如何优化我的区块链？一种多层级推荐方法

Jeeta Ann Chacko
chacko@in.tum.de 慕尼黑工业大学

Ruben Mayer
mayerr@in.tum.de 慕尼黑工业大学

Hans-Arno Jacobsen
jacobsen@eecg.toronto.edu
多伦多大学

arXiv:2301.04719v1 [cs.DC] 11 Jan 2023

摘要

除了提出新的区块链架构之外，文献中的现有区块链优化主要聚焦于当前区块链系统或面向数据的优化。然而，由于区块链涉及组织治理到智能合约设计等多个方面，因此需要一种涵盖特定区块链系统所有不同层面的整体方法，以确保将所有优化机会纳入考量。为此，我们定义了一种多层级优化建议方法，用于识别区块链在系统、数据和用户级别上的优化机会。我们从区块链日志中提取多种指标和属性，并正式定义了九项优化建议。基于这些概念，我们实现了自动化优化建议工具 BlockOptR。我们使用涵盖多个真实场景的广泛工作负载，对该系统进行了全面评估。实施所建议的优化后，我们观察到，交易成功率平均提高了 20%，延迟平均降低了 40%。

ACM 参考格式：
Jeeta Ann Chacko、Ruben Mayer 和 Hans-Arno Jacobsen。2023年。如何优化我的区块链？一种多层级推荐方法。载于 *ACM Conference* 论文集（*Conference'23*）。ACM，纽约州纽约市，15 页。
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 引言

区块链刚问世时，仅支持简单的加密货币兑换交易 [50]。然而，随着时间推移，区块链逐渐支持利用智能合约执行复杂交易，由此进入去中心化事务管理系统的领域，例如分布式数据库 [64]。由于区块链旨在无信任环境中实现共识，因此难以达到数据库的性能 [9, 16, 22, 26, 53, 59, 80]。不过，随着提供访问控制和交易执行策略的许可型区块链问世，区块链力求在提升性能的同时，仍至少提供部分去中心化信任 [3, 5, 28, 48]。

允许为个人或课堂用途免费制作本作品全部或部分内容的数字副本或纸质副本，前提是不得为营利或商业利益制作或分发副本，并且副本须附有本声明及首页上的完整引文。对于本作品中归他人而非 ACM 所有的组件，必须尊重其版权。注明出处后，允许摘录。若要以其他方式复制、重新出版、发布到服务器或重新分发至列表，则须事先获得明确许可和/或支付费用。请通过 permissions@acm.org 申请许可。*Conference'23*，*2023*年*6*月，美国华盛顿州西雅图 © 2023年 计算机协会。ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00 https://doi.org/10.1145/nnnnnnn.nnnnnnn

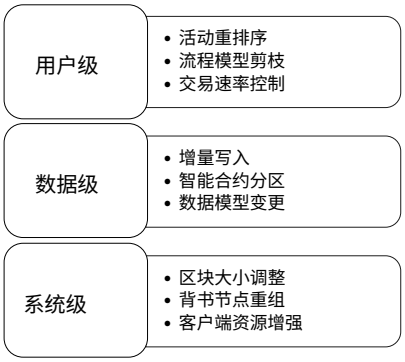

图 1：多层区块链优化

除了新型区块链系统设计层出不穷之外，针对现有许可型区块链中性能提升的系统优化领域，也有大量活跃且多元化的研究 [13, 27, 36, 37, 41, 54, 65 – 68]。相关文献涵盖的范围广泛，目标包括控制参数调优 [13, 41, 68]，、交易执行重构 [27, 37, 66]，以及智能合约优化 [54]。然而，我们注意到，针对特定区块链，将上述所有优化可能性纳入同一框架的综合方法仍然缺失。此外，现有文献尚未提出端到端优化方法，无法同时涵盖系统级调优、数据重构和流程模型重设计。由于许可型区块链主要由企业采用，通常会遵循模型驱动的方法：区块链网络的设置、交易规则、底层智能合约和数据模型，主要依据专为特定应用创建的业务流程模型来确定 [21, 40, 56, 63, 69]。这类流程模型可能由不了解性能影响的业务领域专家设计。例如，在 Hyperledger Fabric（又称 Fabric）[5]，中，许多交易失败都是由交易执行顺序导致的 [13, 65, 67]。如果发起交易的客户端流程从一开始就遵循不同的业务逻辑，这类失败便有望减少。数据库领域经常强调，在执行业务流程时，数据管理的重要性 [11, 20, 34]。我们也提出类似观点，认为区块链中的流程视角同样重要，因为区块链所涵盖的方面多种多样，并不局限于数据。

因此，鉴于特定区块链系统中可进行的优化种类繁多，这些优化在不同案例中的影响各异 [6, 13, 23, 51, 68, 81]，且实施工作量也有所不同，亟需一种推荐系统，根据具体用例，引导用户为所考虑的区块链选择有效的优化策略

。同样，我们可以借鉴数据库参数调优和索引建议方面的大量文献 [1, 2, 42, 73]。然而，区块链涉及组织治理 [62],、数据库定义 [59], 、共识算法 [46], 、溯源跟踪 [60],和智能合约设计 [47], 等多种因素，因此需要从整体视角提出优化建议，而目前这方面仍有欠缺。

为弥补这一不足，我们提出一种面向区块链的多层级优化建议方法，帮助用户全面了解其区块链系统的各种优化可能性，从而做出明智的决策。受数据库抽象层级的启发 [45], ，我们为区块链优化定义了三个抽象层级：系统级、数据级和用户级（参见图 1）。系统级建议包括确定理想的系统配置，例如区块大小或背书策略。数据级建议涉及理解数据模型并优化智能合约。用户级建议则侧重于业务流程模型，以及客户端流程所产生的工作负载。例如，在一个数字版权管理场景中（参见第 5.2 节），我们发现有两个活动经常导致交易冲突，并建议通过流程模型重设计来减少此类故障。我们的方法还可以验证是否符合新建的流程模型。我们设计并实现了一款名为 BlockOptR 的建议工具，用于分析 Fabric（企业广泛使用的区块链之一）的区块链日志 [61], ，以展示我们的方法带来的性能提升。

我们的贡献可总结如下：(1) 我们定义了一种多层级优化建议方法，对区块链日志进行全面分析，并从不同角度提出优化可能性。我们的方法有助于用户全面了解当前系统，并就优化策略做出明智决策。

(2) 我们基于通用属性对推荐策略进行了形式化定义，因此，任何具有相似属性的区块链日志都可以复用我们的方法。我们还探讨了如何将该方法应用于不同的区块链平台，从而为读者提供不受技术限制的视角。
(3) 我们实现了 Fabric 区块链数据的提取、预处理和事件日志生成技术的自动化。因此，我们的工具 BlockOptR 将有助于推动基于日志的分析领域的进一步研究，例如区块链中的流程挖掘，因为可以直接获取经过预处理的区块链日志。
(4) 我们通过实现并评估优化建议，展示了其有效性。实验表明，应用 BlockOptR 提出的建议后，成功交易的成功率平均提高了 20%，延迟平均降低了 40%。

（5）我们使用三种不同类型的工作负载对 BlockOptR 进行了广泛评估：一组通过多种控制变量生成的 24 个合成工作负载、文献中的四种常见基于用例的工作负载，以及一个贷款申请流程的真实世界事件日志。因此，我们的实验涵盖了广泛的场景，能够代表真实的区块链应用。这有助于克服公开可用数据匮乏的问题，而这限制了当前对许可型区块链流程挖掘的研究。BlockOptR 工具、所有智能合约、工作负载生成脚本以及所有事件日志均已作为开源资源发布，以推动该领域的进一步研究 [10]。

(6) 我们进一步证实，在现有区块链优化的基础上，整体推荐方法能够带来积极效果。因此，我们强调，BlockOptR 通过增加更高层级的优化，补充了现有的系统级区块链优化策略，例如 FabricSharp [65] 和 Fabric++ [67] 。

2 背景

2.1 Hyperledger Fabric

Fabic 是一种开源许可型区块链系统，深受企业欢迎 [5]。Fabric 的主要组件包括智能合约（称为链码）、分布式不可变账本、分布式世界状态数据库、一组分布式对等节点，以及排序服务。智能合约定义了区块链上所有受支持的事务。Fabric 的交易流程遵循执行-排序-验证（EOV）模型 [70]。Fabric 的 EOV 模型与数据库中的乐观并发控制相似 [31]，因此容易出现多版本并发控制（MVCC）冲突，进而导致交易失败。
(1) 在执行阶段，客户端创建交易提案并将其发送给背书节点。背书节点是一组特定的对等节点，有权执行智能合约并为事务背书。系统会配置背书策略，以规定事务被视为有效所需的背书节点数量。智能合约执行后，背书节点会生成读写集。背书节点对交易提案和读写集进行签名，然后将其发回客户端。

(2) 在排序阶段，客户端将这些交易转发给排序服务。排序服务使用 Raft [55]，（一种崩溃容错共识算法）对交易进行排序并打包成区块，然后将区块发送给网络中的所有对等节点。可配置参数会限制区块中所含交易的数量（区块大小），限制方式可以是交易数量（区块数量）、超时时间（区块超时），或交易的字节大小（区块字节数）。只要缓冲的传入交易集合满足这三个条件中的任意一个，就会创建区块。

(3) 在验证阶段，每个对等节点都会验证每笔交易。Fabric 网络中的每个对等节点都有一份分布式账本和世界状态的副本。对等节点会根据背书策略验证背书节点签名，并结合当前世界状态验证读写集。如果验证成功，就会更新世界状态；否则，就会检测到失败。如果背书验证失败，则称为背书策略失败；如果读写集验证失败，则称为 MVCC 读冲突。范围读取产生的 MVCC 读冲突称为幻读冲突。无论验证成功还是失败，所有交易都会追加到分布式账本中。此外，在相关文献中，MVCC 读冲突通常根据冲突交易是否位于区块链的同一区块中，分为区块间失败和区块内失败 [13, 67]。

2.2 事件日志与流程挖掘

事件日志是对流程随时间执行情况的记录。流程挖掘 [75]是一种推导流程模型的技术，该模型

能够体现事件日志中最常见的行为。它主要用于流程发现，以帮助理解底层流程模型；用于一致性检查，以识别预测的流程模型与流程实际行为之间的偏差；以及用于模型增强，以识别并消除瓶颈。进行流程挖掘所需的事件日志最少应包含以下数据：

(1) 案例标识：用于区分同一进程的不同执行。例如：供应链管理相关事件日志中的产品标识。进程的一次完整执行称为一条轨迹。

(2) 活动名称：用于标识进程中的不同步骤 .
例如：供应链管理相关事件日志中的 Ship 或卸载活动。

(3) 时间戳：用于确定不同活动的顺序。事件日志还可以包含其他属性，例如流程所有者、资源和成本。可以使用各种算法来推导流程模型，例如 Alpha Miner [76], 启发式挖掘器 [79] 和模糊挖掘器 [30]。这些算法的核心概念都是分析日志中活动集合的不同轨迹，并通过抽象或聚合简化轨迹，从而生成完整的流程模型。现有各种开源和商业流程挖掘工具（ProM [78], Disco [29], Celonis [12]）。

3 区块链的流程视角

我们的工作通过引入流程视角，将区块链优化视为一种整体方法，而非纯粹的系统级方法。本节通过示例强调，理解业务流程与区块链性能之间依赖关系的必要性和有效性。此外，这些示例也说明了优化推荐器的必要性，因为许多流程级优化只有在组织的决策机构批准后才能实施，而且在大多数情况下无法自动应用。

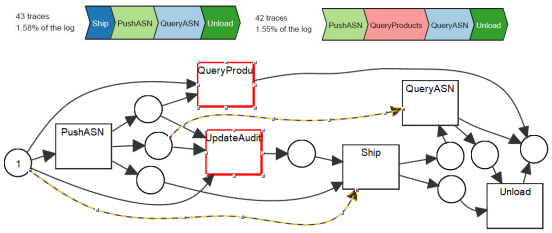

图2：SCM 场景的派生流程模型

流程模型剪枝是流程级优化的一个示例，能够对系统性能产生积极影响。图2展示了从供应链管理（SCM）场景的区块链日志中派生出的流程模型。图中标出的路径以及嵌入其中的轨迹揭示了流程模型中的两个冗余分支。除非推送预先发货通知（PushASN），否则绝不应执行发货活动。

同样，除非产品已发货，否则绝不应执行 Unload 活动。这类不合逻辑的活动路径可能由人为错误或交易失败导致；正如下例所述，智能合约的设计旨在处理此类问题。

如果执行 Unload 事务时没有对应的 Ship，事务只会读取状态，而不会对其进行修改。不过，执行时是让事务失败，还是将只读事务提交到区块链，取决于智能合约设计者。这两种设计各有利弊。提交事务会在区块链上添加不可篡改的记录，有助于追踪偏离预期流程模型的个人或组织。例如，在供应链管理场景中，这一点至关重要，因为整个流程都是分布式的，而区块链在此处的主要用途，是在不受信任的参与者之间提供数据溯源。另一方面，事务一经执行便立即失败，可以确保此类不必要的事务不会经历所有耗时阶段（排序和验证），从而提升系统性能。我们发现，在智能合约中剪除不必要的活动路径后，吞吐量提升了 27%，交易成功率提高了 19%（第6.2节，图 13）。还可以在流程执行层实施剪枝，通过对遵循或偏离预期流程模型的组织或个人施加激励或惩罚措施来实现。这种方法确保系统性能不会凌驾于数据溯源之上，因此兼具上述两种智能合约设计的优势。

未进行活动重排序

活动顺序	活动	读取数据，值	写入数据，值	有效性
1	PushASN	{ 产品标识, 1 }	{ **产品标识, 2** }	成功
2	UpdateAuditInfo	{ **产品标识, 1** } { 审计ID, 001 }	{ 审计ID, 002 }	**中止**

通过活动重排序

活动顺序	活动	读取数据，值	写入数据，值，有效性
1	UpdateAuditInfo	{ 产品标识, 1 } { 审计ID, 001 }	{ 审计ID, 002 } 成功
2	PushASN	{ 产品标识, 1 }	{ ProductID, 2 } 成功

图3：交易依赖冲突示例

失败的另一个原因是交易依赖关系，而有关串行化算法的研究已通过交易重排序有效减少了此类失败 [65, 67]。然而，重排序算法代价高昂，因为它们基本上需要解决生成无冲突依赖图这一 NP 难题 [67]。由于世界状态不一致而导致背书策略失败增加，以及无法处理大型范围查询，都是交易重排序已知的问题 [13]。解决依赖冲突问题的另一种方法，是识别可重排序和不可重排序的 [65] 活动，而不是交易。虽然现有文献会分析交易访问的键以理解可串行化性，但流程级串行化需要分析数据模型。如果两个并发活动读取同一个数据元素，却在数据模型中写入不同元素，那么这些活动就可以重排序。

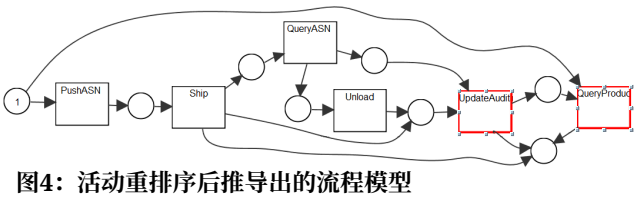

图4：活动重排序后推导出的流程模型

例如，在同一个供应链管理场景中，UpdateAuditInfo 活动读取产品ID并写入审计ID，而 PushASN、Ship 和 Unload 活动则读取并写入产品ID。因此，{UpdateAuditInfo, PushASN}、{UpdateAuditInfo, Ship} 和 {UpdateAuditInfo, Unload} 这些活动对可以重排序，而 {PushASN, Ship, Unload} 则不可重排序。图3展示了一对可重排序活动的示例：如果在 PushASN 提交之后或执行之前执行 UpdateAuditInfo，它都可以成功。根据业务逻辑，可以制定流程，限制或重新安排某些活动，使其仅在特定时段执行。例如，图2中的相应流程模型显示，UpdateAuditInfo 经常发生在 PushASN 和 Ship 活动之间，因此，UpdateAuditInfo 可以在另外两个活动对应的交易提交之前执行。不过，UpdateAuditInfo 并非时间关键型活动，可以重新安排，只在供应链流量较低的特定时段执行。我们发现，在相应的重新设计后，交易吞吐量提高了24%，成功率提高了15%；重新设计后，UpdateAuditInfo 和 QueryProducts 活动在 PushASN、Ship、Unload 之后执行。根据区块链日志得出的新流程模型证实了新设计已得到遵循（图4）。因此，通过识别相互冲突的活动，可以在交易冲突发生之前重新设计流程模型，以减少此类冲突。

4 区块链优化推荐器

我们提出了一种方法，可从三个不同的抽象层级推荐优化：系统级、数据级和用户级。设计并实现此类多级推荐系统的首要要求，是具备涵盖这三个层级的可靠数据。有关系统配置（例如区块大小）和性能（例如吞吐量、交易失败）的知识，对于生成系统级建议至关重要。有关当前数据模型和访问模式的信息，例如键分布和依赖关系，是生成数据级建议的必要条件。最后，有关用例、业务流程和事务工作负载的知识，是生成用户级建议所必需的。需要注意的是，这类信息并不局限于某个特定层级，而是

有助于所有层级。例如，系统级性能可以表明三个层级都需要进行优化。

区块链的定义本身就意味着存在一个分布式账本，其中包含每一笔随时间推移而执行的事务的不可篡改数据。如果考虑智能合约，那么每次执行智能合约都会产生一笔记录在账本中的交易。我们将这些数据（下文称为区块链日志）视为得出优化建议的主要来源，因为据我们所知，大多数区块链都提供了包含所有交易的此类分布式账本。因此，我们以交易为中心的方法适用于为不同的区块链提出优化建议。

我们对区块链中的原始数据进行预处理，以生成区块链日志。然后，我们获取用于检测多层级优化建议的关键指标值。接着，将流程挖掘策略应用于区块链日志，以推导出流程模型。我们结合这些建议和推导出的流程模型，确定适用的优化方案。图 5

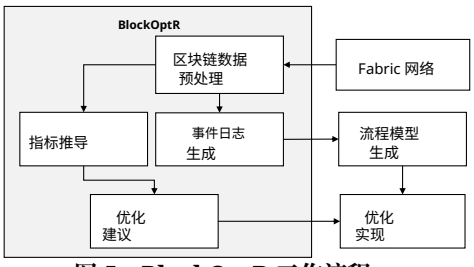

图 5：BlockOptR 工作流程

展示了我们方法的工作流程。我们将该工作流程的主要环节自动化，开发成工具 BlockOptR [10], , 该工具使用 Python 和 Node.js 实现。

4.1 区块链数据预处理

BlockOptR 在 Fabric 网络上注册为客户端，读取整个区块链并将其保存为 JSON 文件。接下来，清理日志，删除无关的配置和设置相关交易，并将其转换为 CSV 格式。Fabric 网络中执行的每笔交易的所有信息都会记录在区块链上。我们从这些大量的日志数据中提取七个属性，并推导出另外两个属性。这些属性使我们能够推导出多项指标，以便提出优化建议。数据预处理步骤的输出是包含以下九个属性的区块链日志。

(1) **客户端时间戳**：客户端生成交易的时间。(2) **活动名称**：执行后创建该交易的智能合约函数名称。A(x) 表示交易 x 的活动名称。(3) **函数参数**：智能合约函数参数的值。(4) **背书节点**：交易所有背书节点的集合。(5) **调用者**：调用交易的所有客户端及其各自所属组织的集合。(6) **读写集**：交易访问（读取或写入）的键集合。交易的读集和写集也会分别保留。RWS(x)、RS(x) 和 WS(x) 分别表示交易 x 的读写集、读集和写集。(7) **交易状态**：交易的状态，可取值为成功、MVCC 读冲突 (MRC)、幻读冲突和背书策略失败。ST(x) 表示交易 x 的状态。(8) **交易类型**：根据读写集推导出的交易类型，可取值为读取、写入、更新、范围读取和删除。交易类型根据读写集推导得出。TT(x) 表示交易 x 的类型。(9) **提交顺序**：区块链日志中交易的顺序，即交易提交到区块链的顺序。

4.2 事件日志生成

从 Fabric 网络中提取的区块链日志可以用作事件日志，应用过程挖掘技术，从而协助推荐用户级优化。然而，与.....不同

流程感知信息系统创建的事件日志 [74], 不同，从区块链中提取的事件日志无法直接获取案例标识。此外，在我们观察到的大多数用例中，并不存在一个可用于所有活动、且能直接作为案例标识的共同属性。因此，我们需要基于领域知识 [4, 8, 17, 19, 44]，为每个用例推导出一个共同元素。由于我们关注的是流程模型的事务视角，因此通过分析日志中可用的函数参数和读写集，为所有活动找出一个共同元素。例如，在 SCM 场景中，productKey 是所有活动的共同元素，也是合适的选择，因为该用例专门用于跟踪多个产品。本文针对所有用例自动完成了共同元素的提取过程，并且可以轻松扩展到其他用例。确定共同元素后，我们将具有相同共同元素值的一组唯一活动定义为一条轨迹。随后，为每条轨迹分配一个新的案例标识。

此外，我们的日志中只能获取客户端发送事务的时间（客户端时间戳）。然而，无法保证客户端发送事务的顺序在事务提交到区块链时仍能保持不变。因此，为了准确推导流程模型，我们使用提交顺序代替时间戳。这样，借助生成的案例标识以及提取/推导出的属性，我们便获得了完整的事件日志。现在，可以对事件日志应用任意流程挖掘技术来推导流程模型。例如，我们使用 Alpha算法推导出图2和 4 [76] 中所示的流程模型。

4.3 指标

我们通过仔细考察多个区块链日志并分析文献中的指标，定义了一组指标。

(1) **速率指标**：BlockOptR 根据事件日志计算平均交易速率，以及各时间间隔内的交易速率分布。**交易速率**（*Tr*）是客户端发送事务的平均速率，根据日志中的事务总数和客户端时间戳推导得出。**交易速率分布**（*Trdi*）是根据日志推导出的特定间隔 i 内的交易速率。计算该指标时，使用以秒为单位、由用户配置的间隔大小（*ins*）。用途：交易速率是了解性能的一项实用指标。速率分布有助于洞察流量高峰或低谷时段。

(2) **故障指标**：与*Tr*类似，**总故障率**（*TFr*）以及各类故障（ MVCC 读冲突、幻读冲突、背书策略失败）的发生率均由日志得出。故障率分布（*Frdi*）的计算方式与*Trd*类似。用途：故障指标有助于检测交易失败率较高的时段。可根据故障指标采取交易速率控制等优化措施。

(3) **区块大小**：从日志中提取用户配置的**区块数量**（B_{count}）和**区块超时**（$B_{timeout}$）。此外，也可从日志中得出**区块中的平均交易数量**（$B_{sizeavg}$）。$B_{sizeavg}$等同于平均区块大小，也可定义为 $min\{B_{count}, Tr * B_{timeout}\}$。用途：$B_{sizeavg}$与速率指标结合，可帮助用户了解其区块大小配置的有效性。例如，如果*Tr* 为 500，B_{count} 为 100，$B_{timeout}$ 为 1，且$B_{sizeavg}$ 为 100，那么打包的每秒实际有 500 笔交易可用时，每个区块却只打包 100 笔交易。这意味着创建的区块多于所需数量，效率低下，因为区块创建的成本很高。同样，如果*Tr*为 100，B_{count} 为 500，$B_{timeout}$ 为 2，且$B_{sizeavg}$ 为 200，那么每 2 秒才创建一个区块，交易会排队等待一段时间后才被打包进区块。这两种情况都会导致性能下降。因此，用户可以根据$B_{sizeavg}$的值，更新B_{count}和$B_{timeout}$，以有效处理交易速率。

(4) **背书者重要性**（*EDsig*）定义每个背书节点背书的交易数量。用途：此指标有助于识别背书者瓶颈。假设总是由数量有限的背书节点执行背书。在这种情况下，用户可以考虑在背书节点之间更均匀地分配交易，或扩大背书节点集合。

(5) **调用者重要性**（*IVsig*）定义每个客户端发起的交易数量。用途：此指标有助于识别发起大多数交易的客户端及其所属组织。可依据此指标制定这些组织的客户端资源分配决策。

(6) **键频率**（*Kfreq*）定义为访问特定键的失败交易数量。**键重要性**（*Ksig*）定义为访问特定键的活动数量。*HK* 根据用户可配置的阈值，定义具有高键频率的热点键集合。用途：识别热点键有助于用户发现智能合约中的优化机会，而键重要性则有助于检测访问热点键的具体活动（对应于智能合约函数）。例如，如果多个函数访问同一个键，就可以将这些不同的函数拆分到多个智能合约中。每个智能合约都在不同的世界状态上执行，从而减少失败（参见第 5 节中的示例）。

(7) **数据值相关性**（*corDV*）定义为：如果两个交易访问同一组键，且其中一个交易失败，则这两个交易相关。用途：数据值相关性有助于识别交易依赖关系。这类存在依赖关系的交易是 MVCC 读冲突的根本原因 [13]。可以对这些相关交易应用多种优化策略，例如流程模型重设计和交易速率控制，以减少失败。

(8) **邻近相关性**（*corP*）用于定义两个数据值相关性较高的交易之间的距离。例如，如果 $corP(x, y) == 1$，那么交易 y 紧接在 x 之后发生，中间没有其他交易。此外，我们还推导出**基于活动的邻近相关性**（*corPA*），用于定义同一活动的交易之间的距离。用法：分析邻近相关性是“小于区块大小”还是“大于区块大小”，可以揭示有关区块间和区块内失败的有用信息。如果区块内失败率很高，较小的区块大小有望减少失败 [13]。此指标还有助于在不同 Fabric 优化所提供的区块间或区块内交易重新排序策略之间做出选择 [65, 67]。

4.4 优化建议

我们采用多层次方法，利用已定义的属性和指标，为区块链推荐特定的优化策略。本节介绍的优化建议技术包含可配置阈值。我们根据

表1：优化建议的形式化

建议	必要条件
活动重排序	如果 $corDV(x,y) == 1 \wedge 有(x) \cap 有(y) == \emptyset$
流程模型剪枝	若 $A(x) = A(y) \wedge TT(x) \neq TT(y)$
交易速率控制	若$(Trd_i \geq Rt_1) \wedge (Frd_i \geq Trd_i * Rt_2)$
增量写入	如果 $corPA(x, y) == 1 \wedge ST(x) == MRC \wedge$ $\lvert 有(x)\rvert == \lvert 有(y)\rvert == 1 \wedge 有(x) \pm 1 == 有(y)$
智能合约分区	如果 $Ksig(HK_i) > 1$
数据模型改动	$if(Ksig(HK_i) == 1) \vee (\lvert HK \rvert == 1)$
区块大小调整	$if(Tr \geq B^{sizeavg} * Bt) \vee (Tr < B^{sizeavg} * Bt)$
背书者重构	如果 $EDsig(e) > \lvert TX \rvert * Et$
客户端资源提升	如果 $IVsig(c) > \lvert TX \rvert * It$
其中 $x, y \in TX, e \in E, c \in I, HK_i \in HK$ TX, E, I, HK 分别是所有交易、背书节点、调用者和热点键的集合 Rt_1, Rt_2, Bt, Et, It 是用户可配置阈值	

对多份日志的分析，为这些阈值定义了合适的默认值，但用户可以调整这些默认值，以微调检测策略。表1对推荐每种优化策略所需满足的条件进行了形式化描述。

4.4.1 用户级建议

在用户级，重点关注正在运行的应用程序的实际工作负载至关重要。交易生成并提交到区块链的速率和顺序对性能有着重大影响。我们分析交易的速率、依赖关系和类型，以推荐用户级优化方案。

(1) **活动重排序**：使用数据值相关性和读写集，可以识别可重排序的交易对。BlockOptR 会识别与此类交易对相对应的活动，并建议进行流程模型重设计。重设计应确保对已识别的活动进行重新组织，以减少冲突（参见第 3 节）。

(2) **流程模型剪枝**：如果活动偏离预期行为，建议进行流程模型剪枝。通过分析与某项活动相关的所有交易的交易类型，识别异常。将事件日志中的轨迹和推导出的流程模型与已识别的异常进行比较，有助于发现模型剪枝的机会（参见第 3 节）。

(3) **交易速率控制**：BlockOptR 会评估交易速率随时间的分布，并识别速率极高的时段。随后，它会检查同一时间间隔内的失败率。如果失败率也极高，则建议进行速率控制。使用两个可配置阈值来调整交易速率和失败的容忍度。

4.4.2 数据层建议

对于数据级建议，我们着重通过分析交易失败、邻近相关性、读写集和键重要性，找出数据模型中可优化的具体部分。这有助于用户修改智能合约及其底层数据模型，从而提升性能。

(4) **增量写入**：仅执行递增或递减操作的更新交易可以转换为增量写入。增量写入允许写入多个唯一的增量键，并可在需要当前值时进行聚合。每次写入前也无需读取该键。因此，更新交易会转换为仅写入唯一键的只写交易。这有助于减少交易依赖相关失败。当失败的交易对单个键执行递增或递减操作时，建议采用增量写入。

(5) **智能合约分区**：减少交易依赖关系的一种方法是将一个智能合约拆分为多个智能合约。每个智能合约都会访问独立的世界状态，从而避免冲突。如果需要交互，可以在两个智能合约之间调用函数，因此原智能合约的功能不会改变。

例如，在音乐版权管理场景中，如果发现键 MusicID 是热点键，且 Play() 和 viewMetaData() 等多个函数都会访问同一个键，就可以将这些函数拆分到两个不同的智能合约中。换句话说，底层数据库被拆分为两个数据库：在两个数据库中复制主键（MusicID），并在每个数据库中设置不同的次级键。MusicID 的播放次数记录在一个数据库中，而元数据则从另一个数据库中读取（参见第6.2节）。这类似于关系数据库中的表布局设计。需要对智能合约进行分析和更新，才能实现这种优化。如果多个活动会访问同一个热点键，则建议采用智能合约分区。

(6) **数据模型变更**：如果活动之间存在依赖关系，那么改动数据模型有助于减少交易冲突。例如，在数字投票场景中，如果发现键 ElectionID 是热点键，且仅由函数 Vote() 访问，那么一种可能的优化方式是使用另一个主键，例如 VoterID。这样，就不必一次性更新所有投票，而可以按选民逐一更新投票（参见第6.2节）。此外，如果检测到的热点键只有一个，那么分析数据模型以了解该特定数据元素访问偏斜的原因会有所帮助（参见第6.3节）。如果某个热点键仅由单一活动访问，或检测到的热点键只有一个，则建议改动数据模型。

4.4.3 系统层建议

在系统级，我们重点关注两项可能显著影响 Fabric 性能的关键系统配置设置：背书策略和区块大小 [13, 68]。此外，我们还会识别客户端瓶颈，以帮助制定资源分配决策。我们使用背书者重要性、调用者重要性、交易速率和实际区块大小等指标，得出系统级优化建议。由于这些建议基于正在运行的应用生成的区块链日志，因此有助于用户根据当前用例和工作负载确定理想的配置设置，从而提升性能。

(7) **区块大小调整**：平均交易速率（Tr）、平均区块大小（$B^{sizeavg}$）和可配置阈值（Bt）用于推荐区块大小调整。文献建议，交易速率较低时采用较小的区块大小，交易速率较高时采用较大的区块大小 [13, 68]。如果区块大小过小，就会创建过多区块，区块创建会成为瓶颈。如果区块大小过大，则需要等待足够的交易，从而延迟区块创建。因此，如果建议调整区块大小，则应设置$Btimeout$ 和$Bcount$ ，使其

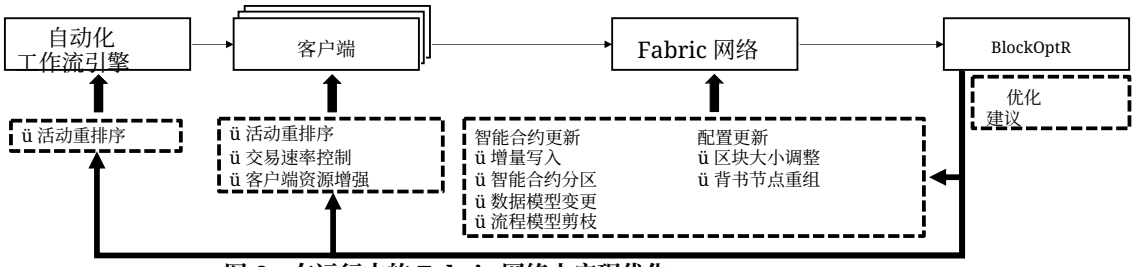

图 6：在运行中的 Fabric 网络上实现优化

满足$min\{B_{count}, Tr * B_{timeout}\}$等于$Tr$。由于难以根据日志准确推导事务的大小（事务可能包含交易载荷、背书节点身份和其他元数据），因此我们不提供区块字节数调整建议。

(8) **背书节点重组**：对于客户端生成的每笔 Fabric 交易，背书策略中指定的背书节点都会执行相应的智能合约函数。智能合约执行会消耗时间和资源。如果某些背书节点接收到更多交易，而其他节点处于空闲状态，则表明存在瓶颈或负载不均衡。当背书策略明确规定必须由一组特定背书节点中的某个节点进行背书时，就可能出现此类负载不均衡。例如，背书策略 And(Org1,OR(Org2,Org3)) 表示必须由 Org1 背书。因此，Org1 可能成为背书瓶颈。我们通过识别背书交易数量超过用户指定阈值的背书节点，来检测背书者瓶颈。默认阈值用于检测所有背书节点是否均等地参与背书流程。调整阈值可以提高或降低对负载不均衡的敏感度。

(9) **客户端资源增强**：Fabric 网络中的客户端需要执行多项耗时任务，包括但不限于交易提案调用、背书者响应验证、将背书者响应打包为交易、向排序服务提交交易，以及收集节点提交响应。调用方重要性指标能够识别发起大多数交易的客户端及其所属组织。这有助于制定资源分配决策，例如增加向已识别组织注册的客户端数量并扩大其规模。这也可能揭示底层业务流程中存在的问题。

4.5 优化的实施

建议的优化可以通过多种方式实施。图 6 展示了在运行中的 Fabric 网络上可以实施不同建议的位置。这里，我们展示了一个根据流程模型触发交易的自动化工作流引擎。这些交易通过客户端发送至 Fabric 网络。BlockOptR 会分析 Fabric 网络的日志，以生成优化建议。如图所示，每项建议的优化都可以在不同层级实施。

活动重排序可通过修改工作流引擎中的底层流程模型来实现，使活动按照无冲突的顺序执行。或者，也可以监控客户端上的交易，并通过客户端管理器按客户端分别重排序，或在所有客户端之间进行重排序。流程模型剪枝可通过组织措施来实现，例如采用激励措施或惩罚措施，确保活动遵循其预期行为（图中未显示）。不过，也可以直接在智能合约中实现剪枝，在背书阶段提前中止异常交易。交易速率控制可以通过多种方式实现。每个客户端都可以监控自身的交易速率，并进行负载削减或排队。也可以使用中央监控器在所有客户端之间执行相同操作。第三种方法是在排序服务中监控交易速率，并在那里进行负载削减。要实现所有数据级优化，都需要修订智能合约。在 Fabric 中，无需重启系统即可动态升级智能合约 [72]。可以通过更改配置文件或在 Fabric 中使用 配置更新 事务来调整区块大小 [71]。可以通过更改背书策略来重组背书节点。可以在 Fabric 配置文件中更改策略，也可以使用 配置更新 事务进行更改 [71]。根据 BlockOptR 确定的每个客户端的交易负载，如果当前资源分配似乎不足以应对负载，则可以扩展客户端资源，并将新客户端动态注册到 Fabric 网络中。

我们的实现。尽管所有优化都可以在实时系统中动态应用，但由于我们的评估是在实验环境中进行的，因此每次实验后我们都会重启 Fabric 网络。我们使用 Caliper基准测试系统 [35]，该系统配有客户端管理器，可配置跨客户端的交易排序并控制生成交易的速率，从而模拟活动重排序和交易速率控制。我们还可以增加客户端数量，以展示客户端资源增强。流程模型剪枝和所有数据级优化均通过分析和修改智能合约实现。区块大小和背书策略则在 Fabric 配置文件中更新。

5 实验方法

我们使用 Hyperledger Lab. [13], 是一个自动化测试平台，适用于 Hyperledger Fabric 2.2，并集成了 Caliper基准测试系统。我们搭建了一个 Kubernetes集群，其中有 1 个主节点和 5 个工作节点，所有 Fabric 网络组件和 Caliper 组件均以 Kubernetes pod 的形式分布其中。每个节点都运行在一台配备 4 个虚拟CPU 和 9.8 GB RAM 的 Ubuntu Focal (20.04) 虚拟机上。实验中我们使用 10 个 Caliper 工作进程。对于每项实验，我们都会测量成功率（即成功交易占交易总数的百分比）、平均延迟，以及所有成功交易的吞吐量。

表 2：控制变量

控制变量	值（默认值以粗体显示）
工作负载类型	**均匀**，读密集型，插入密集型、更新密集型，范围读取密集型
背书策略	P1、P2、**P3**、P4
背书者分布偏斜	**0**, 6
键分布偏斜	**1**, 2
组织数量	**2**, 4
区块数量	50，**300**，1000
发送速率	50，**300**，1000
交易分布偏斜	**0**，70%

成功交易占交易总数的百分比、平均延迟，以及所有成功交易的吞吐量。

5.1 工作负载生成

分布式账本的内容作为我们工具的输入，直接取决于区块链上执行的工作负载。因此，我们使用三种不同类型的工作负载，对 BlockOptR 进行了全面评估。此外，在实施 BlockOptR 生成的建议后，我们使用相同的工作负载重新运行实验，以分析优化的效果。*5.1.1* 合成工作负载

我们使用一种扩展版的合成工作负载生成器，它可以根据一组控制变量，为通用智能合约生成合成工作负载 *genChain* [13]。我们采用表 2 中所述这些控制变量的一系列值，生成多个工作负载，每个工作负载包含 10,000 笔交易。实验中使用的背书策略如下：P1：And(Org1, Or(Org2,Org3,Org4)) P2：And(Or(Org1,Org2), Or(Org3,Org4)) P3：Majority(Org1,...,OrgN) P4：OutOf(2,Org1,Org2,Org3,Org4)

通过生成合成工作负载，我们确保实验涵盖多个真实场景。随后，我们使用这些工作负载逐一评估 BlockOptR，以生成优化建议。此外，我们还实施每项建议的优化，以评估性能提升。

5.1.2 基于用例的工作负载

其次，我们采用文献中的四种热门基于用例的智能合约的扩展版本 [13]，并生成工作负载。随后，使用 BlockOptR 基于这些工作负载生成优化建议。我们使用的四种智能合约如下。

供应链管理（*SCM*）：此智能合约定义了物流网络的各项操作，包括发送预先发货通知、发货、读取发货通知以及卸货（按此顺序）。此外，还提供查询操作，用于查询不同产品的信息（queryProducts），并提供 updateAuditInfo 函数，用于更新包含产品详情的审计条目。这些操作可以随时发生。基于上述假设，我们生成了包含 10,000 笔交易的工作负载：依次发送 pushASN、Ship、queryASN 和 Unload 交易，同时随机发送 QueryProducts 和 UpdateAuditInfo 交易。

数字版权管理（*DRM*）：该智能合约用于管理音乐产业中艺术家的权利。该智能合约包含一个*Play* 函数，每当任何用户播放一首音乐作品时，都会执行该函数。其他智能合约函数包括

表 3：合成工作负载实验

实验数量	控制变量	值	推荐的优化
1	背书策略	P1	背书节点重组 活动重排序
2	背书策略 / 背书者分布偏斜	P2 / 6	背书节点重组 活动重排序
3	组织数量	4	交易速率控制
4	工作负载	读密集型	活动重排序
5	工作负载	更新密集型	交易速率控制
6	工作负载	插入密集型	活动重排序
7	工作负载	范围读取密集型	活动重排序
8	Key 分布偏斜	2	交易速率控制 活动重排序 智能合约分区
9	区块数量	50	区块大小调整 活动重排序
10	区块数量	300	交易速率控制 活动重排序
11	区块数量	1000	交易速率控制 活动重排序
12	发送速率	50	活动重排序
13	发送速率	300	活动重排序
14	发送速率	1000	区块大小调整 交易速率控制 活动重排序
15	事务分布偏斜	70%	交易速率控制 活动重排序 客户端资源增强

添加新的音乐作品、查询权利、查看元数据以及计算权利持有人的收益。在实际场景中，*Play* 事务的执行频率远高于所有其他事务。因此，我们为此用例创建了一个以*Play* 为主的工作负载。我们随机生成 10,000 笔事务，其中 70% 的事务为*Play*。其余 30% 则由所有其他事务构成，并以均匀随机的方式生成。

电子健康记录（*EHR*）：在此智能合约中，患者可以向医疗机构和研究机构授予或撤销访问其医疗记录的权限。我们假设患者数量将多于其他参与者，并生成一个包含 10,000 笔事务、其中 70% 为更新密集型的工作负载。

数字投票（*DV*）：此智能合约包含用于选举投票、查询政党、查询结果以及结束选举的功能。我们可以假设，在实际选举期间，投票进行时会出现高流量时段。因此，我们生成一个工作负载：首先以每秒 100 笔事务的速率发送 1,000 笔查询政党交易，然后以每秒 300 笔事务的速率发送 5,000 笔投票交易，最后分别发送 1 笔查看结果交易和 1 笔结束选举交易。

5.1.3 贷款申请流程（*LAP*）

第三，我们利用荷兰某金融机构公开的贷款申请流程真实事件日志，创建了智能合约和工作负载 [77]，并使用了相应的流程图 [57]。我们提取了前 2,000 笔贷款申请的所有事件，并创建了相应的 20,000 笔交易。随后，我们创建了一个智能合约，其中贷款申请流程图中的每项活动都对应一个智能合约函数。事件日志中包含负责处理贷款申请的银行员工的员工ID，以及银行处理的每笔贷款申请的申请ID。我们实现的智能合约使用员工ID作为键，键的值是一个结构体数组，每个结构体都包含

表 4：实现优化的设置

优化 推荐	设置
活动重排序	重排序工作负载生成
交易速率控制	将发送速率设为 100 每秒交易数
流程模型剪枝	更新智能合约
增量写入	
智能合约分区	
数据模型变更	
区块大小调整	将区块数量设置为推导出的交易速率
背书节点重组	将背书策略设置为 P4
客户端资源增强	将推荐组织的客户端数量加倍

申请ID、贷款类型、贷款金额和贷款状态。因此，查询特定员工ID即可轻松获取该员工处理的所有贷款申请。随后，我们以每秒 10 笔交易的低速率在智能合约上执行了这 20,000 笔交易，以模拟人工处理贷款申请耗时较长的真实场景。我们还以每秒 300 笔交易的较高速率进行了相同的实验，以模拟自动化的贷款申请与验证流程。我们使用 BlockOptR 生成优化建议，以改进智能合约实现，从而提升性能。

尽管 LAP 事件日志来自数据库环境，但这对于区块链而言是一个现实的用例，因为自动化贷款申请系统需要安全性和去中心化信任（例如小额贷款、去中心化贷款申请，以及更广义上的去中心化金融 [33, 58, 82, 83]）。因此，这项实验展示了 BlockOptR 在现实场景中的实用性。在基于用例的实验中，所有交易都按照我们基于所定义假设预期的顺序执行。相比之下，对于这一真实事件日志，我们评估交易实际执行的顺序，而该顺序可能会偏离流程模型。

6 实验结果

我们通过各种工作负载和智能合约，对推荐方法进行了全面评估。请注意，实施交易速率控制时，吞吐量预计会下降。不过，客户端会因成功率提高而大为受益，而表面上的吞吐量下降，只是使其更接近系统的可持续吞吐量。在所有实验中，阈值的默认值为 $Et = 0.5$，$Rt1 = 300$，$Rt2 = 0.3$，$Bt = 0.6$ 和 $It = 0.5$。表 4 展示了所有设置，包括为实施每项推荐优化而更改的控制变量值。

6.1 合成工作负载

受篇幅限制，我们在表 3 中展示了 15 个工作负载。完整的实验列表和结果可在我们的代码仓库 [10] 中查看。每项实验所调整的控制变量及其值一并列出。其他所有控制变量均采用表 2 所示的默认值。实验 1 至 15 均在未应用任何优化的情况下进行，随后使用 BlockOptR 得出优化建议。BlockOptR 生成的建议也列于表 3 中。由于合成智能合约的逻辑简单，没有分支、递增/递减操作或复杂的数据模型，因此此处不建议进行流程模型剪枝、增量写入和数据模型变更。接下来，我们实现建议的优化并重新执行所有实验。实验结果根据优化建议分组，见图 7、8、9、10、11 和 12。我们还将说明实验中阈值的设置方式，以及用户如何对其进行调整。

6.1.1 背书节点重组：背书节点重组的影响如图 7 所示。当背书策略为 P1 时，由于特定的背书策略，所有客户端都必须将交易发送至 Org1，因此 Org1 出现了背书瓶颈。由于背书策略要求来自两个组织的签名，我们将策略更改为 OutOf(2,Org1,Org2,Org3,Org4)，这样客户端便可将交易均匀分配给所有背书节点。这项优化使吞吐量提升了 29%（图 7）。在实验 2 中，由于背书节点分布存在偏斜，客户端发送交易的分布不均，因此其中两个组织的背书频率远高于另外两个组织。我们重新执行了实验，将交易均匀分配给背书节点，观察到吞吐量提升了 26%（图 7）。这项优化的主要影响体现在吞吐量和延迟上，因为它减少了少数特定对等节点上的交易排队，并将交易均匀分配到各节点。

我们设置了此项建议的阈值，使事务能够均匀分配给所有背书节点，也就是说，即使是轻微的瓶颈也能被检测到。可以调整这些阈值，使其仅检测严重瓶颈。此外，由于这些是合成实验，更改背书策略并非关键。在实际场景中，更改策略前必须征求企业管理机构的意见。尽管如此，Block-OptR 的建议有助于凸显瓶颈，进而说服管理层更改策略。

6.1.2 客户端资源增强：图 8 展示了客户端资源扩展的效果。增加客户端数量后，我们观察到延迟降低了 75%，吞吐量提高了 15%，成功率提高了 7%。阈值的设置使得当同一组织发起的事务超过 50% 时，系统会推荐此项优化。可以进一步微调阈值，以检测不那么严重的瓶颈。

6.1.3 区块大小调整：区块大小调整的效果如图 9 所示。在实验中，我们使用默认的 1 秒区块超时时间。因此，每当建议调整区块大小时，我们都会使区块数量等于交易速率。调整区块大小后，我们观察到吞吐量最高提升了 93%，成功率最高提升了 85%（图 9；区块数量：50）。阈值的设置方式是：当日志推导出的平均区块大小比交易发送速率大或小 60% 时，系统便会建议进行此项优化。降低阈值可以让建议对交易速率变化更加敏感。

6.1.4 交易速率控制：交易速率控制的效果如图 10 所示。在这些实验中，高流量时段（约 300 每秒交易数）也被认定为故障率较高的时段。随后，我们将客户端的交易发送速率降至 100 每秒交易数，并重新执行了实验。结果显示，延迟最多改善了 87%，成功率提高了 36%（图 10；发送速率：1000）。我们将此建议的阈值设为 300 每秒交易数，这是实验中的默认发送速率。这意味着我们将当前流量视为

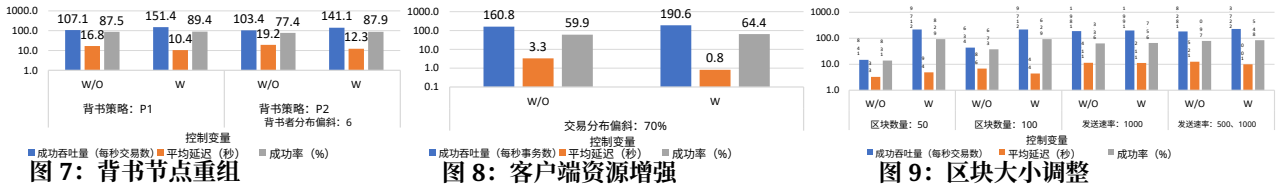

图 7：背书节点重组

图 8：客户端资源增强

图 9：区块大小调整

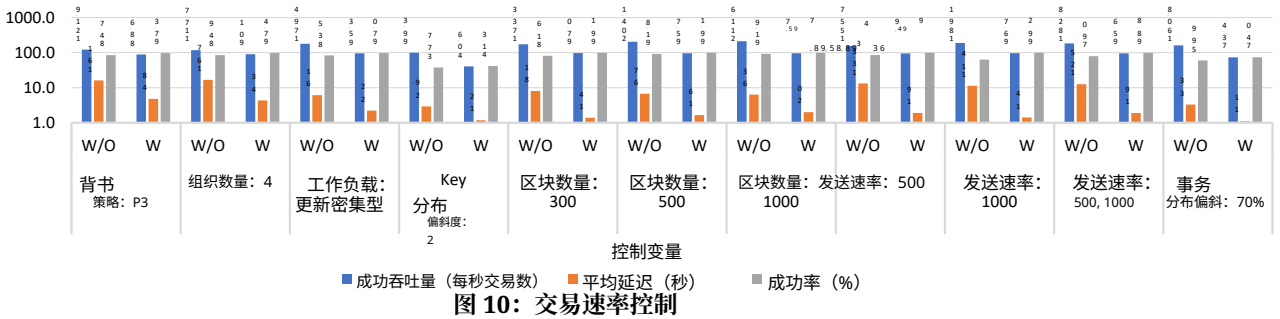

图 10：交易速率控制

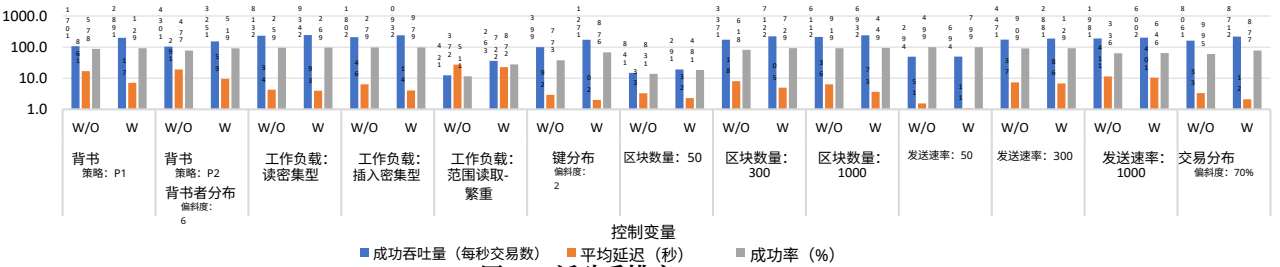

图11：活动重排序

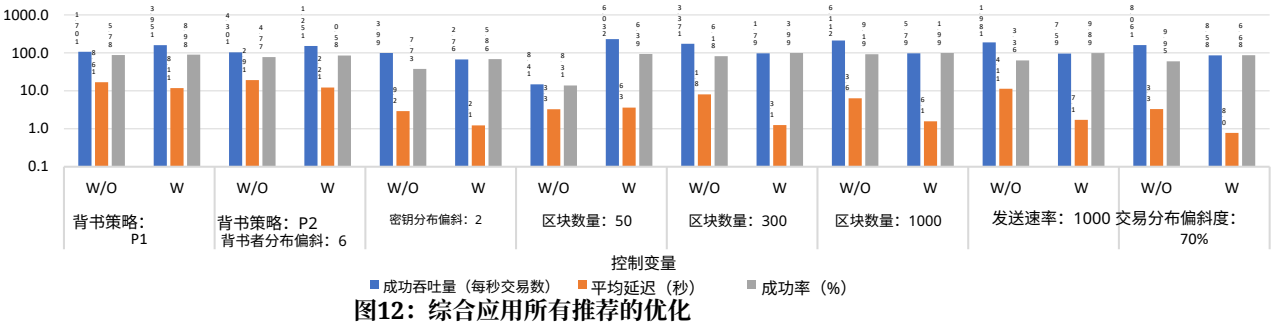

图12：综合应用所有推荐的优化

系统将其视为高流量，并希望检测故障时段。用户可以根据其 Fabric 网络部署中对高流量的定义（高于可持续流量速率）调整此阈值。

6.1.5 活动重排序：活动重排序的效果见图11。我们观察到，除实验3和5外，BlockOptR 在所有实验中都建议进行活动重排序（表 3）。建议对两个彼此冲突的活动（读取和更新）进行重排序。我们更新了客户端管理器的配置，使其在所有其他交易之前生成读取交易。此实现模拟了通过组织措施强制进行活动重排序的场景。随后，我们重新执行实验，并观察到所有实验的性能均有所提升。吞吐量最高提升了 65%，成功率最高提升了 58%（图11；工作负载：范围读取密集型）。我们设置的阈值是：如果 40% 的 MVCC失败是由可重排序的活动导致的，就建议采用此策略。提高阈值可以使这一策略更加宽松，也就是说，只有在影响非常显著的情况下才建议重排序。对于实验3和5，由可重排序的两个活动导致的 MVCC冲突不到 40%。例如，活动“更新”依赖于自身，这种依赖关系无法通过重排序消除。

6.1.6 综合优化：我们还在综合应用所有推荐的优化后进行了实验。我们观察到吞吐量最高提升了 93%，成功率最高提升了 85%（图12：区块数量：50）。在所有实验中，综合应用所有优化所获得的性能与提升幅度最高的单项优化所带来的性能相当。

进一步说明。尽管针对实验8建议进行智能合约分区，但这种优化需要了解智能合约的功能。遗憾的是，对于仅包含通用读取、更新和插入函数的合成智能合约，我们无法重新设计该智能合约。

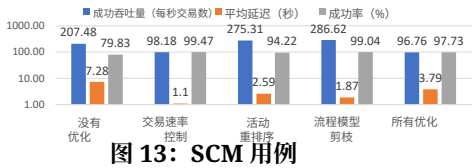

图 13：SCM 用例

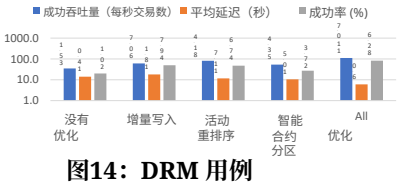

图14：DRM 用例

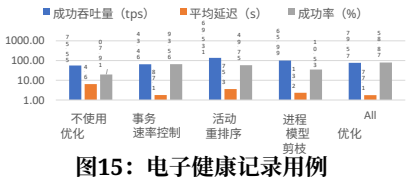

图15：电子健康记录用例

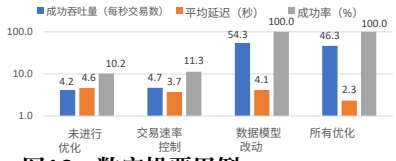

图16：数字投票用例

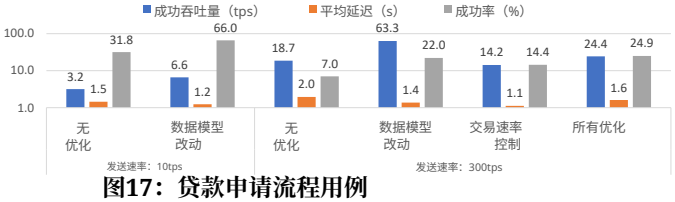

图17：贷款申请流程用例

6.2 基于用例的工作负载

供应链管理（*SCM*）：对于 SCM 用例，BlockOptR 建议采用三种优化：活动重排序、流程模型剪枝和交易速率控制（图 13）。对可重排序活动（query-Products 和 UpdateAuditInfo）进行重排序后，我们观察到吞吐量提高了 24%，成功率提高了 15%。建议对在 PushASN 活动之前发生或未发生 PushASN 活动时执行的 Ship 活动进行剪枝。此外，还建议对在 Ship 活动之前发生或未发生 Ship 活动时执行的 Unload 活动进行剪枝。我们调整了智能合约，以实施剪枝建议。这样一来，吞吐量提高了 27%，成功率提高了 19%。交易速率控制以及一并应用所有建议，也能提升性能。

数字版权管理（*DRM*）：在 DRM 用例中，BlockOptR 推荐了三种优化：活动重排序、增量写入和智能合约分区。图 14展示了应用这些优化后的结果。为了实施增量写入建议，我们发现智能合约中的 Play 函数包含一项递增操作，用于统计某个音乐作品的播放次数。我们将其转换为增量写入，并在调用 calcRevenue 函数时聚合增量键（因为该函数需要播放次数）。通过这一优化，吞吐量显著提升了 42%，成功率提升了 50%。不过，在这种情况下，平均延迟有所增加，因为 calcRevenue 函数现在需要花费更多时间进行聚合。由于 calcRevenue 的执行频率不如 Play 高，因此整体性能并未受到影响。

建议对 calcRevenue 和 queryRightHolders 函数进行活动重排序，并重新配置客户端，使其在所有其他活动之后发送这些活动。这模拟了组织将特定交易限制在特定时间段内的场景。我们发现，采用此优化后，吞吐量和成功率均提升了 50% 以上。

检测到四个活动频繁使用热点键。我们分析了智能合约，发现尽管这四个
这些函数依赖同一个键，但功能各不相同。Play 和 calcRevenue 只需要播放次数，而 viewMetaData 和 queryRightHolders 需要音乐作品的元数据，而不需要播放次数。因此，我们将智能合约拆分为两个：一个智能合约包含 Play 和 calcRevenue 函数，另一个智能合约包含

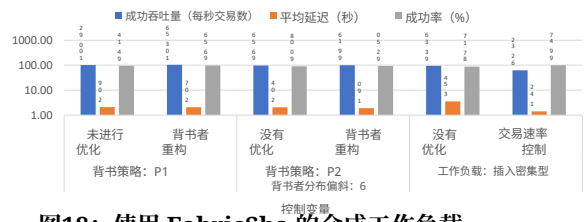

图18：使用 FabricSha 的合成工作负载

另外两个函数。create 函数包含在两个智能合约中，调用第一个智能合约时，也会调用第二个智能合约中的同一函数。我们观察到，采用这项优化后，吞吐量提升了 35%，成功率提升了 26%。综合应用所有优化后，性能提升超过 50%。

电子健康记录（*EHR*）：在此用例中，建议采用三种优化方法：活动重排序、流程模型剪枝和交易速率控制（图15）。对读取活动进行活动重排序后，吞吐量和成功率提高了 60–65%。将智能合约更新为剪除不合逻辑的路径（撤销对记录的访问权限，且不授予访问权限）后，我们观察到吞吐量和成功率提高了约 43%。应用交易速率控制后，成功率提高了 69%。同时应用所有优化也能提升性能。

数字投票（*DV*）：在此用例中，推荐了两项优化：交易速率控制和数据模型变更。结果如图16所示。在投票交易频繁的时段，检测到较高的失败率。应用交易速率控制后，吞吐量略有提升，增幅为11%。检测到热点键由投票函数最频繁地使用，因此建议改动数据模型。我们分析了智能合约，发现 partyID 被用作投票函数的键，而该函数会在投票阶段由多个投票者调用。我们重新设计了智能合约，将 voterID 设为主键。由于投票者只能投票一次，使用这一新智能合约后，我们观察到100%的成功率，因为不再存在更多交易依赖。我们还观察到，同时应用两项优化后，性能有所提升。

6.3 贷款申请流程（LAP）

The LAP 用例是数据模型变更（图17）。员工ID 1 的键值较高
e 项优化建议

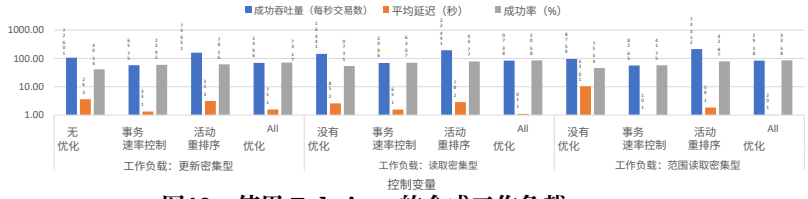

图19：使用 Fabric++ 的合成工作负载

由于该员工处理的贷款申请数量最多，因此频率最高。随后，我们重新实现了智能合约，将 applicationID 指定为键，并将值建模为一个包含 employeeID、贷款金额、贷款类型和贷款状态的结构。这个新实现消除了热点键，使较低和较高发送速率下的吞吐量和成功率均提升了 50% 以上。

6.4 Fabric扩展

作为一种整体性的推荐方法，我们的工作与现有文献中的 Fabric 优化正交。本节展示了我们的方法如何基于 Fabric 的两种优化扩展运行：FabricSharp [65] 和 Fabric++ [67]。二者实现了不同的交易重排序策略，以缓解 MVCC 读冲突。Fabric++调度器已集成到 FabricSharp 实现 [25] 中，我们在实验中使用了该实现。我们在这两种扩展上运行合成工作负载，然后使用 BlockOptR 生成推荐。文献指出，FabricSharp 会增加背书策略失败的情况，并且在插入密集型工作负载下性能较差；而 Fabric++ 在更新密集型、读密集型和范围读取密集型工作负载下性能最差 [13]。因此，我们使用合成工作负载开展了图18和图19所示的这些特定实验。推荐的方法包括活动重排序、交易速率控制和背书节点重组。实施这些建议后，我们观察到吞吐量最多提升了 55%，成功率最多提升了 46%（图 19：范围读取密集型工作负载）。针对这些 Fabric 扩展的实验表明，即使采用了有效的系统级优化，Fabric 仍能从各个抽象层级的优化中获益。

7 经验教训与局限性

我们证明了 BlockOptR 能够有效地推荐合适的优化策略。此外，我们还说明了如何实施这些优化，并量化了实施后的性能提升。本节将讨论我们从实验中获得的见解。

用户级优化。 在我们的实验中，活动重排序是最常被推荐的优化之一。我们重点介绍了 SCM 等适用此类重排序的用例。我们的模型剪枝建议强调，识别流程模型中的不足，不仅能提高流程执行效率，还能提升底层系统的性能。当系统无法处理工作负载时，通常会采用负载削减或排队。根据我们的建议，可以找出最适合采用此类速率控制技术的特定活动和时间段。例如，在数字投票用例中，建议对投票活动进行速率控制。因此，无需在整个系统范围内实施速率控制，只需处理已识别活动的特定客户端采用速率控制技术即可。

数据级优化。 这些优化展示了智能合约和数据模型的设计如何显著影响性能。智能合约最初是根据特定的流程模型设计的。然而，通过分析区块链日志，我们可以了解智能合约在实践中的使用方式。BlockOptR 能够找出导致瓶颈的函数和键，从而帮助智能合约开发者进行适当的修改。

系统级优化。 设置背书策略是一项管理层决策，而设计区块链的技术团队往往不会参与相关讨论。我们的建议强调，管理层和技术团队需要共同讨论，以确定最优的配置设置。此外，我们还说明了验证策略是否得到有效使用的必要性。例如，即使策略规定背书应均匀分配，客户端发送交易的方式也可能存在偏斜。在这种情况下，我们建议采取管理措施，例如在客户端之间均匀分配背书节点，使一个组织的客户端仅向特定背书节点发送交易。BlockOptR 也可以检查是否遵守了此类措施。区块大小优化在文献中经常被讨论，并与系统的交易速率相关联 [13, 36, 68]。我们不采用使用交易速率监控器等系统级改动，而是从日志中推导交易速率和实际区块大小。这有助于了解流量模式随时间的变化，并找到合理的区块大小设置。尽管文献主要关注优化 Fabric 的对等节点和排序服务组件 [27, 65, 67]，但我们针对客户端提出的建议强调，同样需要关注客户端优化。

技术独立性。 我们以 Fabric 区块链为例，展示了多层级推荐方法。由于众多区块链系统之间存在大量实现差异，且分布式账本的内容也相应不同，因此很难实现技术独立性。不过，我们列举了一些具体示例，可为未来的研究人员提供参考，帮助他们将我们的方法应用于其他区块链系统。在 Quorum 中，区块时间或挖矿频率对交易延迟的影响呈线性比例关系 [7]，这与我们的区块大小调整推荐策略类似。此外，Corda 采用公证节点来认证交易；预计将交易分发给多个公证节点能够提高吞吐量 [18]。这同样类似于我们的背书重构推荐。此外，文献中还提出了许多针对以太坊智能合约的燃气费降低和漏洞检测策略 [54]，这些策略可转化为我们在数据级提出的建议。

Lorikeet 和 Caterpiller 等工具可以自动将进程模型转换为以太坊智能合约并执行这些合约，从而更便于实施我们建议的用户级优化 [43, 69]。

局限性。 BlockOptR 推荐的优化需要由用户手动实施。可以构建一个带有反馈回路、能够自动实施这些推荐的自适应系统。然而，在企业场景中，对于背书节点重组、活动重排序和流程模型剪枝等许多优化，在实施之前可能需要获得管理层批准。此外，对于不遵循特定流程模型的应用，事件日志可能会产生误导。在此类场景中，活动重排序和流程模型剪枝等用户级优化并不适用。因此，需要具备用例相关的领域知识，才能恰当地实施所推荐的优化措施。进一步而言，在此类基准测试场景中，我们对交易速率控制等部分优化的实现较为简单，未考虑现实世界中的额外开销。不过，这些实现主要用于演示。我们的工作重点在于 BlockOptR 所采用的多层级推荐方法，而不是优化措施的具体实现。最后，我们在实施和未实施所推荐优化的情况下进行的实验，均采用由相同输入参数生成的相似工作负载；也就是说，我们假设优化应用后，工作负载模式仍会延续此前的趋势。然而，在工作负载发生波动或优化实施被延迟的场景中，可能需要重新执行 BlockOptR，以生成新的推荐结果。

8 相关工作

文献提出了多种 Fabric 优化策略，例如交易重排序 [13, 65, 67], 区块大小优化 [13, 36], 无冲突复制数据类型 [52],以及对各种组件进行并行化 [27]。我们的工作与这些优化策略正交，重点在于优化推荐方法。我们展示了如何将我们的建议与文献中的两种优化策略结合使用，以进一步提升性能。

数据库领域也对索引和查询优化进行了广泛研究，其中包括自调优系统和推荐系统 [2, 14, 15, 38, 73]。尽管我们可以从这些研究中借鉴思路，但我们的工作聚焦于区块链特定优化建议。不同的配置设置（如区块大小和背书策略）以及智能合约的概念，为推荐方法引入了新的维度，而这些维度对于数据库而言并非必需。

目前已有研究探索将流程挖掘技术应用于区块链，以获取流程层面的洞见 [24, 32, 39, 49]。Klinkmüller 等人和 Mühlberger 等人介绍了从以太坊区块链中提取流程数据的不同方法。Hobeck 等人 [32] 使用流程挖掘分析基于以太坊的博彩应用，以找出该应用的不足之处。目前，区块链上的流程挖掘仅关注无许可区块链，因为它们可公开访问。然而，推导和研究流程模型对于私有区块链同样至关重要，因此，我们的工作为这一较少受到探索的研究领域作出了贡献。此外，与相关研究不同，我们关注的是利用流程挖掘推荐区块链优化策略。我们仅找到一篇使用许可型区块链的论文，其中 Duchmann 等人 [24] 从 Fabric 中提取流程数据，并检测智能合约中的语义错误。尽管我们的工作与之相似，但我们不仅提取流程数据，还从 Fabric 中提取区块链特定属性、推导多项指标，并推荐优化策略。

数据库领域对数据感知型业务流程进行了广泛研究，这类流程倡导从业务流程视角审视数据库管理系统 [11, 20, 34]。Calvanese 等人 [11] 全面综述了这一领域的研究成果，并整理了数据库理论和流程管理等多个领域的贡献。这些研究促使我们从业务流程视角看待区块链。然而，我们的工作带来了新的贡献，因为区块链除了涉及数据之外，还涉及智能合约和背书策略等其他多个要素。

9 结论

本文展示了全面审视区块链优化的必要性和有效性。我们基于从区块链日志中提取的若干指标和属性，定义了一种多层级推荐方法。我们针对区块链的系统级、数据级和用户级，总共定义了九项优化。基于这些概念，我们实现了自动化优化推荐工具 BlockOptR。此外，我们还展示了如何实施这些优化来提升系统性能。实施推荐的优化后，我们观察到成功率平均提升了 20%，延迟平均降低了 40%。我们使用涵盖多个真实场景的各类工作负载，对系统进行了广泛评估。我们希望能激励企业利用我们的成果来发现区块链优化策略，并为该领域的进一步研究贡献经过匿名化处理的实时区块链日志。BlockOptR 工具、所有智能合约、工作负载生成脚本以及所有事件日志均以开源形式提供 [10]。我们还计划扩展该工具，纳入更多优化建议。

关于未来工作，我们目前正在开发一个 ProM 插件，为 BlockOptR 提供用户友好界面。目前，BlockOptR 的阈值设置取决于业务网络配置。例如，在我们的配置中，速率阈值为 300 每秒交易数，因为更高的速率会导致不稳定，但其他部署的阈值可能有所不同。因此，未来可以扩展 BlockOptR，使其能够自动调整这些阈值。另一个有趣的扩展方向是定义应用程序可以记录的其他属性，从而为优化建议提供更多数据。此外，研究工作负载波动的影响，以及应用优化建议时的延迟，也是另一个具有挑战性的未来方向。

致谢

本研究部分得到了德国研究联合会（DFG，德国研究联合会）项目 392214008 以及巴伐利亚自由州巴伐利亚州合作研究计划项目 DIK-2002-0013//DIK0114/02 的资助。

参考文献

[1] Parinaz Ameri。2016 年。数据库索引推荐面临的挑战：针对 NoSQL 数据库的具体评估。载于德国数据库基础研讨会（Grundlagen von Datenbaken）第 28 届 GI-Workshop，德国诺尔滕-哈登贝格。[2] Parinaz Ameri。2016 年。数据库自调优索引推荐方法。载于 2016 年 IEEE 第 32 届数据工程国际会议研讨会（ICDEW）。 201 205 10 1109 2016 7495648 –。https://doi.org/。/ [3] 2021 ICDEW。。Mohammad Javad Amiri、Divyakant Agrawal 和 Amr El Abbadi。。许可型区块链：属性、技术与应用。计算机协会，纽约州纽约市，美国。https://doi.org/ 10 1145 3448016 3457539 。/。格拉纳达市议会 [4] 2020 投诉流程分析。 h t t p s:// w w w. t f- p m.org/resources/casestudy/analyzing-the-complains-prociess-at-granada-city-council.pdf。（2020 年）。[在线；访问日期：07-July-2022]。[5] Elli Androulaki、Artem Barger、Vita Bortnikov、Christian Cachin、Konstantinos Christidis、Angelo De Caro、David Enyeart、Christopher Ferris、Gennady Laventman、Yacov Manevich、Srinivasan Muralidharan、Chet Murthy、Binh Nguyen、Manish Sethi、Gari Singh、Keith Smith、Alessandro Sorniotti、Chrysoula Stathakopoulou、Marko Vukolić、Sharon Weed Cocco 和 Jason Yellick。2018 年。Hyperledger Fabric：面向许可型区块链的分布式操作系统。载于第十三届 EuroSys 会议论文集（EuroSys '18）。ACM，纽约州纽约市，美国，文章 30, 15 10 1145 3190508 3190538 页。https://doi.org/。/。 [6] Arati Baliga、Nitesh Solanki、Shubham Verekar、Amol Pednekar、Pandurang Kamat 和 Siddhartha Chatterjee。2018 年。Hyperledger Fabric 的性能表征。载于 2018 年加密谷区块链技术会议（CVCBT）。 65 74 10 1109 2018 00013 –。https://doi.org/。/ [7] CVCBT。。Arati Baliga、I Subhod、Pandurang Kamat 和 Siddhartha Chatterjee。2018 年。Quorum 区块链平台的性能评估。（2018 年）。https://doi.org/ 10 48550 1809 03421 。/ARXIV。。Dina Bayomie、Iman Helal、Ahmed [8] Awad、Ehab Ezat 和 Ali Elbastawissi。2015 年。推断无标签事件日志的案例 ID，第 256 卷。https://doi.org/10.1007/978-3-319-42887-1 20[9] Sara Bergman、Mikael Asplund 和 Simin Nadjm-Tehrani。2020 年。许可型区块链与分布式数据库：一项性能研究。Concurrency and Computation: Practice and Experience 32，12（2020），e5227。https://doi.org/10.1002/cpe.5227 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5227 e5227 cpe.5227。[10] BlockOptR 2022。https://github.com/anonysubm/BlockOptR。（2022 年）。[在线；访问日期：13 -April-2022]。[11] Diego Calvanese、Giuseppe De Giacomo 和 Marco Montali。2013 年。数据感知流程分析基础：数据库理论视角。载于第 32 届 ACM SIGMOD-SIGACT-SIGAI 数据库系统原理研讨会论文集（PODS '13）。计算机协会，纽约州纽约市，美国， 1 12 10 1145 2463664 2467796 –。https://doi.org/。/。Celonis 流程挖掘。 [12] https://www.celonis.com/。（2022 年）。[在线；访问日期：07-April-2022]。[13] Jeeta Ann Chacko、Ruben Mayer 和 Hans-Arno Jacobsen。2021 年。为什么我的区块链交易会失败？Hyperledger Fabric 研究。载于 2021 年数据管理国际会议论文集（SIGMOD/PODS '21）。计算机协会，纽约州纽约市，美国，221–234。https://doi.org/ 10 1145 3448016 3452823 。/。 [14] Gloria Chatzopoulou、Magdalini Eirinaki 和 Neoklis Polyzotis。2009 年。面向交互式数据库探索的查询推荐。载于《科学与统计数据库管理》，Marianne Winslett（编）。Springer Berlin Heidelberg，柏林，海德堡，3–18。[15] Surajit Chaudhuri 和 Vivek Narasayya。2007 年。自调优数据库系统：十年进展。载于第 33 届超大型数据库国际会议论文集（VLDB '07）。VLDB基金会，3–14。[16] Mohammad Jabed Morshed Chowdhury、Alan Colman、Muhammad Ashad Kabir、Jun Han 和 Paul Sarda。2018 年。区块链与数据库之比较：批判性分析。载于 2018 年第 17 届 IEEE 计算与通信信任、安全与隐私国际会议/第 12 届 IEEE 大数据科学与工程国际会议（TrustCom/BigDataSE）。1348–1353。https://doi.org/10. 1109 2018 00186 /TrustCom/BigDataSE。。将多个列组合为案例 ID [17] 2020。https://fluxicon.com/book/read/perspectives/# combining-multiple-columns-as-case-id。（2020 年）。[在线；访问日期：07-July-2022]。[18] Corda 2022。 https://docs.r3.com/en/platform/corda/4.7/open-source/key-concepts-notaries.html。（2022 年）。[在线；访问日期：02-April-2022]。[19] 数据要求：案例 ID 2020。 https://fluxicon.com/book/read/dataext/#case-id。（2020 年）。[在线；访问日期：07-July-2022]。[20] Daniel Deutch 和 Tova Milo。2011 年。追求美与财富（或数据库研究人员的业务流程）。载于第 30 届 ACM SIGMOD-SIGACT-SIGART 数据库系统原理研讨会论文集（PODS '11）。计算机协会，纽约州纽约市，美国，1–12。https://doi.org/10.1145/1989284.1989286

[21] Claudio Di Ciccio, Alessio Cecconi, Marlon Dumas, Luciano García-Bañuelos, Orlenys López-Pintado, Qinghua Lu, Jan Mendling, Alexander Ponomarev, An Binh Tran 和 Ingo Weber。2019年。支持协作业务流程的区块链。Informatik Spektrum 42, 3 (2019), 182–190。[22] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi 和 Ji Wang。2018年。解读区块链：区块链系统的数据处理视角。IEEE Transactions on Knowledge and Data Engineering 30, 7 (2018), 1366 1385 10 1109 2017 2781227 – 。https://doi.org/ 。/ TKDE. [23] 2020 。Julian Dreyer, Marten Fischer 和 Ralf Tönjes。。Hyperledger Fabric 2.0 区块链平台的性能分析。载于智能物联网系统云连续体服务研讨会论文集（CCIoT '20）。计算机协会，纽约，NY，美国，32–38。https://doi.org/10.1145/ 3417310.3431398[24] Frank Duchmann 和 Agnes Koschmider。2019年。使用流程挖掘验证智能合约。载于 ZEUS。CEUR研讨会论文集，第2339卷。13–16。[25] FabricSharp Git 代码仓库 2022年。https://github.com/ooibc88/FabricSharp。（2022年）。[在线；访问日期：4月13日-2022]。[26] Ghareeb Falazi、Vikas Khinchi、Uwe Breitenbücher 和 Frank Leymann。2019年。许可型区块链的事务属性。SICS Software-Intensive Cyber-Physical Systems (2019)。 https://doi.org/10.1007/s00450-019-00411-y [27] Christian Gorenflo、Stephen Lee、Lukasz Golab 和 Srinivasan Keshav。2019年。FastFabric：将 Hyperledger Fabric 扩展至每秒20,000笔交易。载于2019年IEEE区块链与加密货币国际会议（ICBC）。455–463。https://doi.org/ 10 1109 2019 8751452 。/ BLOC。。Gideon Greenspan 等。[28] 2015 。Multichain 私有区块链白皮书。网址：http://www. multichain. com/download/MultiChain-White-Paper. pdf (2015), 57–60。[29] Christian W Günther 和 Anne Rozinat。2012年。Disco：发现你的流程。BPM（演示）940 (2012), 40–44。[30] Christian W Günther 和 Wil MP Van Der Aalst。2007年。模糊挖掘——基于多视角指标的自适应流程简化。载于业务流程管理国际会议。Springer，328–343。[31] Theo Härder。1984年。关于乐观并发控制方案的观察。Information Systems 9, 2 (1984), 111–120。 https://doi.org/10.1016/0306-4379(84) 90020-6[32] Richard Hobeck、Christopher Klinkmüller、Hmn Dilum Bandara、Ingo Weber 和 Wil Van der Aalst。2021年。区块链数据上的流程挖掘：Augur案例研究。技术报告。EasyChair。[33] Christian Hugo Hoffmann。2021年。区块链用例再探：面向零售银行和金融普惠的小额贷款解决方案。Journal of Systems Science and Information 9, 1 2021 1 15 10 21078 2021 001 15 ( ), – 。 [34] 2008 https://doi.org/doi: 。/JSSI- - - Richard Hull。。以制品为中心的业务流程模型：研究成果与挑战简述。载于《迈向有意义的互联网系统：OTM 2008》，Robert Meersman 和 Zahir Tari（编）。Springer Berlin Heidelberg，柏林，海德堡，1152–1163。[35] Hyperledger Caliper 2020年。 https://hyperledger.github.io/caliper/。（2020年）。[在线；访问日期：4月7日-2022]。[36] Zsolt István、Alessandro Sorniotti 和 Marko Vukolić。2018年。Streamchain：区块链需要区块吗？载于第二届分布式账本可扩展与弹性基础设施研讨会论文集。1–6。[37] Haris Javaid、Chengchen Hu 和 Gordon Brebner。2019年。优化 Hyperledger Fabric 的验证阶段。载于2019年IEEE第27届建模、分析与计算机及电信系统仿真国际研讨会（MASCOTS）。 269 275 10 1109 2019 00038 – 。 [38] https://doi.org/ 。/MASCOTS. 。Sadhana J. Kamatkar、Ajit Kamble、Amelec Viloria、Lissette Hernández-Fernandez 和 Ernesto García Cali。2018年。数据库性能调优与查询优化。载于《数据挖掘与大数据》，Ying Tan、Yuhui Shi 和 Qirong Tang（编）。Springer International Publishing，Cham，3–11。[39] Christopher Klinkmüller、Alexander Ponomarev、An Binh Tran、Ingo Weber 和 Wil van der Aalst。2019年。区块链流程挖掘：从区块链应用中提取流程挖掘数据。载于业务流程管理国际会议。Springer，71–86。[40] Olga Labazova、Erol Kazan、Tobias Dehling、Tuure Tuunanen 和 Ali Sunyaev。2021年。区块链系统与应用的管理：区块链配置流程模型。arXiv预印本 arXiv:2105.02118 (2021)。[41] Mengting Liu、F. Richard Yu、Yinglei Teng、Victor C. M. Leung 和 Mei Song。2019年。区块链赋能的工业物联网（IIoT）系统性能优化：深度强化学习方法。IEEE Transactions on Industrial Informatics 15, 6 (2019), 3559–3570。https://doi.org/10.1109/TII.2019. 2897805[42] Jiaheng Lu、Yuxing Chen、Herodotos Herodotou 和 Shivnath Babu。2019年。加速分析：数据库与大数据系统的自动参数调优。Proc. VLDB Endow. 12, 12 (2019年8月)。https://doi.org/10.14778/ 3352063.3352112[43] Orlenys López-Pintado、Luciano García-Bañuelos、Marlon Dumas、Ingo Weber 和 Alex Ponomarev。2018年。CATERPILLAR：业务流程执行引擎 on the Eth ereum Blockchain。2018 h ttps: //doi.org/10.48550/ARXIV. . 1808 03517

[44] Heidy M. Marin-Castro 和 Edgar Tello-Leal。2021 年。流程挖掘的事件日志预处理：综述。Applied Sciences 11，22（2021）。https://doi.org/10.3390/ app112210856

[45] Dennis McLeod 和 John Miles Smith。1980 年。数据库抽象。载于《1980 年数据抽象、数据库与概念建模研讨会论文集》。计算机协会，美国纽约州纽约市。https://doi.org/ 10 1145 800227 806871 。/ 。Du Mingxiao、Ma Xiaofeng、Zhang Zhe、Wang Xiangwei 和 Chen Qijun。2017 年。区块链共识算法综述。载于《2017 IEEE 系统、人类与控制论国际会议》（SMC）。2567–2572。https: //doi.org/ 10 1109 2017 8123011 。/SMC。 。Bhabendu Kumar Mohanta、Soumyashree S Panda 和 Debasish Jena。2018 年。区块链技术中的智能合约与用例概述。载于《2018 年第九届计算、通信与网络技术国际会议》（ICCCNT）。 1 4 10 1109 2018 8494045 – 。https://doi.org/ 。/ICCCNT。 。JP Morgan。 。Quorum 白皮书。纽约：JP Morgan Chase（）。[49] Roman Mühlberger、Stefan Bachhofner、Claudio Di Ciccio、Luciano García-Bañuelos 和 Orlenys López-Pintado。2019 年。从区块链上存储的数据中提取用于流程挖掘的事件日志。载于业务流程管理国际会议。Springer，690–703。[50] Satoshi Nakamoto。2008 年。比特币：一种点对点电子现金系统。去中心化商业评论（2008），21260。[51] Q. Nasir、Ilham A. Qasse、M. Talib 和 A. B. Nassif。2018 年。Hyperledger Fabric 平台的性能分析。Secur. Commun. Networks 2018（2018），3976093:1–3976093:14。[52] Pezhman Nasirifard、Ruben Mayer 和 Hans-Arno Jacobsen。2019 年。FabricCRDT：一种面向许可型区块链的无冲突复制数据类型方法。载于《第 20 届 Middleware 国际会议论文集》（Middleware '19）。计算机协会，美国纽约州纽约市，110—122。https: //doi.org/10.1145/3361525.3361540[53] Senthil Nathan、Chander Govindarajan、Adarsh Saraf、Manish Sethi 和 Praveen Jayachandran。2019 年。区块链遇上数据库：区块链关系数据库的设计与实现。Proc. VLDB Endow. 12，11（2019 年 7 月）。https: //doi.org/ 10 14778 3342263 3342632 。/ 。Keerthi Nelaturu、Sidi Mohamed Beillahi、Fan Long 和 Andreas Veneris。2021 年。智能合约细化以实现 Gas优化。载于《2021 年第 3 届区块链研究、创新网络与服务应用会议》（BRAINS）。 229 236 10 1109 52497 2021 9569819 – 。https://doi.org/ 。/ BRAINS 。 。Diego Ongaro 和 John Ousterhout。 。寻找一种易于理解的共识算法。载于《2014 年 USENIX 会议暨 USENIX 年度技术会议论文集》（USENIX ATC'14）。USENIX 协会，美国加利福尼亚州伯克利， 305 320 =2643634 2643666 – 。http://dl.acm.org/citation.cfm?id 。Orlenys Pintado、Luciano García-Bañuelos、Marlon Dumas、Ingo Weber 和 Alexander Ponomarev。2019 年。Caterpillar：以太坊区块链上的业务流程执行引擎。Software: Practice and Experience（2019 年 5 月）。https: //doi.org/ 10 1002 2702 。/spe。荷兰金融机构贷款申请流程中的流程挖掘 2017。https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2017: bpi2017_ winner_professional.pdf。（2017）。 [在线；访问日期：2022] 年 4 月 4 日。[58] Yuncheng Qiao、Chaoqun Ma、Qiujun Lan 和 Zhongding Zhou。2019/12。基于区块链技术的库存融资模型。载于《第四届经济与商业管理国际会议论文集》（FEBM *2019* 337 342 10 2991 19 2019 7 ）。Atlantis Press，– 。https://doi.org/ 。/febm- 。 。Mayank Raikwar、Danilo Gligoroski 和 Goran Velinov。 。数据库与区块链发展趋势。载于《2020 年第七届软件定义系统国际会议》（SDS）。177–182。https://doi.org/10.1109/SDS49854.2020。9143893[60] Aravind Ramachandran 和 Dr. Murat Kantarcioglu。2017 年。使用区块链和智能合约进行安全数据溯源管理。arXiv（2017）。[61] Michel Rauchs、Apolline Blandin、Keith Bear 和 Stephen B McKeon。2019 年。第二次全球企业区块链基准研究。可在 SSRN 3461765 查阅（2019）。[62] Olivier Rikken、Marijn Janssen 和 Zenlin Kwee。2019 年。区块链与去中心化自治组织的治理挑战。Information Polity 24 11 2019 1 21 10 3233 190154 （），– 。https://doi.org/ 。/IP- Henrique Rocha 和 Stéphane Ducasse。 。面向区块链的软件建模初步步骤。载于《2018 IEEE/ACM 区块链软件工程新兴趋势第一届国际研讨会》（WETSEB）。52–57。[64] Pingcheng Ruan、Tien Tuan Anh Dinh、Dumitrel Loghin、Meihui Zhang、Gang Chen、Qian Lin 和 Beng Chin Ooi。2021 年。区块链与分布式数据库之比较：二分法与融合。计算机协会，美国纽约州纽约市。https://doi.org/ 10 1145 3448016 3452789 。/ 。Pingcheng Ruan、Dumitrel Loghin、Quang-Trung Ta、Meihui Zhang、Gang Chen 和 Beng Chin Ooi。2020 年。执行-排序-验证区块链的事务视角。载于《2020 ACM SIGMOD 数据管理国际会议论文集》（SIGMOD '20）。计算机协会，美国纽约州纽约市，543–557。https://doi.org/10.1145/3318464.3389693

[66] Gary Shapiro、Christopher Natoli 和 Vincent Gramoli。2020年。拜占庭容错区块链的性能。载于*2020 IEEE*第*19*届网络计算与应用国际研讨会（*NCA*）。1–8。https://doi.org/10.1109/ NCA51143.2020.9306742[67] Ankur Sharma、Felix Martin Schuhknecht、Divya Agrawal 和 Jens Dittrich。2019年。模糊区块链与数据库系统之间的界限：Hyperledger Fabric案例。载于*2019*年数据管理国际会议论文集（*SIGMOD '19*）。ACM，纽约，纽约州，美国，105–122。https: //doi.org/ 10 1145 3299869 3319883 . / . Parth Thakkar、Senthil Nathan 和 Balaji Viswanathan。2018年。Hyperledger Fabric区块链平台的性能基准测试与优化。载于*2018 IEEE*第*26*届计算机与电信系统建模、分析与仿真国际研讨会（*MASCOTS*）。264–276。https://doi.org/ 10.1109/MASCOTS.2018.00034[69] An Binh Tran、Qinghua Lu 和 Ingo Weber。2018年。Lorikeet：用于基于区块链的业务流程执行和资产管理的模型驱动工程工具。载于*BPM*（学位论文/演示/行业）（*CEUR*研讨会论文集），第2196卷。CEUR-WS.org。[70] 交易流程 2022年。https://hyperledger-fabric.readthedocs.io/en/release- 2.2/txflow.html。（2022年）。 [在线；访问日期：2022]年4月7日。[71] 更新通道配置 2022年。https://hyperledger-fabric.readthedocs. io/en/release-2.2/config_update.html。（2022年）。 [在线；访问日期：2022]年4月7日。[72] 升级智能合约 2022年。https://hyperledger-fabric.readthedocs.io/en/ release-2.2/deploy_chaincode.html#upgrading-a-smart-contract。（2022年）。 [在线；访问日期：2022]年7月18日。[73] Gary Valentin、Michael Zuliani、Daniel C. Zilio、Guy Lohman 和 Alan Skelley。2000年。DB2顾问：一种足够智能、能够推荐自身索引的优化器。载于第*16*届数据工程国际会议论文集。https://doi. org/ 10 1109 2000 839397 . /I CDE. . Wil van der Aalst。.流程感知信息系统：从流程挖掘中汲取的经验。*T. Petri*网与其他并发模型 2（ 01 2009 1 26 10 1007 978 3 642 00899 3_1），–。https://doi.org/。/ - - - - Wil Van Der Aalst。.流程挖掘。*ACM*通讯 55, 8（2012年），76–83。[76] Wil van der Aalst、T. Weijters 和 L. Maruster。2004年。工作流挖掘：从事件日志中发现流程模型。*IEEE Transactions on Knowledge and Data Engineering* 16, 9 2004 1128 1142 10 1109 2004 47 （），–。https://doi.org/。/TKDE. . Boudewijn F van Dongen。.（）。https://doi.org/。/。v2[78] Boudewijn F Van Dongen、Ana Karla A de Medeiros、HMW Verbeek、AJMM Weijters 和 Wil MP van Der Aalst。2005年。ProM框架：流程挖掘工具支持的新时代。载于*Petri*网应用与理论国际会议。Springer，444–454。[79] A.J.M.M. Weijters、Wil M.P. Aalst, van der 和 A.K. Alves De Medeiros。2006年。使用*HeuristicsMiner*算法进行流程挖掘。埃因霍温理工大学。[80] Karl Wüst 和 Arthur Gervais。2018年。你需要区块链吗？载于*2018*年加密谷区块链技术会议（*CVCBT*）。45–54。https://doi.org/10. 1109/CVCBT.2018.00011[81] Xiaoqiong Xu、Gang Sun、Long Luo、Huilong Cao、Hongfang Yu 和 Athanasios V. Vasilakos。2021年。Hyperledger Fabric区块链网络的延迟性能建模与分析。《信息处理与管理》58, 1（2021年）， 102436 10 1016 2020 102436 。https://doi.org/。/ j.ipm. . Qi Yang、Xiao Zeng、Yu Zhang 和 Wei Hu。.基于智能合约的新贷款系统（*BSCI '19*）。计算机协会，纽约，纽约州，美国， 121 126 10 1145 3327960 3332395 –。https://doi.org/。/。Dirk A Zetzsche、Douglas W Arner 和 Ross P Buckley。2020年。去中心化金融。《金融监管期刊》6, 2（2020年9月），172–203。https://doi.org/10.1093/jfr/fjaa010 arXiv:https://academic.oup.com/jfr/article- pdf/6/2/172/37064506/fjaa010.pdf