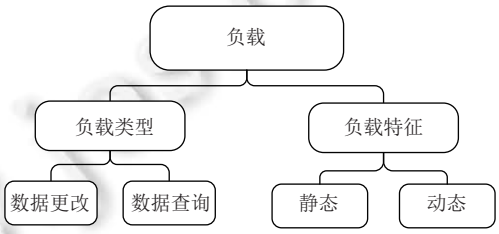

图 6 负载的组成

4.3.1 负载类型

常见的区块链负载可以分为两类: 数据更改负载和数据查询负载, 如图 6 所示. 这两种负载也是数据库测试基准中常见的类型, 分别对应着 OLTP (联机事务处理) 业务和 OLAP (联机分析处理) 业务.

(1) 数据更改负载: 模拟真实的交易和操作, 包括数据插入、数据更新、数据删除等, 类似数据库的 OLTP 负载. 数据更改负载可以根据不同的应用场景和需求而有所不同. 例如, 在供应链应用中, 数据更改负载包括订单的创建、订单状态的更新、物流信息的更新、库存数量的更新等. 在金融领域的应用中, 数据更改负载可能主要包括转账和支付等操作. 通过模拟不同类型的数据更改负载, 可以评估区块链系统的在处理事务操作时的性能和吞吐率.

(2) 数据查询负载: 模拟对区块链数据进行分析和复杂查询的场景, 包括复杂查询、统计分析和数据挖掘等操作. 复杂查询是指模拟对区块链数据进行复杂的查询操作. 例如, 在供应链应用场景中, 查询特定时间范围内销售额高于 500 万元的供应商的名字和地址; 在公安多维布控场景中, 根据时间、地区、车牌号与车辆品牌查询重点人员. 统计分析是指模拟对区块链数据进行统计分析, 计算交易数量、平均交易金额、交易频率等. 例如, 在医疗应用中, 根据患者的生理指标进行病理分析等; 在供应链场景中, 查询特定时间范围内计算某个产品的平均销售量. 数据挖掘是指模拟对区块链数据进行数据挖掘操作, 发现交易模式、关联规则、异常检测等. 例如, 在电商应用场景中, 基于销售数据挖掘销售产品之间的关联关系, 或检测异常交易行为. 这些操作可以涉及多个区块和交易, 并需要对数据进行聚合、过滤、排序和计算等处理, 类似于数据库的 OLAP 负载. 通过模拟不同的分析负载, 可以评估区块链系统在处理复杂查询和分析任务时的响应能力.

4.3.2 负载特征

负载的特征包括静态特征和动态特征.

(1) 静态特征指的是负载在某个时间点上的固定属性或状态, 包括交易种类和比例、交易频率和数据大小等, 描述了负载的基本属性.

● 交易种类和比例: 描述不同类型的交易操作, 以及每种交易所占的比例, 反映了应用场景的主要操作类型. 在企业调研报告中显示, 有些应用场景以插入和更新为主, 例如航天信息的交通运输场景, 有些应用场景以查询为主, 如北京大数据中心的政务场景. 在供应链应用场景中, 交易操作类型主要包含订单创建、订单状态更新、物流信息更新和库存数量更新等, 每一种交易类型比例可能是不同的.

● 交易频率: 在一定时间内交易的提交速率. 例如, 每分钟提交的订单数量或每小时更新的物流状态数量.

● 数据大小: 交易中涉及的数据大小, 即每个交易中包含的信息量. 例如, 订单交易中包含的产品数量、物流信息交易中包含的位置数据等.

● 并发用户的数量: 确定参与测试的客户端数量. 客户端的数量会影响系统的负载和性能表现, 需根据业务应用场景进行合理设置.

(2) 动态特征指的是负载在一段时间内的变化, 包括负载频率变化、峰值负载和负载类型变化等特性, 描述了负载的变化模式.

● 负载频率变化: 描述交易提交频率随着时间发生的变化. 例如在供应链场景中, 节假日或者特殊促销期间, 交易的提交频率会增加, 而在工作日, 交易的提交频率可能相对较低.

● 峰值负载: 描述负载的最高水平, 即在某个时间点上负载达到的峰值. 例如在电商应用场景中, 每年的双 11 购物节零点, 订单数量会激增出现峰值负载. 股票应用场景中, 每天早晨 9 点出现交易峰值.

● 负载类型变化: 交易类型随着时间发生的变化. 例如在银行的营业时间内, 负载的类型以转账、支付、存款和取款等在线交易为主, 而且非营业时间或深夜时段, 负载的类型批量更新、对账清算和报表分析等交易为主.

测试基准的负载需要同时考虑负载的静态和动态特征, 以准确地模拟真实场景下的负载.

4.4 指　标

区块链测试基准中的指标, 主要包含指标定义和指标计算两部分, 如图 7 所示.

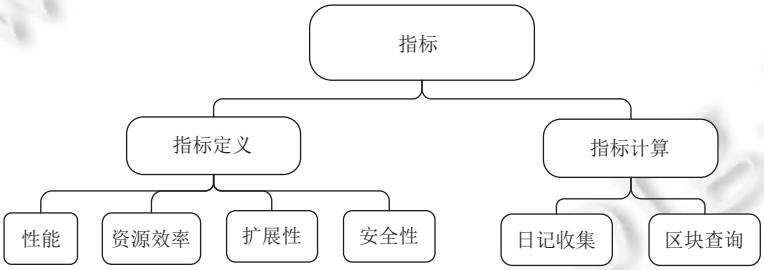

图 7　基准指标的组成

4.4.1 指标定义

根据第 3 节的内容, 将区块链的指标划分为 4 个主要维度: 性能、资源效率、安全性和扩展性. 这些指标描绘了区块链系统在各个方面的表现, 为我们评估系统的性能和可行性提供了依据. 在设定区块链的测试基准时, 我们可以根据业务场景和需求, 选择一组或多组度量指标来衡量待测系统的性能. 以金融支付应用场景为例, 吞吐率和延迟就是关键的指标. 由于系统需要处理大量的交易, 我们可以通过测试每秒处理的交易数量 (*TPS*) 来评估系统的吞吐率. 同时, 通过测试交易确认所需的时间, 我们可以评估系统的响应速度和延迟.

4.4.2 指标计算

区块链的交易与传统数据库的交易有一些不同之处. 区块链的交易通常需要经过一定的时间和一定数量的区块确认才能被视为有效和不可逆转. 因此, 区块链的指标计算过程相对于数据库来说更加复杂. 目前, 计算区块链指标的方式主要包括日志收集和区块查询两种方法. 这两种方法都在真实的区块链网络中进行, 以收集系统的性

能数据和资源使用情况.

(1) 日志收集: 区块链的日志记录了系统的各种操作和事件, 包括交易处理时间、资源消耗和错误信息等. 通过对区块链日志进行分析, 可以获取系统的性能数据. 例如, 通过分析某段时间内上链的区块数以及区块内的交易数, 可以计算出吞吐率等指标.

(2) 区块查询: 区块查询是一种通过与区块链节点进行交互, 获取统计数据, 然后进行深度分析和计算以得出关键性能指标的方法. 比特币和以太坊等主要的区块链平台都提供了 JSON-RPC 接口, 允许开发者发送 RPC 请求来查询区块链数据和交易数量等. 利用这些数据, 可以计算出系统的指标. 例如, 在比特币中, 可以使用 GetblockRPC 调用来获取某个区块的详细信息, 其中包含了交易数量. 通过计算特定时间段内所有区块的交易数量除以时间长度, 就可以获得系统的吞吐率.

这两种方式都可以用于实际测试中, 以获取区块链系统的性能数据和指标. 区块查询可能会对系统的性能产生一定的影响, 因此基准设计时需要综合考虑这些因素.

4.5 执行规则

执行规则规定了如何进行测试, 以确保正确地获取相应的测试指标, 同时保证测试的准确性、重复性和可比性. 在区块链中, 执行规则包括区块链测试配置和测试流程. 如图 8 所示.

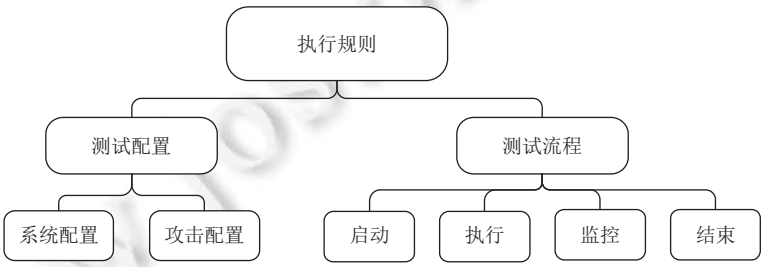

图 8 执行规则的组成

4.5.1 测试配置

测试配置包括了系统配置和攻击配置.

(1) 系统配置. 系统配置是指针对被测区块链系统进行的环境部署配置过程. 系统配置包括了硬件配置和软件配置. 硬件配置是指区块链运行的硬件环境, 包括架构 (X86/ARM)、处理器速度、CPU 数量、内存大小和硬盘的容量以及配置等. 软件配置主要包括区块设置、网络设置等. 区块设置主要包括共识协议、区块大小、区块生成间隔等参数配置; 网络配置主要包括网络的拓扑结构以及节点的设置、数量、连接等. 不同的配置对区块链的性能影响较大[89], 基准应详细描述系统各种配置, 以保证实验的可重复性和公平性. 有些基准[45]通过 YAML 配置文件进行系统配置, 其中包含全局配置、节点配置、网络拓扑配置和部署配置这 4 个部分. 配置文件的结构和内容应根据具体的测试场景进行定义和修改.

(2) 攻击配置. 由于区块链系统必须支持拜占庭容错, 区块链的测试基准还应包括攻击配置, 以模拟真实环境中存在的安全威胁, 全面评估区块链系统的性能、安全性和稳定性. 攻击配置是指在区块链系统中模拟真实故障和攻击情况, 旨在评估系统在面对故障和攻击时的抵御能力. 常见的攻击配置主要有节点故障、网络分区和恶意攻击. 节点故障是指在事务负载运行期间, 通过关闭一个或多个节点, 模拟节点宕机或崩溃的情况, 观察系统的响应和数据同步情况. 网络分区是指在事务负载运行期间, 通过网络命令将网络分割成多个部分, 模拟网络分区, 观察区块链分叉情况. 恶意攻击配置是指在事务负载运行期间, 通过模拟自私挖矿、日蚀、分叉、女巫、DDoS 攻击等恶意行为, 观察系统的吞吐率变化和延迟变化. 通过模拟这些故障, 可以评估区块链系统在面对故障和攻击时的抵御能力和稳定性.

4.5.2 测试流程

测试流程定义了明确的测试流程和步骤, 确保测试的一致性和可重复性, 主要包括测试的启动、执行、监控

和结束等阶段的设计.

以供应链的应用场景为例, 介绍测试流程各阶段的任务. 图 9 显示了一个简化的流程图.

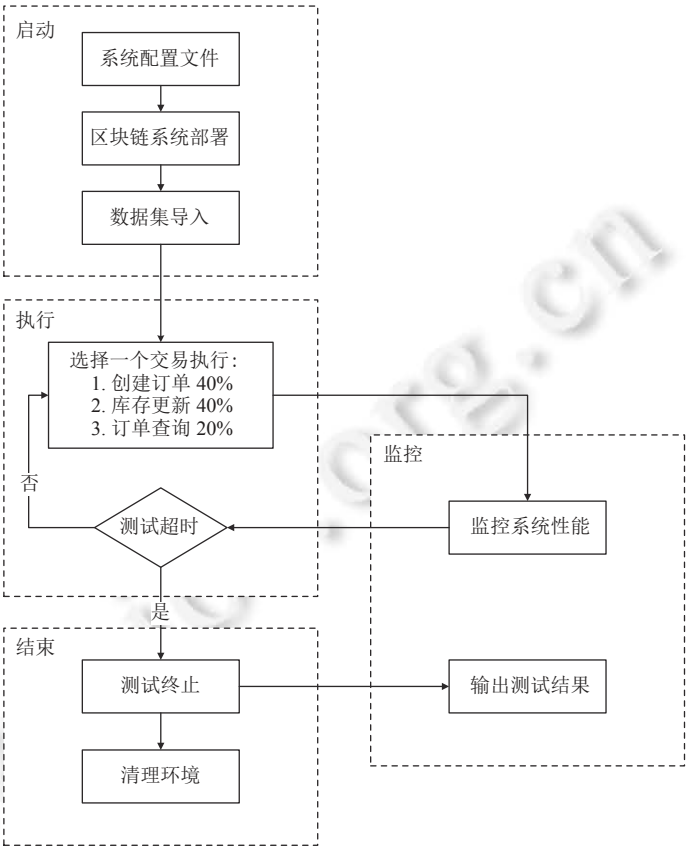

图 9 测试流程案例

(1) 测试的启动阶段, 按照系统配置文件搭建测试环境, 包括设置网络节点、定义智能合约以及初始化区块链状态.

(2) 测试执行阶段, 按照负载配置启动测试, 执行区块链交易, 如创建订单、库存更新、订单查询等. 这些交易通过智能合约在区块链上执行, 触发状态的更新.

(3) 测试监控阶段, 在测试执行的过程, 监控并获取系统指标和资源利用情况, 包括区块链的吞吐量、延迟、资源消耗, 扩展指标等.

(4) 测试结束阶段, 停止负载, 清理环境, 收集整理测试结果并生成测试报告.

4.6 小 结

本节介绍了区块测试基准的主要要素, 包括应用模型、数据模型、负载、指标和执行规则. 由于区块链系统的网络环境存在不稳定性和恶意攻击, 区块链测试基准中应考虑攻击设置. 另外由于区块链的系统设置对区块链的性能影响较大, 因此基准设置包含了系统的配置过程.

5 区块链测试基准分析

本节将基于第 4 节提出的区块链测试基准参考框架对现有的区块链测试基准进行全面的分析比较, 即分别从应用模型、数据模型、负载、指标和执行规则来衡量现有区块链测试基准的合理性.

除区块链测试基准之外, 还有很多工作通过实验分析或模拟对区块链性能进行测试评估. 实验分析是通过设计和执行一系列的实验, 衡量不同参数下系统的性能. 文献 [90−95], 针对有许可的区块链平台进行了实证性能评估, 通过实验的方式对多个区块链平台进行性能测试. 但是, 实验环境的特定性和参数的复杂性导致实现的结果存在不可重复性. 模拟是通过建立区块链模拟器, 模拟区块链的交易过程来衡量区块链的理论性能. BlockLite[96]和 VIBES[97]都是公开的区块链模拟器, 可以在单个节点模拟区块链的基本操作, 评估区块链的扩展性等. 模拟器通常在理想的或者受控的环境中运行, 与区块链实际运行环境存在一定的差别, 无法完全反映区块链系统在实际环境中的性能.

本文主要针对系统级的测试基准进行研究. 测试基准是对不同架构的系统进行统一评估的规范和工具, 具有相关性、可重复性、公平性、可验证性和可用性[76]. 根据第 4 节介绍的区块链测试基准参考框架, 我们对目前的区块链测试基准进行了对比分析, 结果如表 6 所示.

表 6　区块链基准对比

基准	支持类型	应用模型	数据模型	负载	指标	执行规则
BlockBench[39]	PR	✓	✓	✓	✓	✓
Hyperledger Caliper[40]	PR	–	–	–	✓	✓
C2B2[98]	PR	–	–	–	✓	✓
BlockMeter[48]	PR	–	–	✓	✓	✓
Klenik等人[50]	PR	✓	✓	✓	✓	✓
BBSF[88]	PR	✓	✓	✓	✓	✓
Touloupou等人[99]	PU/PR	–	–	✓	✓	✓
Diablo[42]	PU/PR	✓	–	✓	✓	✓
DAGBENCH[46]	PU/PR	✓	✓	✓	✓	✓
Gromit[43]	PU/PR	✓	–	✓	✓	✓
BCTMark[45]	PU/PR	–	–	✓	✓	✓
BCadvisor[44]	PU/PR	–	–	✓	✓	✓
xBCBench[47]	PU/PR	✓	–	✓	✓	✓

表 6 “支持类型”中, PU 表示该基准评测了公有链, PR 表示该基准评测了许可链. 可以看到, 超过半数的区块链测试基准同时评测了公有链和许可链. 从评测内容来看, 大部分基准在测试公有链和许可链时采用了相同的指标和执行规则, 只有 BCTMark 采用了不同的指标. 另外我们也注意到, 只测试许可链的基准, 如 BlockBench 和 BBSF, 没有测量资源效率. Diablo 主要针对的也是许可链, 同样没有测资源效率. 可见公有链和许可链在评测指标上应该有所区分. 此外, 公有链和许可链面临的安全威胁也有差异, 测试时也应该采用不同攻击配置, 目前只有 BCadvisor 对此有针对性设计. 我们将在第 5.4 和 5.5 节详细介绍.

5.1 应用模型

现有的区块链测试基准中, 针对应用模型的研究比较少. Diablo[42]开发了 5 个去中心化的应用程序, 涵盖了交易所、视频分享、游戏、网站服务和移动应用等业务场景. 交易所在早晨 9 点开市时会出现交易高峰, 因此其应用场景呈现出峰值特征. 移动应用选择了 Uber 的计算距离场景, 通过计算欧几里得距离来匹配司机和乘客, 因此该场景具有计算密集型的特征. 网站应用场景模拟的是 1998 年足球世界杯期间对 FIFA 网站的访问情况, 因此该场景具有高竞争性的特征. 视频分享和游戏应用场景则都具有高请求率的特征. 然而该研究并没有深入分析这些业务场景是否真实反映了区块链应用的实际需求.

BBSF[100]定义了 4 个区块链应用场景, 包括代币交易所、NFT 市场、NFT 铸造以及体育博彩网站. 该研究没有深入分析每一个应用场景具体的特点.

此外, 大多数的基准并没有考虑应用模型的设计, 例如, DAGBENCH[46]、Gromit[43]和 xBCBench[47]等测试基

准主要将资产转移 (即转帐业务) 作为其操作负载的核心, 并没有介绍复杂的应用模型的设计. BlockBench[39] 采用了 3 个广受欢迎的以太坊智能合约 EtherId、Doubler 和 WavesPresale, 以及两个数据库基准 YCSB 和 Small Bank 作为工作负载. 然而, 它并未详细介绍这些工作负载的应用场景设计.

针对目前的基准分析, 将区块链基准的应用模型大体分为以下 3 类. 表 7 展示对目前基准的分析.

表 7 应用模型对比分析

基准	记账应用	复杂业务	基本操作
BlockBench[39]	✓	✓	✓
Klenik等人[50]	✓	—	—
BBSF[88]	✓	✓	—
Diablo[42]	✓	✓	—
DAGBENCH[46]	—	—	✓
Gromit[43]	—	—	✓
xBCBench[47]	—	—	✓

第 1 类: 记账应用, 包括交易所、NFT 市场、NFT 铸造以及 Small Bank 等场景. 此类场景特征是数据模式简单, 合约逻辑简单, 操作简单, 可能某个时间段出现峰值交易.

第 2 类: 复杂业务应用, 包括游戏, 视频分享和移动应用等. 这些场景通常具有复杂的数据模型和操作流程, 并且可能包含计算密集型的交易处理.

第 3 类: 基本操作类应用, 包括键值存储 (YCSB)、资产转移和算术运算等. 这些场景以简单的操作集合为特征, 缺乏复杂的业务逻辑, 主要用于基础性能的评估.

5.2 数据模型

区块链测试基准的数据模型主要包含数据模式和数据集.

目前对数据模式的设置有两种方式, 一是基于数据库基准中的数据模式进行修改, 二是基于现有区块链中流行的智能合约进行修改.

BlockBench[39]基于 Small Bank 场景设计了一个转账的智能合约. 合约中定义了账户表、储蓄表和支票表这 3 个表, 存储的键值由表名和账户 ID 组合而成. 利用 ChaincodeStub 接口 GetState 和 PutState 方法来获取和更新账户的状态. 此外, BlockBench 使用了 3 个以太坊流行的智能合约 EtherId、Doubler 以及 WavesPresale 设计交易内容, 并根据被测系统进行了调整, 例如在 Hyperledger 中, EtherId 合约创建了两个不同的键值命名空间. 第 1 个用于存储域名数据结构, 在域名创建中, 合约将域名值插入此命名空间, 并使用域名作为键. 第 2 个命名空间用于存储用户账户余额, 使用表名和账户名作为键. 在进行域名转让时, 交易需传入发送者、购买价值、域名、新价格和新所有者的信息等参数.

文献 [50] 基于数据库基准 TPC-C 来构造合约和交易, 将 TPC-C 的关系数模式转换映射到 Hyperledger Fabric 键值数据库中实体. 键空间沿着“数据库表”进行分片, 使用数据库表名作为前缀来区分实体类型. 每个实体的键是由主键连接而成的, 例如一个仓库键表示为: concat(‘WAREHOUSE’, w_id), 其中, w_id 是仓库实体的唯一主键.

数据集主要包括区块 (含交易) 数据和状态数据库. 现有的区块链测试基准中大多描述了区块交易, 对状态数据库讨论较少. BlockBench 通过实现了一个键值存储的智能合约, 利用 YCSB 驱动的负载客户端预加载若干记录, 用于读写测试. DAGBENCH[46]在执行查询负载以前, 预加载了不同数量的交易作为测试数据集. 对于数据加载的方式, 智能合约的实现没有详细的描述. BBSF 中提到测试基准中需要设置测试的起始条件, 包含测试数据集, 钱包与合约的初始余额和初始状态以及外部结构 (变量、函数、结构) 的初始值等. 由于数据集特别是状态数据库大小对查询性能影响较大, 如果测试基准面向查询为止的应用场景, 应该包含数据集的具体参数.

5.3 负 载

区块链测试基准的负载描述应该包含负载的内容和负载的静态动态特征描述. 目前大多数区块链基准定义了

工作负载, 但以数据更改的负载为主, 对负载静态和动态特征的规定也存在不同程度的缺失, 如表 8 所示. 另一方面, 现有区块链测试基准采用了不同的负载生成方式: 一是借鉴和利用数据库测试基准的负载生成方式; 二是从区块链的实际应用中, 抽象出操作集合作为负载; 三是使用简单的负载生成器产生负载. 下面详细介绍.

表 8 区块链基准负载对比

测试基准	负载类型		负载特征	
	数据更改	数据查询	静态	动态
BlockBench[39]	✓	✓	✓	✓
BlockMeter[48]	✓	–	✓	–
Klenik等人[50]	✓	–	✓	–
Touloupou等人[99]	✓	–	✓	–
BBSF[88]	✓	–	✓	✓
Diablo[42]	✓	–	✓	✓
DAGBENCH[46]	✓	✓	✓	–
Gromit[43]	✓	–	✓	–
BCTMark[45]	✓	–	✓	–
BCadvisor[44]	✓	–	✓	–
xBCBench[47]	✓	–	✓	–

5.3.1 数据更改负载

(1) 移植数据库负载

BlockBench[39]中通过智能合约形式将两个数据库的基准 YCSB 和 Small Bank 移植到私有区块链中. xBCBench[47]中设计了一个智能合约, 将 Small Bank 工作负载移植到 Sawtooth 中, 其中包括交易储蓄、存款支票、发送付款、写支票和合并操作. 两者均未对智能合约的设计做详细描述. Klenik 等人[50]将 TPC-C 的负载移植到 Hyperledger 中, 开发了 TPC-C 的负载产生器并与 Hyperledger Caliper 进行集成, 实现负载调度和执行, 负载的产生速率可以通过参数进行控制. 上述基准都描述了负载的频率, 但对数据大小、并发数量、负载类型变化等介绍较少.

(2) 根据实际应用产生

大部分区块链测试基准根据某个应用场景来构造负载. BlockBench[39]使用了 3 个流行以太坊的智能合约 EtherId、Doubler 以及 WavesPresale 作为工作负载, 并通过修改键值的方式, 将这 3 种智能合约引入到 Hyperledger 中. 负载的分布和发送速率通过参数设置.

Diablo[42]中设计了实现 5 个去中心化应用程序的智能合约, 每个合约中定义了一些方法函数来实现相应的业务逻辑, 并将其作为工作负载. 交易所 DApp 通过 ExchangeContractGafam 智能合约实现, 包含 CheckStock、BuyGoogle、BuyApple、BuyFacebook、BuyAmazon、BuyMicrosoft 等函数. 视频分享 DApp 通过 Decentralized-YouTube 智能合约实现, 其中包含一个上传函数, 该函数将一些视频数据作为参数. 游戏 DApp 通过 DecentralizedDota 智能合约实现, 其中, Update 函数实现了 10 个玩家在 250×250 地图的 x 轴和 y 轴上移动. FIFA Web 服务 DApp 通过一个简单的 Counter 智能合约实现, 其中包含一个 add 函数, 该函数会随着 FIFA 网站请求数量的增加而递增. 移动服务 DApp 通过 ContractUber 智能合约实现, 其中包含 CheckDistance 函数, 用于计算顾客 (请求者) 与一个区域 (一个二维网格) 中的 10000 名司机之间的距离, 以便将最近的司机与顾客匹配. Diablo[42]中每一种工作负载的大小和随时间的分布与真实业务场景相符合. 负载的组合类型、比例和时间分布在基准配置文件中定义.

BBSF[100]中定义了去中心化代币交易所、NFT 市场、NFT 铸造、体育博彩网站这 4 个工作负载. 这 4 个工作负载的智能合约按照 Web2 等价物的历史需求, 裁剪冗余功能, 仅保留完成任务所需的核心功能实现. 去中心化代币交易所: 一个基于 Uniswap-V2 的去中心化代币交易所, 用户可以以小费用交换不同类型的代币, 并通过提供流动性来获得奖励. 工作负载包括提供流动性、取回流动性和交换代币的交易. 工作负载的大小、交易组合和到达

分布是根据 Uniswap-V2 的历史数据生成的. NFT 市场: 一个 NFT 市场的交易量. 工作负载包括列出交易、发布 NFT、销售交易以及资金和 NFT 的转移. 工作负载的大小、交易组合和到达分布基于 OpenSea 的历史数据. NFT 铸造: 通过生成独特的随机数确定 NFT 特征, 使用 IPFS 将艺术作品与代币绑定, 并将代币交给用户完成交易. 这个工作负载的交易在时刻 0 就完成了, 不需要到达分布, 因为典型的 NFT 项目在初始部署时完成大部分或全部的铸造. 工作负载的大小基于常见的 NFT 项目规模. 体育博彩网站: 工作负载涉及计算赢家并支付奖金, 所有交易都从时刻 0 开始, 区块链以最大吞吐量运行. 工作负载的大小是根据当前 Web2 体育博彩网站 (如 Fanduel) 上不同规模比赛的数据确定. BBSF[88]中使用历史数据来确定交易组合和到达分布, 交易组合描述了负载中交易类型及其比例, 到达分布描述了负载的特征, 包括负载类型和频率随着时间的变化.

DAGBENCH[46]中转账操作作为事务处理的工作负载. 转账操作被定义为将货币或任意数据从一个用户转移到另一个用户. Gromit[43]使用固定的资产转移作为工作负载, 工作负载由至多 64 个客户端发出, 均匀地分配给每一个验证节点. 但是文献中没有详细介绍工作负载的智能合约实现. xBCBench[47]根据不同的区块链系统开发了不同的工作负载. 对于以太坊, 开发了 3 种工作负载: 账户创建、账户查询和账户之间的转账. 对于 Sawtooth, 开发了查询和 Small Bank 工作负载. 对于 FiscoBcos, 开发了设置 (set) 和获取 (get) 两种工作负载, 其中设置负载负责生成一个 HelloWorld 智能合约并部署该智能合约, 获取负载负责调用智能合约并输出 Hello World. xBCBench 中所有负载的发送速率通过基准测试层进行配置. BlockMeter[48]中选取了 3 个典型的区块链应用作为负载. 一是简单的键值存储应用程序: 该应用程序在区块链中根据唯一 ID 存储特定的值, 并执行插入、更新和删除数据等操作. 二是数据密集型应用程序: 该应用程序在每个事务中处理大量数据, 对大小为 10 KB 的记录执行读写操作. 三是 CPU 密集型应用程序: 该应用程序包括对随机数进行重复算术运算操作.

(3) 负载产生器

少量区块链测试基准提供了一个负载产生器, 可以自由定义负载的类型和比例, 但并不绑定具体应用场景. Touloupou 等人[99]提出的测试基准中开发了两个组件, 负责生成账户并在网络中执行交易. 用户可以通过参数, 调整要执行的交易数量和要交换的货币金额. BCTMark[45]中提供了两种方式产生负载, 一种是使用 Python 编写的负载生成器 Locust 产生随机负载, 另一种是基于区块链主网的历史交易数据生成负载, 重放交易过程. BCadvisor[44]开发了一个工作负载引擎, 为区块链生成预定义的工作负载量.

5.3.2 数据查询负载

目前区块链基准中只有 BlockBench[39]和 DAGBENCH[46]定义了数据查询负载. BlockBench[39]中定义了一个查询负载用来测试区块链数据层的性能. 该负载包含以下两个查询, 实现从区块链数据获取统计信息.

- Q1: 计算在第 i 个块和第 j 个块之间提交的总交易金额.
- Q2: 计算在第 i 个块和第 j 个块之间涉及给定状态 (账户) 的最大交易金额.

DAGBENCH[46]定义了交易查询的负载用来测试 DAG 网络对历史数据检索的性能. 它包含两个查询: 查询 1 (Q1): 计算给定账户的输入交易数量和输出交易数量. 查询 2 (Q2): 计算给定账户的余额.

5.3.3 负载总结

目前的测试基准大多数都包含了对工作负载的说明, 但是仍然存在一些问题.

- 对于实现工作负载的智能合约描述的都比较简单, 缺少了对合约具体实现方法的设计和描述.
- 大多数文献缺少了对负载动态特性的考虑, 只有 BlockBench[39]、BBSF[100]和 Diablo[42]关注了负载的动态特性, 并将其设置为与真实业务场景相匹配.
- 数据查询负载比较少, 只有 BlockBench[39]和 DAGBENCH[46]中定义了查询负载, 其他测试基准均没有考虑数据查询负载.

5.4 指 标

5.4.1 指标定义

根据业务场景不同, 区块链测试基准中的指标定义也不同, 现有测试基准的指标总结如表 9 所示. 绝大部分区

块链测试基准都包含了性能指标, 超过一半的基准包含了资源效率指标, 只有少量基准包含扩展性和安全性指标. 下面逐一介绍.

表 9 区块链基准指标对比

基准	性能	资源效率	扩展性	安全性
BlockBench[39]	✓	—	✓	✓
Hyperledger Caliper[40]	✓	✓	—	—
C2B2[98]	✓	✓	—	—
Klenik等人[50]	✓	—	—	—
Touloupou等人[99]	✓	✓	—	—
BlockMeter[48]	✓	✓	—	—
BBSF[88]	✓	—	✓	✓
Diablo[42]	✓	—	—	—
DAGBENCH[46]	✓	✓	✓	—
Gromit[43]	✓	✓	—	—
BCTMark[45]	—	✓	—	—
BCadvisor[44,87]	✓	✓	—	✓
xBCBench[47]	✓	✓	—	—

(1) 性能指标. 几乎所有基准都包含性能指标, 只有 BCTMark[45]中缺少了对性能指标的详细描述, 仅描述了对资源消耗指标的收集. xBCBench[47]中除了定义了事务的吞吐率和延迟以外, 还定义了读取延迟和读取吞吐率两个指标, 用来衡量读取操作的性能, 这两个指标也可以通过第 3 节定义的公式 (1) 和公式 (3) 计算.

(2) 资源效率指标. 超过一半的测试基准包含了资源效率指标, 其中, BCadvisor[44]是一个面向资源消耗的区块链测试基准, 它定义了一个名为 TsRoB 的流程, 用于对区块链每层的资源效率进行评估. 对于共识层和网络层, BCadvisor 定义了功率、CPU、内存、硬盘和网络的资源效率公式, 还考虑了客户端 RPC 请求的消耗, 通过引入权重因子, 将 RPC 请求和区块链交易两类任务结合在一起. BCTMark 在指标设计上考虑了公有链和许可链的差异, 其评测了以太坊的功耗情况, 但未对 Hyperledger Fabric 进行相同的实验.

(3) 扩展性. 目前只有 BlockBench[39]、BBSF[88]和 DAGBENCH[46]中对区块链的扩展性指标进行了定义. 其中, BBSF 和 DAGBENCH 通过在不同节点数量下运行测试基准并进行曲线拟合来推断区块链在扩展过程中的表现. BlockBench 除了观察节点的变化外, 还观察不同并发工作负载数量下吞吐量和延迟的变化来推断区块链的扩展性能.

(4) 安全性. 在当前的基准中, BlockBench[39]、BBSF[88]都定义了安全性指标. BlockBench 中将安全性指标量化为主分支中包含的总区块数与用户确认的总区块数的比值, 比值越低, 系统抵抗双重支付和自私挖矿的安全性越高. BBSF 中通过模拟故障节点和恶意节点的行为, 测量它们对系统性能的影响, 安全性指标由不同数量的故障节点下系统的性能比率组成, 该指标揭示了区块链在面对不同攻击时的性能受影响程度. BCadvisor[44,87]考虑了 DDoS 攻击和 Sybil 攻击, 设计了一组防御性能 (DP) 的指标, 使用攻击时吞吐率与系统的最大吞吐率的比值来衡量区块链受到攻击时服务的可用性.

5.4.2 指标计算

目前大多数区块链测试基准使用区块查询的方法来获取区块链的性能数据. 然而, 在实时监控的情况下, 这种数据获取方式会给验证节点本身带来了很大的开销, 影响测试结果的准确性. 此外, 区块查询方法不能获取区块链系统全部指标, 存在一定局限性.

(1) 区块查询. 在当前的测试基准中, 大多数性能指标的获取是通过与区块链进行交互, 收集数据, 然后进行深入的分析和计算来实现的. 不同的测试基准中实现方式有些差异. 例如, 有些测试基准, 如 Diablo、BlockBench、

Gromit、BBSF 等, 是通过利用 RPC 接口与区块链进行交互, 从而获取区块中的交易数量和提交时间等关键信息, 进一步通过计算来得出性能指标数据. 另一方面, 有些测试基准, 如 BCTMark 等, 是通过使用外部插件工具来收集区块链的性能指标数据. 还有一些测试基准借助 Prometheus 和 Grafana 等开源工具对指标数据进行存储和可视化. 下面对常见的区块链基准中指标的获取关键过程进行描述.

Diablo[42]将发送事务前记录提交时间, 通过轮询区块链节点, 检查最新区块是否包含已发送的事务, 当检测到时, 记录为事务的确认时间. 测试完成后通过记录的数据来计算指标. Gromit[43]通过两种方式获取事务的确认时间, 一是查询区块中时间戳, 二是定期轮询区块链系统, 以确定事务是否已经确认. BlockBench[39]中将事务提交后返回的交易 ID 存储到本地队列中, 利用 GetLatestBlock(h) 方法获取区块链上完成的交易列表, 并与本地队列做比较, 删掉匹配的交易. 通过该方法可以获取一定时间完成的交易总数. BBSF 使用 RPC 与区块链交互, 并监视区块获取完成的交易数量, 根据交易数量、当前区块的时间戳以及第 1 个交易的调用时间, 计算吞吐量和延迟.

Touloupou 等人[99]的研究中使用 XRPL 中提供的 rippled 进程来捕获性能数据, 使用 Prometheus 来获取资源利用率数据, 并将这些指标数据存储到 InfluxDB 中.

BCadvisor[44]中的数据解析器模块可通过 RPC 与区块链模块进行交互, 获取资源利用情况和最新的账本状态, 获取所需数据后, 根据公式计算各个指标的值. 计算出来的结果使用 Prometheus 和 Grafana 进行存储和可视化.

BlockMeter[48]中通过一个性能监控器收集区块链测试过程中的数据, 包括事务的开始时间、结束时间以及容器的资源状态数据等. 该性能监控器使用 NodeJS 和 Express 搭建, 通过 API 网关与底层区块链进行交互, 获取性能和资源状态数据.

Caliper 和 xBCBench 的核心层通过实现不同功能的北向接口获取资源状态数据和交易过程数据. 其中资源监控模块可以获取区块链系统的资源效率数据, 包括 CPU 利用率、内存利用率、网络使用等. 性能分析器模块在执行交易的过程中记录关键指标数据, 例如交易创建时间、交易提交时间、交易返回结果等. 这些数据用于生成最终的指标数据, 并汇总成测试报告.

BCTMark 使用 Telegraf 的 HTTP 插件通过以太坊的 HTTPAPI 收集区块链节点的指标数据和服务器 (CPU、内存消耗、硬盘使用等) 的数据, 并使用 InfluxDB 和 Grafana 进行存储和展示.

Klenik 等人[50]的研究中使用了开源的 Elasticstack, 处理来自 Fabric 节点和 Caliper 工作服务的日志.

(2) 日记收集. 通过区块查询获取指标数据的方式会对区块链节点产生一定的性能开销, 这可能会影响性能指标结果的准确性. 文献 [51] 提出了一个基于日志的区块链分析工具. 该工具为每个区块链验证节点配置了一个分析器, 通过监控区块链节点的运行, 解析日记, 提取统一的动作, 并将其记录在固定格式的元组中. 该工具能够通过日记分析获取区块链系统不同阶段的详细信息, 并基于解析节点运行日记中的统计数据评估实时性能. 与基于 RPC 的区块查询的方法相比, 该方法具有更低的开销, 更加适合实时监控. 然而, 由于不同的区块链遵循不同的日志格式, 这个方法无法扩展到所有的区块链.

5.4.3 指标总结

目前的测试基准都包含指标的定义, 但是对于指标的选择、指标获取等方面存在一些差异.

- 大多数基准中考虑了性能指标和资源效率指标, 对于安全性和可扩展性指标考虑的比较少, 只有 BBSF 和 BlockBench 定义了安全指标.
- 目前的基准中对指标的定义公式各不相同, 有基准中甚至缺少了指标的定义公式, 例如 BlockMeter、C2B2 和 Gromit 等.
- 测试基准中对于指标的获取方式也不统一, 目前绝大多数采用区块查询的方式获取指标, 这其中有的基准采用自研的性能分析函数, 有的借助一些外部插件工具, 不同的方式对于结果的影响不同.
- 大多数的基准中对指标的获取过程进行了描述, 但是有些基准中没有明确指标的获取方式, 例如 DAGBENCH.

5.5 执行规则

区块链执行规则分为测试配置和测试流程, 其中测试配置进一步分为系统配置和攻击配置. 本节从系统配置、

攻击配置和测试流程这 3 方面对现有的基准进行对比分析. 如表 10 所示, 这里的系统配置我们只考虑了提供部署工具的基准. 总体来看, 现有基准对系统配置和测试流程定义较多, 但只有少量基准考虑攻击配置.

表 10 区块链基准执行规则对比

基准	系统配置	攻击配置	测试流程
BlockBench[39]	—	✓	—
Hyperledger Caliper[40]	✓	—	✓
C2B2[98]	✓	—	—
BlockMeter[48]	—	—	✓
Klenik等人[50]	—	—	✓
Touloupou等人[99]	✓	—	—
BBSF[88]	—	✓	✓
Diablo[42]	—	✓	—
DAGBENCH[46]	✓	—	✓
Gromit[43]	—	—	✓
BCTMark[45]	✓	—	✓
BCadvisor[44,87]	✓	✓	—
xBCBench[47]	✓	—	✓

5.5.1 系统配置

为了确保基准测试结果的独立性、公平性和可重复性, 测试基准需提供详细的软硬件配置和参数设置, 包括硬件配置、软件配置、共识协议、网络配置、节点数量等. 所有的测试基准中实验实施部分都对测试环境进行了描述, 但部分基准没有详细介绍环境配置, 难以实现测试结果的可重复性以及保证结果的公平性.

由于区块链的配置过程比较复杂[101], 为了保证实验的可重复性, 有些基准提供了自动化的工具简化了系统配置过程. 下面进行详细的分析.

BCTMark[45]基于两个开源工具 EnosLib 和 Ansible 简化了区块链的部署过程. 用户可通过使用 YAML 配置文件描述所需的区块链部署配置, BCTMark 利用 YAML 配置文件、Ansible 的 Playbook 和 SSH 连接实现了任意物理平台上自动化的部署, 从而实现了部署的可重复性和可扩展性.

C2B2 在 BAF (blockchain automation framework) 的基础上改进了配置的过程, 通过高级配置描述自动生成 BAF 的配置文件, 简化了 Fabric 在云上的部署过程. BAF 是 Hyperledger 项目的自动化部署脚本集合, 用于在云上部署区块链系, BAF 结合了 Ansible 和 Helm 图表的功能, 前者具有配置部署功能, 而后者描述区块链组件在 Kubernetes 集群上的部署规则. C2B2 的高级配置文件中包括了部署的联盟组织的数量和性质, 每个组织的节点数量, 是否具有排序节点等内容, 这为用户提供了灵活且简化的配置方式, 提高了基准测试的可重复性.

Touloupou 等人[99]提出了一个区块链测试基准框架的初始化设置, 指定区块链协议和节点数量后, 可以通过一行代码完成区块链网络的部署.

xBCBench 提供了一个图形化的界面接口, 通过输入相关配置参数, 完成区块链网络的搭建和智能合约的部署.

Caliper 采用 Docker 容器来运行区块链网络. 网络配置文件中详细描述了区块链系统的拓扑结构、节点位置以及启动脚本等信息. 管理进程根据这个配置文件, 配置部署相应的区块链网络. Caliper 支持多个区块链平台, 如 Hyperledger Fabric、Sawtooth、Iroha 等.

有些测试基准提供了统一的适配接口, 但是区块链系统可能需要使用各自的脚本进行适配. 有些测试基准系统提到了部署管理, 但没有详细描述其具体内容和过程.

BlockBench 中各区块链需要实现 IBlockchainConnector 接口来集成到基准框架中, 该接口包含了部署应用程序、通过发送交易调用应用程序以及查询区块链状态的操作. Diablo 中为了兼容不同的区块链系统, 在配置文件

中将区块链抽象为 (E, R, I) 的元组, 其中 E 代表区块链的节点集合, R 代表了资源状态的结合, 例如账户余额、智能合约状态等, I 代表了交易类型, 例如转账、合约调用等. 新的区块链加入需要实现至少 1 种交易类型和 4 个具体的区块链执行函数. 这些函数包括创建客户端、创建资源、编码交互和触发交互发送到区块链. 上述两个基准中虽然提供了统一的适配接口, 但是各区块链系统需要通过自己的部署脚本进行部署适配.

BCadvisor[44]是一个面向资源的区块链测试基准, 它采用了模块化架构, 包括 3 个协作模块: 区块链模块、数据解析器和指标可视化器. 其中, 区块链模块采用 Docker 技术进行部署管理, 保证了新区块链协议的兼容性. 研究中没有针对 Docker 部署的方式进行详细的描述. DAGBENCH[46]中配置文件包含网络、负载、环境的配置, 通过读取配置文件, 可以完成 DAG 实例的初始化. 但是研究中缺少针对部署管理的详细描述.

5.5.2 攻击配置

区块链系统的安全性是其成功应用和广泛采用的关键因素之一, 也是其区别于数据库技术的显著特征, 因此攻击配置是区块链基准的要素之一. 根据第 4 节的分析, 可以通过模拟节点故障、网络分区和恶意攻击等常见故障及攻击来测试系统的安全性. 目前的基准中仅有 BlockBench、BBSF、Diablo 和 BCadvisor 考虑了攻击配置, 但是前三者均没有考虑恶意攻击的情况. C2B2 和 BCTMark 提供了网络仿真的功能, 能够通过配置文件的描述实现对各种网络故障的模拟, 但是二者均缺少了实验分析.

BlockBench[39]在压测过程中通过关闭服务器来模拟节点故障, 评估了各系统应对系统崩溃故障的能力. 通过模拟导致网络分区的攻击, 评估了区块链系统抵御双重支付或自私挖矿攻击的能力. Diablo[42]通过注入高工作负载, 测试区块链的性能, 以此来评估区块链系统抵御拒绝服务攻击的能力. BBSF[88]在一个 32 节点的区块链上运行工作负载, 通过分别设置 0、4 和 8 个节点返回空值和随机值来模拟崩溃故障和恶意攻击, 测量和分析不同故障情况下的性能变化, 从而评估系统在面对安全攻击时的性能影响程度.

BCTMark[45]中提供了网络仿真功能, 通过配置文件的描述, 可模拟数据包丢失率、网络延迟和网络速率 (带宽) 等网络特性, 这个功能可用于研究网络故障、延迟对区块链性能的影响. 但是研究中没有针对网络故障进行测试. C2B2 集成了 Kubernetes 的云原生混沌工程库 ChaosMesh, 并扩展了其网络仿真能力, 支持多个目标网络配置. 用户可以使用简单的高级描述文件描述目标网络环境和网络故障 (例如网络性能损失、节点崩溃、分区等). 该研究中利用该功能评估了地理分布对性能的影响, 但缺少了对网络故障等条件下区块链系统性能的分析.

文献 [47] 中给出了故障负载 (Faultload) 的定义, 是指当出现磁盘故障、网络超载或故意攻击等情况时, 区块链系统应能够保证账本的不可变性, 并在不崩溃的情况下正常运行. 但是该文献在实验设计中没有对区块链进行攻击配置.

BCadvisor[44,87]模拟了两种常见的攻击: DDoS 攻击和 Sybil 攻击. 评估了这两类攻击对系统性能的影响情况.

5.5.3 测试流程

在现有的区块链测试基准中, 超过一半的基准都对测试流程进行了详细的描述. 虽然各个基准的工具实现方式不同, 其基准测试的流程也存在差异, 但大多数都包括了环境准备、测试执行和测试报告这 3 个部分. 以下进行详细对比分析.

DAGBENCH 定义了基准测试的步骤, 包括初始化区块链环境、部署负载、执行负载、产生报告和环境清理等. BCTMark[45]定义基准测试的基本流程包括申请资源、部署测试环境、执行测试、备份结果和环境清理等.

xBCBench 的工作流程包括 3 个阶段: 准备阶段、测试阶段和报告阶段, 其中准备阶段包括环境部署、合约部署和监控组件的启动. 测试阶段是按照预定的测试计划执行负载, 并收集进度信息. 报告阶段的主要工作是收集资源效率数据和性能数据, 并汇集成可视化报告.

BlockMeter[48]的工作流程主要包括用户创建、负载创建、流量/负载生成、事务请求、请求预处理、性能监控和事务响应等环节. 在测试过程中, 它利用 JMeter 生成 HTTP 请求, 并将其发送至 API 网关. API 网关在接收到 HTTP 请求后, 会创建实例并初始化性能矩阵. 请求随后被转发至事务处理程序, 进行预处理并发送至区块链平台. 当事务准备提交至区块链时, 性能监控器开始记录性能矩阵以及 Docker 容器/实例的使用情况. 一旦区块链完成事务处理, 它会向框架发送响应, 此时性能矩阵被最终确定, 并向 JMeter 返回成功的状态码. 性能数据, 包括事务

延迟、事务吞吐量和资源消耗, 会被以 JSON 格式记录在文件系统中, 并利用 NodeJS 和 Python 库进行标准化和可视化分析

BBSF 中定义了一个标准的驱动程序, 以标准化的方式发起负载, 测量指标. 该驱动程序采用服务器/客户端的模式, 服务器负载编译和部署合约, 客户端充当连接和使用区块链的用户. 工作流程为服务器将事务发送至客户端, 客户端调用事务, 并记录时间戳. 服务器监控区块链, 并测试吞吐量和延迟.

Gromit 的工作流程分为实验设置、场景文件解析、实验执行和收集结果这 4 部分. 在实验设置部分, Gromit 会生成一个协调器进程, 该进程读取配置文件, 并进行区块链系统部署和环境的初始设置. 场景文件描述了基准的负载操作, 由场景解析器进行解析. 实验执行部分, Gromit 模拟客户端发出事务的过程, 通过 API 与区块链节点交互. 另外 Gromit 使用 Procfs 库监控 CPU、内存、磁盘和网络等资源的利用率. 在收集实验结果阶段, 协调器收集区块链产生的数据进行分析和存储.

Caliper 的核心组件设定了系统的工作流程, 主要包括 5 个步骤. 首先, 根据网络配置文件部署区块链网络. 其次, 初始化区块链系统, 如创建用户、加入通道等. 接着, 部署用于基准测试的智能合约. 然后, 根据基准配置文件执行工作负载. 最后, 生成测试报告并进行清理工作.

5.5.4 执行规则总结

目前的现有测试基准都对执行规则的某些要素进行了描述, 但是在实现基准测试的可重复性方面, 现有的基准仍然存在一些问题.

- 实验的不可重复性: BlockBench、Diablo、BBSF 和 Gromit 等基准缺少了环境部署管理能力, 无法通过简单的脚本和配置文件实现环境的快速部署, 保证实验的可重复性.
- 实验的准确性: BCadvisor 使用 Docker 等模拟环境进行测试, 与与真实应用的配置存在差异, 可能导致测试结果的准确性不够高.
- 安全性测试: BlockBench、BBSF、Diablo 和 BCadvisor 进行了安全性评估, C2B2 和 BCTMark 提供了网络故障模拟的能力, 其他基准均没有涉及安全性测试.
- 测试流程: 目前测试基准的工作流程中缺少数据集导入步骤的描述, 这一步作为测试环境的预配置, 对于实验结果的复现极为重要.

5.6 小 结

综上所述, 当前的区块链测试基准已呈现百花齐放局面, 但还存在一些普遍性问题. 首先, 缺乏对应用模型和数据模型的设计, 难以准确地模拟真实应用场景和用户行为. 其次, 缺少面向真实应用的负载设计, 无法全面评估区块链系统在实际使用中的性能表现. 第三, 现有的测试基准更多地关注性能指标, 而缺乏对扩展性和安全性等重要指标的考虑. 第四, 尽管跨链、侧链和分片等技术在区块链领域中越来越重要, 但是现有的测试基准没有充分地考虑这些新兴技术. 最后, 目前还没有针对国内典型区块链应用场景如电子政务、数据要素流通或司法的测试基准, 这在一定程度上制约了国内区块链行业生态的发展.

6 总 结

测试基准的目的在于公正客观地评估不同系统的性能. 随着区块链技术的快速发展, 近年来也涌现出一大批区块链测试基准, 但其包含的内容差异较大, 所采用的指标也参差不齐.

本文对现有区块链测试基准进行了充分的调研, 参考目前广泛使用的数据库测试基准, 结合区块链系统自身特点, 建立了一个全面衡量区块链系统的指标体系, 涵盖了性能、资源效率、扩展性和安全性这 4 类指标. 提出了一个规范化的区块链测试基准参考框架 UFBCB, 该框架包括应用模型、数据模型、负载、指标和执行规则 5 个要素. 基于该框架, 对现有区块链测试基准进行了全面对比分析. 本文工作可对未来区块链测试基准的制订和实施提供参考, 有助于推动区块链行业的规范和有序发展.

释放数据要素潜能、大力发展新质生产力已成为国家战略. 区块链技术作为构建可信安全数据基础设施的主

要技术, 可以通过构建数据安全服务, 消除各类数据要素市场参与者能力不同、认知不同造成的差异, 解决数据要素流通过程中的信任问题和多主体协作问题, 助力加速数据流通、释潜和赋能. 然而, 目前区块链还缺乏一个统一的测试基准参考框架. 而这也是正是本文希望解决的问题.

未来我们将基于本文提出的测试基准参考框架, 深入数据要素流通场景, 制订适合国情的、公正客观的区块链测试基准. 此外, 区块链技术正在快速发展, 离链、侧链、分片、图链、跨链等技术不断涌现. 本文提出的测试基准参考框架主要针对主流的 PoX 和 PBFT 区块链系统, 未来我们将研究如何扩展基准框架, 支持新兴的区块链系统.

References:

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

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

[3] Conti M, Kumar ES, Lal C, Ruj S. A survey on security and privacy issues of Bitcoin. IEEE Communications Surveys & Tutorials, 2018, 20(4): 3416–3452. [doi: 10.1109/COMST.2018.2842460]

[4] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (SoK). In: Proc. of the 6th Int'l Conf. on Principles of Security and Trust. Uppsala: Springer, 2017. 164–186. [doi: 10.1007/978-3-662-54455-6_8]

[5] Ruan PC, Dinh TTA, Loghin D, Zhang MH, Chen G, Lin Q, Ooi BC. Blockchains vs. distributed databases: Dichotomy and fusion. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1504–1517. [doi: 10.1145/3448016.3452789]

[6] Cocco L, Pinna A, Marchesi M. Banking on blockchain: Costs savings thanks to the blockchain technology. Future Internet, 2017, 9(3): 25. [doi: 10.3390/fi9030025]

[7] Nguyen QK. Blockchain—A financial technology for future sustainable development. In: Proc. of the 3rd Int'l Conf. on Green Technology and Sustainable Development (GTSD). Kaohsiung: IEEE, 2016. 51–54. [doi: 10.1109/GTSD.2016.22]

[8] Perboli G, Musso S, Rosano M. Blockchain in logistics and supply chain: A lean approach for designing real-world use cases. IEEE Access, 2018, 6: 62018–62028. [doi: 10.1109/ACCESS.2018.2875782]

[9] Tijan E, Aksentijević S, Ivanić K, Jardas M. Blockchain technology implementation in logistics. Sustainability, 2019, 11(4): 1185. [doi: 10.3390/su11041185]

[10] Xu LD, He W, Li SC. Internet of Things in industries: A survey. IEEE Trans. on Industrial Informatics, 2014, 10(4): 2233–2243. [doi: 10.1109/TII.2014.2300753]

[11] Li ZT, Kang JW, Yu R, Ye DD, Deng QY, Zhang Y. Consortium blockchain for secure energy trading in industrial Internet of Things. IEEE Trans. on Industrial Informatics, 2018, 14(8): 3690–3700. [doi: 10.1109/tii.2017.2786307]

[12] Liu YQ, Wang K, Lin Y, Xu WY. LightChain: A lightweight blockchain system for industrial Internet of Things. IEEE Trans. on Industrial Informatics, 2019, 15(6): 3571–3581. [doi: 10.1109/TII.2019.2904049]

[13] Wu ML, Wang K, Cai XQ, Guo S, Guo MY, Rong CM. A comprehensive survey of blockchain: From theory to IoT applications and beyond. IEEE Internet of Things Journal, 2019, 6(5): 8114–8154. [doi: 10.1109/JIOT.2019.2922538]

[14] Hölbl M, Kompara M, Kamišalić A, Zlatolas LN. A systematic review of the use of blockchain in healthcare. Symmetry, 2018, 10(10): 470. [doi: 10.3390/sym10100470]

[15] 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]

[16] Aitzhan NZ, Svetinovic D. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Trans. on Dependable & Secure Computing, 2018, 15(5): 840–852. [doi: 10.1109/TDSC.2016.2616861]

[17] Huang X, Cheng R, Wu JX, Guo QY, Chen SJ, Liu CJ, Wang Z, Chen YL, Hao SF, Zhang J. A blockchain-based authentication scheme for energy trading in electric transportation. IEEE Trans. on Smart Grid, 2024, 15(6): 6048–6062. [doi: 10.1109/TSG.2024.3422238]

[18] Fan CX, Ghaemi S, Khazaei H, Musilek P. Performance evaluation of blockchain systems: A systematic survey. IEEE Access, 2020, 8: 126927–126950. [doi: 10.1109/ACCESS.2020.3006078]

[19] Vukolić M. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In: Open Problems in Network Security: IFIP WG 11.4 Int'l Workshop. Zurich: Springer, 2016. 112–125. [doi: 10.1007/978-3-319-39028-4_9]

[20] Ferdous MS, Chowdhury MJM, Hoque MA, Colman A. Blockchain consensus algorithms: A survey. arXiv:2001.07091, 2020.

[21] Vukolić M. Rethinking permissioned blockchains. In: Proc. of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts. Abu

Dhabi: ACM, 2017. 3–7. [doi: 10.1145/3055518.3055526]

[22] 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 Conf. on Financial Cryptography and Data Security. Barbados: Springer, 2016. 106–125. [doi: 10.1007/978-3-662-53357-4_8]

[23] Scalability of Bitcoin. 2023. https://en.bitcoin.it/wiki/Scalability

[24] Kalodner H, Goldfeder S, Chen XQ, Weinberg SM, Felten EW. Arbitrum: Scalable, private smart contracts. In: Proc. of the 27th USENIX Security Symp. Baltimore: USENIX Association, 2018. 1353–1370.

[25] Gaži P, Kiayias A, Zindros D. Proof-of-stake sidechains. In: Proc. of the 2019 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2019. 139–156. [doi: 10.1109/SP.2019.00040]

[26] Li F, Li ZR, Zhao 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]

[27] Dickerson T, Gazzillo P, Herlihy M, Koskinen E. Adding concurrency to smart contracts. In: Proc. of the 2017 ACM Symp. on Principles of Distributed Computing. Washington: ACM, 2017. 303–312. [doi: 10.1145/3087801.3087835]

[28] Hashim F, Shuaib K, Zaki N. Sharding for scalable blockchain networks. SN Computer Science, 2022, 4(1): 2. [doi: 10.1007/s42979-022-01435-z]

[29] 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]

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

[31] Lewenberg Y, Sompolinsky Y, Zohar A. Inclusive block chain protocols. In: Proc. of the 19th Int'l Conf. on Financial Cryptography and Data Security. San Juan: Springer, 2015. 528–547. [doi: 10.1007/978-3-662-47854-7_33]

[32] Rauchs M, Blandin A, Bear K, McKeon S. 2nd Global Enterprise Blockchain Benchmarking Study. Cambridge: University of Cambridge, 2019.

[33] Valenta M, Sandner P. Comparison of Ethereum, Hyperledger Fabric and Corda. Technical Report, Blockchain Center, Frankfurt School, 2017. https://www.smallake.kr/wp-content/uploads/2017/07/2017_Comparison-of-Ethereum-Hyperledger-Corda.pdf

[34] Leutenegger ST, Dias D. A modeling study of the TPC-C benchmark. ACM SIGMOD Record, 1993, 22(2): 22–31. [doi: 10.1145/170036.170042]

[35] Stets R, Barroso LA, Gharachorloo K, Verghese B. A detailed comparison of TPC-C versus TPC-B. In: Proc. of the 3rd Workshop on Computer Architecture Evaluation Using Commercial Workloads in Conjunction with HPCA. 2000. https://www.barroso.org/publications/caecw2k.pdf

[36] Chen SM, Ailamaki A, Athanassoulis M, Gibbons PB, Johnson R, Pandis I, Stoica R. TPC-E vs. TPC-C: Characterizing the new TPC-E benchmark via an I/O comparison study. ACM SIGMOD Record, 2011, 39(3): 5–10. [doi: 10.1145/1942776.1942778]

[37] Dreseler M, Boissier M, Rabl T, Uflacker M. Quantifying TPC-H choke points and their optimizations. Proc. of the VLDB Endowment, 2020, 13(8): 1206–1220. [doi: 10.14778/3389133.3389138]

[38] Poess M, Rabl T, Jacobsen HA. Analysis of TPC-DS: The first standard benchmark for SQL-based big data systems. In: Proc. of the 2017 Symp. on Cloud Computing. Santa Clara: ACM, 2017. 573–585. [doi: 10.1145/3127479.3128603]

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

[40] Hyperledger Caliper: A blockchain performance benchmark framework. 2023. https://github.com/hyperledger/caliper

[41] Cheng Y, Wei K, Zhang YH, Jiang CY, Pang WW, Zhang Q, Liu B, Zhang LF, Liu TT, Wu YQ. TrustedBench: An efficient and user-friendly distributed performance testing tool for blockchain system. In: Proc. of the 22nd IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom). Exeter: IEEE, 2023. 2141–2146. [doi: 10.1109/TrustCom60117.2023.00298]

[42] Gramoli V, Guerraoui R, Lebedev A, Natoli C, Voron G. Diablo: A benchmark suite for blockchains. In: Proc. of the 18th European Conf. on Computer Systems. 2023. 540–556. [doi: 10.1145/3552326.3567482]

[43] Nasrulin B, De Vos M, Ishmaev G, Pouwelse J. Gromit: Benchmarking the performance and scalability of blockchain systems. In: Proc. of the 2022 IEEE Int'l Conf. on Decentralized Applications and Infrastructures (DAPPS). Newark: IEEE, 2022. 56–63. [doi: 10.1109/DAPPS55202.2022.00015]

[44] Qian K, Liu YQ, Han YM, Wang K. BCadvisor: Enabling green blockchain systems through resource-oriented benchmarking. In: Proc. of the 2022 IEEE Int'l Conf. on Communications. Seoul: IEEE, 2022. 4031–4036. [doi: 10.1109/ICC45855.2022.9838249]

[45] Saingre D, Ledoux T, Menaud JM. BCTMark: A framework for benchmarking blockchain technologies. In: Proc. of the 17th IEEE/ACS

Int'l Conf. on Computer Systems and Applications (AICCSA). Antalya: IEEE, 2020. 1–8. [doi: 10.1109/AICCSA50499.2020.9316536]

[46] Dong ZL, Zheng EM, Choon Y, Zomaya AY. DAGBENCH: A performance evaluation framework for DAG distributed ledgers. In: Proc. of the 12th IEEE Int'l Conf. on Cloud Computing (CLOUD). Milan: IEEE, 2019. 264–271. [doi: 10.1109/CLOUD.2019.00053]

[47] Wang R, Ye KJ, Wang Y, Xu CZ. xBCBench: A benchmarking tool for analyzing the performance of blockchain systems. In: Proc. of the 3rd Int'l Conf. on Blockchain and Trustworthy Systems. Guangzhou: Springer, 2021. 101–114. [doi: 10.1007/978-981-16-7993-3_8]

[48] Alom I, Ferdous MS, Chowdhury MJM. BlockMeter: An application agnostic performance measurement framework for private blockchain platforms. arXiv:2202.05629, 2022.

[49] Ren ZJ, Xiang HT, Zhou ZH, Wang N, Jin HQ. AlphaBlock: An evaluation framework for blockchain consensus algorithms. In: Proc. of the 9th Int'l Workshop on Security in Blockchain and Cloud Computing. ACM, 2021. 17–22. [doi: 10.1145/3457977.3460297]

[50] Klenik A, Kocsis I. Porting a benchmark with a classic workload to blockchain: TPC-C on Hyperledger Fabric. In: Proc. of the 37th ACM/SIGAPP Symp. on Applied Computing. ACM, 2022. 290–298. [doi: 10.1145/3477314.3507006]

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

[52] Touloupou M, Themistocleous M, Iosif E, Christodoulou K. A systematic literature review toward a blockchain benchmarking framework. IEEE Access, 2022, 10: 70630–70644. [doi: 10.1109/ACCESS.2022.3188123]

[53] Chacko JA, Mayer R, Fekete A, Gramoli V, Jacobsen HA. How to benchmark permissioned blockchains. In: Proc. of the 15th TPC Technology Conf. on Performance Evaluation & Benchmarking (TPCTC 2023). 2023. https://gramoli.github.io/pubs/TPCTC2023-preprint.pdf

[54] Anderson JC, Lehnardt J, Slater N. CouchDB: The Definitive Guide: Time to Relax. Sebastopol: O'Reilly Media, 2010.

[55] Dwork C, Naor M. Pricing via processing or combatting junk mail. In: Proc. of the 12th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 1993. 139–147. [doi: 10.1007/3-540-48071-4_10]

[56] Lamport L. The part-time parliament. ACM Trans. on Computer Systems, 1998, 16(2): 133–169. [doi: 10.1145/279227.279229]

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

[58] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Trans. on Computer Systems (TOCS), 2002, 20(4): 398–461. [doi: 10.1145/571637.571640]

[59] Szabo N. Smart contracts: Building blocks for digital markets. EXTROPY: The Journal of Transhumanist Thought, 1996, 18(2): 28.

[60] Luu L, Chu DH, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 254–269. [doi: 10.1145/2976749.2978309]

[61] Cheng JR, Xie LY, Tang XY, Xiong NX, Liu BY. A survey of security threats and defense on blockchain. Multimedia Tools and Applications, 2021, 80(20): 30623–30652. [doi: 10.1007/s11042-020-09368-6]

[62] Saad M, Spaulding J, Njilla L, Kamhoua C, Shetty S, Nyang D, Mohaisen D. Exploring the attack surface of blockchain: A comprehensive survey. IEEE Communications Surveys & Tutorials, 2020, 22(3): 1977–2008. [doi: 10.1109/COMST.2020.2975999]

[63] Huang JJ, Han SM, You W, Shi WC, Liang B, Wu JZ, Wu YJ. Hunting vulnerable smart contracts via graph embedding based bytecode matching. IEEE Trans. on Information Forensics and Security, 2021, 16: 2144–2156. [doi: 10.1109/TIFS.2021.3050051]

[64] Zhuang Y, Liu ZG, Qian P, Liu Q, Wang X, He QM. Smart contract vulnerability detection using graph neural networks. In: Proc. of the 29th Int'l Joint Conf. on Artificial Intelligence. 2021. 3283–3290.

[65] Apostolaki M, Zohar A, Vanbever L. Hijacking Bitcoin: Routing attacks on cryptocurrencies. In: Proc. of the 2017 IEEE Symp. on Security and Privacy (SP). San Jose: IEEE, 2017. 375–392. [doi: 10.1109/SP.2017.29]

[66] Marcus Y, Heilman E, Goldberg S. Low-resource eclipse attacks on Ethereum's peer-to-peer network. Cryptology ePrint Archive, 2018/236, 2018.

[67] Saad M, Thai MT, Mohaisen A. POSTER: Deterring DDoS attacks on blockchain-based cryptocurrencies through mempool optimization. In: Proc. of the 2018 Asia Conf. on Computer and Communications Security. Incheon: ACM, 2018. 809–811. [doi: 10.1145/3196494.3201584]

[68] Li HN, Wang K, Miyazaki T, Xu CH, Guo S, Sun YF. Trust-enhanced content delivery in blockchain-based information-centric networking. IEEE Network, 2019, 33(5): 183–189. [doi: 10.1109/MNET.2019.1800299]

[69] Swathi P, Modi C, Patel D. Preventing sybil attack in blockchain using distributed behavior monitoring of miners. In: Proc. of the 10th Int'l Conf. on Computing, Communication and Networking Technologies (ICCCNT). Kanpur: IEEE, 2019. 1–6. [doi: 10.1109/ICCCNT45670.2019.8944507]

[70] Yu HF, Gibbons PB, Kaminsky M, Xiao F. SybilLimit: A near-optimal social network defense against sybil attacks. In: Proc. of the 2008 IEEE Symp. on Security and Privacy (SP 2008). Oakland: IEEE, 2008. 3–17. [doi: 10.1109/SP.2008.13]

[71] Eyal I, Sirer EG. Majority is not enough: Bitcoin mining is vulnerable. Communications of the ACM, 2018, 61(7): 95–102. [doi: 10.1145/3212998]

[72] Zheng J, Huang HW, Zheng ZB, Guo S. Adaptive double-spending attacks on pow-based blockchains. IEEE Trans. on Dependable and Secure Computing, 2024, 21(3): 1098–1110. [doi: 10.1109/TDSC.2023.3268668]

[73] Couto da Silva FJ, Damsgaard SB, Mousing Sorensen MA, Marty F, Altariqi B, Chatzigianni E, Madsen TK, Schwefel HP. Analysis of blockchain forking on an Ethereum network. In: Proc. of the 25th European Wireless Conf. Aarhus: Verband Deutscher Elektrotechniker, 2019. 1–6.

[74] Gervais A, Ritzdorf H, Karame GO, Capkun S. Tampering with the delivery of blocks and transactions in Bitcoin. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: ACM, 2015. 692–705. [doi: 10.1145/2810103.2813655]

[75] Kistowski JV, Arnold JA, Huppler K, Lange KD, Henning JL, Cao P. How to build a benchmark. In: Proc. of the 6th ACM/SPEC Int'l Conf. on Performance Engineering. Austin: ACM, 2015. 333–336. [doi: 10.1145/2668930.2688819]

[76] Huppler K. The art of building a good benchmark. In: Performance Evaluation and Benchmarking. Lyon: Springer, 2009. 18–30. [doi: 10.1007/978-3-642-10424-4_3]

[77] Gray J. Database and transaction processing performance handbook. In: Gray J, ed. The Benchmark Handbook. San Mateo: Morgan Kaufmann, 1993.

[78] Bitton D, DeWitt DJ, Turbyfil C. Benchmarking database systems: A systematic approach. Technical Report, #526, Madison: Computer Sciences Department, University of Wisconsin—Madison, 1983.

[79] DeWitt DJ. The Wisconsin benchmark: Past, present, and future. In: Gray J, ed. The Benchmark Handbook. San Mateo: Morgan Kaufmann, 1993.

[80] Turbyfill C, Orji C, Bitton D. As^{3}AP—An ANSI SQL standard scalable and portable benchmark for relational database systems. In: Benchmark Handbook for Database and Transaction System. 2nd ed., 1993.

[81] Difallah DE, Pavlo A, Curino C, Cudre-Mauroux P. OLTP-bench: An extensible testbed for benchmarking relational databases. Proc. of the VLDB Endowment, 2013, 7(4): 277–288. [doi: 10.14778/2732240.2732246]

[82] O'Neil PE. The Set Query Benchmark. In: Benchmark Handbook for Database and Transaction System. 2nd ed., 1993.

[83] Nambiar RO, Poess M. The making of TPC-DS. In: Proc. of the 32nd Int'l Conf. on Very Large Data Bases. Seoul: VLDB Endowment, 2006. 1049–1058.

[84] O'Neil P, O'Neil B, Chen XD. The Star schema benchmark. 2009. http://www.cs.umb.edu/~poneil/StarSchemaB.PDF

[85] Cole R, Funke F, Giakoumakis L, Guy W, Kemper A, Krompass S, Kuno H, Nambiar R, Neumann T, Poess M, Sattler KU, Seibold M, Simon E, Waas F. The mixed workload CH-benCHmark. In: Proc. of the 4th Int'l Workshop on Testing Database Systems. Athens: ACM, 2011. 8. [doi: 10.1145/1988842.1988850]

[86] Milkai E, Chronis Y, Gaffney KP, Guo ZH, Patel JM, Yu XY. How good is my HTAP system? In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 1810–1824. [doi: 10.1145/3514221.3526148]

[87] Qian K, Liu YQ, Shu CR, Sun YF, Wang K. Fine-grained benchmarking and targeted optimization: Enabling green IoT-oriented blockchain in the 6G era. IEEE Trans. on Green Communications and Networking, 2023, 7(2): 1036–1051. [doi: 10.1109/TGCN.2022.3185610]

[88] Ren K, Van Buskirk JFB, Ang ZY, Hou SZ, Cable NR, Monares M, Korth HF, Loghin D. BBSF: Blockchain benchmarking standardized framework. In: Proc. of the 1st Workshop on Verifiable Database Systems. Seattle: ACM, 2023. 10–18. [doi: 10.1145/3595647.3595649]

[89] Shalaby S, Abdellatif AA, Al-Ali A, Mohamed A, Erbad A, Guizani M. Performance evaluation of Hyperledger Fabric. In: Proc. of the 2020 IEEE Int'l Conf. on Informatics, IoT, and Enabling Technologies (ICIoT). Doha: IEEE, 2020. 608–613. [doi: 10.1109/ICIoT48696.2020.9089614]

[90] Baliga A, Subhod I, Kamat P, Chatterjee S. Performance evaluation of the Quorum blockchain platform. arXiv:1809.03421, 2018.

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

[92] Nasir Q, Qasse IA, Talib MA, Nassif AB. Performance analysis of Hyperledger Fabric platforms. Security and Communication Networks, 2018, 2018: 3976093. [doi: 10.1155/2018/3976093]

[93] Pongnumkul S, Siripanpornchana C, Thajchayapong S. Performance analysis of private blockchain platforms in varying workloads. In: Proc. of the 26th Int'l Conf. on Computer Communication and Networks (ICCCN). Vancouver: IEEE, 2017. 1–6. [doi: 10.1109/ICCCN.

2017.8038517]
[94] Rouhani S, Deters R. Performance analysis of Ethereum transactions in private blockchain. In: Proc. of the 8th IEEE Int'l Conf. on Software Engineering and Service Science (ICSESS). Beijing: IEEE, 2017. 70–74. [doi: 10.1109/ICSESS.2017.8342866]
[95] Zhang JS, Gao JB, Wu ZH, Yan WT, Wo Q, Li QS, Chen Z. Performance analysis of the libra blockchain: An experimental study. In: Proc. of the 2nd Int'l Conf. on Hot Information-centric Networking (HotICN). Chongqing: IEEE, 2019. 77–83. [doi: 10.1109/HotICN48464.2019.9063213]
[96] Wang XY, Al-Mamun A, Yan F, Zhao DF. Toward accurate and efficient emulation of public blockchains in the cloud. In: Proc. of the 12th Int'l Conf. on Cloud Computing. San Diego: Springer, 2019. 67–82. [doi: 10.1007/978-3-030-23502-4_6]
[97] Stoykov L, Zhang KW, Jacobsen HA. Demo: VIBES: Fast blockchain simulations for large-scale peer-to-peer networks. In: Proc. of the 18th ACM/IFIP/USENIX Middleware Conf.: Posters and Demos. Las Vegas: ACM, 2017. 19–20. [doi: 10.1145/3155016.3155020]
[98] Kassab A, Rivière E, Rosinosky G, Sadre R, Tran VT. C2B2: A cloud-native chaos benchmarking suite for the Hyperledger Fabric blockchain. In: Proc. of the 18th European Dependable Computing Conf. (EDCC). Zaragoza: IEEE, 2022. 89–96. [doi: 10.1109/EDCC57035.2022.00024]
[99] Touloupou M, Christodoulou K, Inglezakis A, Iosif E, Themistocleous M. Benchmarking blockchains: The case of XRP ledger and beyond. In: Proc. of the 55th Hawaii Int'l Conf. on System Sciences. Maui: IEEE, 2022. 6005–6012.
[100] Van Buskirk J. The standardization of blockchain benchmarking [MS. Thesis]. Bethlehem: Lehigh University, 2023.
[101] Tran NK, Babar MA, Walters A. A framework for automating deployment and evaluation of blockchain networks. Journal of Network and Computer Applications, 2022, 206: 103460. [doi: 10.1016/j.jnca.2022.103460]

附中文参考文献:

[1] 邵奇峰, 张召, 朱燕超, 周傲英. 企业级区块链技术综述. 软件学报, 2019, 30(9): 2571–2592. http://www.jos.org.cn/1000-9825/5775.htm [doi: 10.13328/j.cnki.jos.005775]
[26] 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]

张孝(1972－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为数据库体系结构, 大数据管理与分析, 数据基准测评, 区块链.

刘昊(2000－), 男, 硕士生, 主要研究领域为区块链.

秦春玲(1986－), 女, 博士生, 主要研究领域为数据库, 区块链.

陈晋川(1978－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为分布式数据管理, 区块链.

王文收(1986－), 男, 工程师, 主要研究领域为软件工程, 数据库, 财务信息化.

杜小勇(1963－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为数据库系统, 大数据管理与分析, 智能信息检索, 知识工程.