

分类号: 按中国图书分类法, 学位办网上可查

单位代码: 10335

密 级: 注明密级与保密期限

学 号: _____

浙 江 大 学

硕士学位论文



中文论文题目: 基于区块链的物联网时序数据
高效可信存储技术研究

英文论文题目: Research on Efficient and Trusted Storage
Technology for Time Series Data in the
Internet of Things Based on Blockchain

申请人姓名: 滕依筱

指导教师: 董玮

合作导师: _____

学科(专业): 计算机技术

研究方向: 区块链

所在学院: 计算机科学与技术学院

论文递交日期 2024-12-25

基于区块链的物联网时序数据

高效可信存储技术研究



论文作者签名: _____

指导教师签名: _____

论文评阅人 1: _____ 姓名

评阅人 2: _____ 姓名

评阅人 3: _____ 姓名

评阅人 4: _____ 姓名

评阅人 5: _____ 姓名

答辩委员会主席: _____

委员 1: _____

委员 2: _____

委员 3: _____

委员 4: _____

委员 5: _____

答辩日期 _____

**Research on Efficient and Trusted Storage
Technology for Time Series Data in the
Internet of Things Based on Blockchain**



Author's signature: _____

Supervisor's signature: _____

External reviewers: _____

Name _____

Name _____

Name _____

Name _____

Name _____

Examining Committee Chairperson:

Examining Committee Members:

Date of oral defence: _____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名:

签字日期:

年 月 日

学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 浙江大学 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后适用本授权书)

学位论文作者签名:

导师签名:

签字日期:

年 月 日

签字日期:

年 月 日

致谢

感谢导师对本人的精心指导。他的言传身教将使我终生受益。

感谢 EmNets 的各位，他们的讨论和帮助使我受益匪浅。

感谢我的家人，他们的支持和鼓励是我最大的动力。

感谢我的对象，他的陪伴和理解使我度过了这段难忘的时光。

感谢我自己。

摘要

与传统的集中式存储相比，基于区块链的分布式存储系统提供了更高的安全性、透明度和更低的成本，使其成为点对点协作的理想选择。然而，它们较低的事务处理速度和较高的计算要求限制了它们在物联网（IoT）等高密度数据场景中的部署。

为了解决这个问题，我们提出了 **TimeChain**，这是一种用于物联网时序数据的高效链下区块链存储系统。**TimeChain** 对离散时序数据进行批处理，仅在链上存储每个批的哈希值，同时保持链下的完整数据。这大大降低了区块链上的存储开销和 37.4 倍的存储延迟。

为了减少范围查询中的额外传输延迟，**TimeChain** 采用了自适应打包机制。我们通过将数据和历史协同查询分别表示为图顶点和边权重，将批处理问题转化为图划分问题。为了减少数据完整性验证中的传输大小，一种基于局部敏感哈希（LSH）的数据完整性校验机制，通过仅传输非冗余部分来最大限度地减少完整性检查所需的数据。**TimeChain** 还集成了基于共识协议的节点选择机制，通过结合节点选择和共识过程来降低开销。

我们基于开源组件（如 Hyperledger Fabric 和 IPFS）实现了 **TimeChain**，并基于此评估 **TimeChain** 的性能。结果表明，与现有的基于区块链的存储系统相比，**TimeChain** 平均减少了 64.6% 的查询延迟和 35.3% 的存储延迟。

关键词：物联网时序数据、区块链、数据库

Abstract

Blockchain-based distributed storage systems offer enhanced security, transparency, and lower costs compared to traditional centralized storage, making them ideal for peer-to-peer collaboration. However, their lower transaction processing speed and higher computational requirements demands restrict their deployment in high-density data scenarios such as the Internet of Things (IoT).

To address this, we propose TimeChain, an efficient off-chain blockchain storage system for IoT time series data. TimeChain batches discrete time series data, storing only the hash value of each batch on-chain while keeping the complete data off-chain. This significantly reduces storage overhead on the blockchain and storage latency by 37.4 times.

In order to reduce the additional transmission latency in range queries, TimeChain employs an adaptive packaging mechanism. We convert the batching problem to a graph partitioning problem by representing data and historical co-query as graph vertices and edge weights respectively. To reduce the size of the transmission size in data integrity verification, a Locality-Sensitive Hashing (LSH)-based data integrity verification mechanism, which minimizes the data required for integrity checks by transmitting only non-redundant parts. TimeChain also integrates a node selection mechanism based on consensus protocol, which reduces the overhead by combining node selection and consensus processes.

We implement TimeChain based on top of production-ready open-source components such as Hyperledger Fabric and IPFS, and evaluate the performance of TimeChain. The result shows that compared to existing blockchain-based storage systems, TimeChain reduces 64.6% query latency and 35.3% storage latency on average.

Key words: IoT Time Series Data, Blockchain, Database

缩略词表

| 英文缩写 | 英文全称 | 中文全称 |
|-------|---|-------------|
| IoT | Internet of Things | 物联网 |
| ZB | ZettaBytes | 泽字节 |
| LSH | Locality Sensitive Hashing | 局部敏感哈希 |
| CRUSH | Controlled Replication Under Scalable Hashing | 可扩展哈希下的受控副本 |
| PG | Placement Group | 放置组 |
| MST | Merkle Semantic Trie | 默克尔语义树 |
| AMT | Authenticated Multipoint Evaluation Tree | 认证多点评估树 |
| IPFS | InterPlanetary File System | 星际文件系统 |
| POR | Proofs of Retrievability | 可检索性 |
| UWG | Undirected Weighted Graph | 无向加权图 |
| PC | Personal Computer | 个人电脑 |

目录

| | |
|---------------------------|------|
| 致谢 | I |
| 摘要 | III |
| Abstract | V |
| 缩略词表 | VII |
| 目录 | IX |
| 图目录 | XIII |
| 表目录 | XV |
| 1 绪论 | 1 |
| 1.1 研究背景及意义 | 1 |
| 1.2 论文主要工作 | 2 |
| 1.3 全文结构 | 3 |
| 1.4 本章小结 | 3 |
| 2 相关工作 | 5 |
| 2.1 面向物联网的分布式存储系统 | 5 |
| 2.2 基于区块链的存储系统 | 6 |
| 2.2.1 面向交易数据的链上存储 | 6 |
| 2.2.2 面向文件数据的链下存储 | 8 |
| 2.3 本章小结 | 9 |
| 3 基于区块链存储系统的测量与研究 | 11 |
| 3.1 基于区块链存储系统的架构 | 11 |
| 3.2 基于区块链存储系统的测量 | 13 |
| 3.3 性能瓶颈的根本原因 | 14 |
| 3.4 高效的链下存储架构 | 15 |
| 3.5 本章小结 | 16 |
| 4 面向链下存储的自适应聚合机制 | 19 |
| 4.1 基于历史查询的自适应图构建过程 | 19 |
| 4.2 基于谱聚类算法的封装机制 | 20 |

| | | |
|-------|----------------------|----|
| 4.3 | 本章小结 | 21 |
| 5 | 基于共识协议的节点选择机制 | 23 |
| 5.1 | 结合共识的节点选择过程 | 23 |
| 5.2 | 共识过程的安全性分析 | 24 |
| 5.3 | 本章小结 | 26 |
| 6 | 基于 LSH 树的验证机制 | 27 |
| 6.1 | LSH 树结构与工作流程 | 27 |
| 6.2 | LSH 树的尾部合并机制 | 28 |
| 6.3 | 本章小结 | 29 |
| 7 | 实验结果与分析 | 31 |
| 7.1 | 实验设置 | 31 |
| 7.1.1 | 基线 | 31 |
| 7.1.2 | 数据集和工作负载 | 32 |
| 7.2 | 总体性能 | 33 |
| 7.2.1 | 存储延迟 | 33 |
| 7.2.2 | 查询延迟 | 33 |
| 7.2.3 | 查询延迟分解 | 34 |
| 7.2.4 | 支持的最大存储设备数量 | 35 |
| 7.3 | 性能影响因素评估 | 36 |
| 7.3.1 | 不同查询大小下的查询延迟 | 36 |
| 7.3.2 | 不同存储网络规模下的存储延迟 | 36 |
| 7.4 | 消融实验 | 37 |
| 7.4.1 | 聚类算法 | 37 |
| 7.4.2 | 节点选择 | 38 |
| 7.4.3 | LSH 树 | 39 |
| 7.5 | 本章小结 | 39 |
| 8 | 总结与展望 | 41 |
| 8.1 | 总结研究 | 41 |
| 8.2 | 未来工作 | 41 |

| | |
|------------|----|
| 参考文献 | 43 |
| 作者简历 | 45 |

图目录

| | | |
|-------|------------------------|----|
| 图 2.1 | Ceph 存储流程 | 6 |
| 图 2.2 | SEBDB 数据结构 | 7 |
| 图 3.1 | 基于区块链的存储系统的工作流 | 12 |
| 图 3.2 | 区块链存储系统的性能 | 13 |
| 图 3.3 | 性能低下的根本原因 | 14 |
| 图 3.4 | TimeChain 架构 | 15 |
| 图 5.1 | 基于共识的节点选择机制工作流 | 24 |
| 图 6.1 | LSH 树结构 | 28 |
| 图 6.2 | 非满二叉 LSH 树 | 28 |
| 图 6.3 | 尾部合并下的非满二叉 LSH 树 | 29 |
| 图 7.1 | 存储延迟 | 33 |
| 图 7.2 | 查询延迟 | 33 |
| 图 7.3 | 查询延迟分解 | 34 |
| 图 7.4 | 最大支持存储设备数 | 34 |
| 图 7.5 | 不同查询大小下的查询延迟 | 35 |
| 图 7.6 | 不同存储网络下的存储延迟 | 35 |
| 图 7.7 | 聚类算法消融实验 | 37 |
| 图 7.8 | 节点选择消融实验 | 37 |
| 图 7.9 | LSH 树消融实验 | 38 |

表目录

1 绪论

1.1 研究背景及意义

近年来,物联网(Internet of Things, IoT)发展迅速,促进了农业、制造业、医疗保健、零售业和公用事业等众多应用的发展。根据 Gartner 的分析^[1],数十亿部署的物联网设备将产生泽字节(ZettaBytes, ZB)的数据。随着 IoT 的发展趋势,可以预见,未来将有越来越多的数据集成到网络中。对于如此大规模的数据,使用去中心化服务器(例如 AWS IoT、阿里云等)来管理它会遇到单点故障等问题。尽管分布式数据库可以避免单点故障,但由于数据安全性和不可篡改性较弱,在需要高度透明的情况下,它们很容易受到数据篡改攻击,例如物联网数据共享^[2]。

区块链以可追溯性和不变性为特征,可以解决集中式存储的单点故障问题和分布式数据库的恶意篡改问题。它将数据存储在不中心化的账本中,并使用共识协议来解决平等节点之间的冲突,这种方式增强了安全性,提高了透明度,降低了成本,支持点对点协作。尽管它具有巨大的潜力,但其较低的交易速度和较高的计算要求使其仅广泛应用于价值巨大但密度很低的领域,如金融服务、供应链管理等。

目前,有许多研究人员致力于提高基于区块链的存储系统的性能。根据数据的存储位置,这些工作可以分为两类,即链上存储和链下存储。

对于链上存储,数据作为存储在区块链上的交易记录的一部分,用户通过区块链中的索引,即默克尔树(Merkle Tree, MT)获取这些数据。现有工作通过提供用户友好的查询语言^[3-5]提高了系统可用性,并通过改进索引方案^[6-7]、区块链存储分片^[8-10]等方式提高系统吞吐量。然而,在链上存储物联网数据的开销非常高。对于大量快速生成的物联网数据,将其连续存储在区块链上需要实现共识和分类账复制的过程,这可能会导致巨大的存储压力和通信开销。因此,物联网数据存储的一种更实用的方法是利用链下存储解决方案。

对于链下存储,数据存储在区块链之外,区块链只存储必要的元数据或对数据的引用,如哈希或加密指针。链下存储提供了比链上解决方案更大的可扩展性和更低的成本,因此业界和学术界对链下存储非常有兴趣,例如 Storj^[11]、BigchainDB^[12]、Sia^[13]等。

然而，现有的工作大多是为存储大文件而设计的。对于单位粒度较小的物联网数据，考虑到物联网的海量大小，在区块链中存储每个数据项的哈希值将产生难以接受的开销。此外，物联网应用场景通常需要存储系统支持高效查询（例如聚合查询），而现有的基于文件的存储系统不支持这一点。

1.2 论文主要工作

为了对海量物联网数据进行可信、高效的存储，本文提出了一种基于区块链的物联网时序数据高效存储技术，TimeChain。受链下存储方案的启发，TimeChain 对离散时间序列数据进行批处理，仅将每批数据的哈希值存储在链上，并将完整的原始数据保存在链外。这种批处理存储方法大大减少了数据开销。我们对该方案的性能进行了测量。根据我们的测量，与单个数据存储相比，该方案的存储延迟平均减少了 37.4 倍。这种存储性能使基于区块链的时间序列数据存储成为可能。

然而，不可否认的是，这样的批处理方案会影响查询性能。具体来说，当用户执行范围查询时，低效的聚合方法会导致在多个传输节点上获取数据，从而导致额外的传输延迟。此外，由于区块链上只存储了批处理单元的哈希值，因此存储系统必须传输额外的信息（例如默克尔树的哈希路径），以支持数据拥有者执行数据完整性验证。这进一步增加了查询开销。

为了减少范围查询过程中跨越的存储节点数量，我们提出了一种新的自适应打包机制。通过从原始数据构建无向加权图，我们将批处理问题转化为图划分问题。我们使用谱聚类算法通过将频繁查询的数据分组在一起来解决分区问题，以减少聚合查询期间的节点访问次数。

为了在确保系统安全的同时进一步减少查询延迟，我们提出了一种基于共识的存储节点选择机制。在选择节点时，我们基于之前的测量，受已有方案的启发，同时考虑存储节点信誉、剩余存储空间和传输距离。为了在基于区块链的存储系统中快速做出节点选择决策，我们将共识过程与节点相结合，以减少传播延迟。

为了减少验证过程中的传输开销，我们提出了一种基于局部敏感哈希（Locality Sensitive Hashing, LSH）树的数据完整性验证机制。该机制基于物联网场景下相邻时间序列数据点之间的相似性，通过仅传输证明的非冗余部分显著减少了完整性验证所需的

数据。

我们基于开源组件（如 Hyperledger Fabric 和 IPFS）实现了 TimeChain，并基于此评估 TimeChain 的性能。结果表明，与现有的基于区块链的存储系统相比，TimeChain 平均减少了 64.6% 的查询延迟和 35.3% 的存储延迟。

1.3 全文结构

本文围绕基于区块链的物联网时序数据高效可信存储技术研究展开，其余部分组织如下：第二章介绍了相关工作，包括分布式存储系统、基于区块链的存储系统和基于区块链的文件系统；第三章介绍了目前基于区块链的物联网存储系统面临的需求和挑战，并提出了一种高效的物联网数据链下存储框架 TimeChain；第四章介绍了 TimeChain 的自适应聚合机制；第五章介绍了 TimeChain 的存储节点选择机制；第六章介绍了 TimeChain 的数据验证机制；第七章介绍了 TimeChain 的性能评估；第八章总结了全文，并展望未来工作。

1.4 本章小结

本章首先对本文的研究背景进行了介绍，阐明了边缘计算系统定制化的原因，介绍了系统构建的四个阶段并对现有问题进行了分析，从而引出本文选题的重要性。在此基础上，对本文的主要研究工作进行了总结。

2 相关工作

2.1 面向物联网的分布式存储系统

近年来,随着数据规模的急剧增长和应用需求的多样化,集中式数据存储解决方案的固有漏洞逐渐显露出来,尤其是对于易受单点故障影响的局限性,引起了人们对分布式数据库的日益关注。在传统集中式数据存储方式中,数据通常集中存储在单一节点或少数几个节点上,这种架构容易导致单点故障问题。一旦出现故障,整个系统的稳定性和可用性将受到严重影响,甚至导致数据的不可靠或不可用。相比之下,分布式数据库将数据分散存储在多个节点或数据中心中,这使系统本身就具备了抗单点故障的能力。即使某个节点或数据中心发生故障,系统仍然可以继续运行并提供服务,从而保障了数据的可靠性和稳定性。因此,随着对数据安全性、可用性和性能要求的不断提高,分布式数据库作为一种能够克服集中式存储固有弱点的解决方案,逐渐成为了当今大数据时代的主流选择。

Apache Cassandra^[14]作为一个分布式结构化键值存储系统,借鉴了 Dynamo^[15]的分布式系统技术和 Google BigTable^[16]的数据模型优势,通过特殊的数据分发策略确保数据存储的高可用性和容错性。其灵活的数据模型和分布式特性使得它适用于需要横向扩展和高性能的应用场景。

Spanner^[17]是谷歌开发的全球分布式数据库。它通过 TrueTime API^[18]提供的时间戳服务实现全球性一致性和高可用性,并结合关系型数据库的 ACID 属性和分布式数据库的横向扩展,适用于需要高度一致性和水平扩展的工业应用场景。

Ceph^[19]作为一个开源的分布式存储系统,提供对象存储、块存储和文件系统存储功能。为了平衡各存储节点的压力,它提出了可扩展哈希下的受控副本 (Controlled Replication Under Scalable Hashing, CRUSH) 算法,用于数据的分布式存储和数据冗余备份。如图 2.1 所示,对于需要被存储的文件而言, Ceph 会首先将文件划分成多个独立的对象,然后 Ceph 将对象根据哈希分成若干放置组 (Placement Group, PG)。随后, CRUSH 算法结合地理位置信息将数据对象映射到存储集群中的存储节点,实现数据的均衡分布和高效访问。这种分布式、无中心化的数据分布方式使 Ceph 系统具有高度的可扩展性和

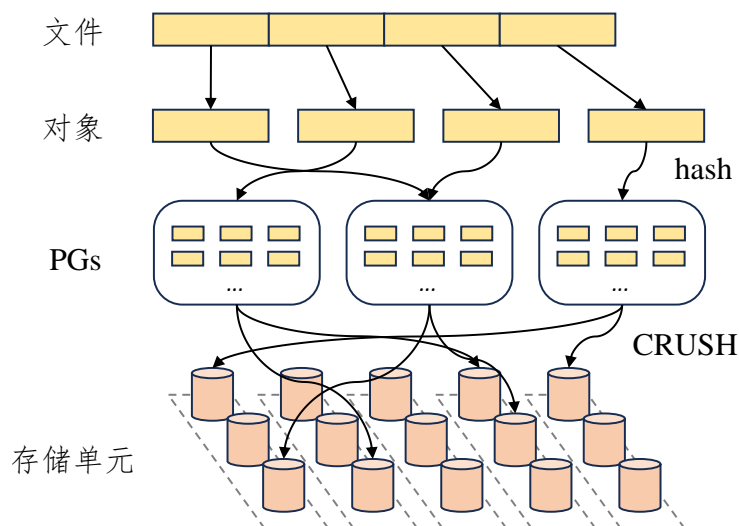


图 2.1 Ceph 存储流程

容错性，即使在节点故障或网络分区的情况下，系统仍能保持数据的可靠性和可用性。

CockroachDB^[20]在 Spanner 的基础上，通过设计结合地理信息的复制和自动恢复机制来实现高容错性。它将数据分布在不同的地理位置或数据中心，确保数据的冗余备份分布在不同的地理区域。这种地理信息复制策略使得即使某个地理区域发生故障，数据仍然可以从其他地理位置的备份节点进行恢复。

然而，这些分布式存储系统仍然存在着单点故障和高存储成本的问题。这是因为即使数据存储在不同位置，数据的管理仍然是中心化的。在这些存储系统中，公司有可能在数据拥有者不知情的情况下轻松篡改数据，直接危及数据的完整性和安全性。这些缺陷限制了这些系统在处理物联网数据激增和复杂协作需求方面的灵活性和效率，同时也使得数据的安全性和可靠性备受质疑。因此，解决这些问题对于推动分布式存储系统在未来应用中获得广泛应用至关重要。

2.2 基于区块链的存储系统

2.2.1 面向交易数据的链上存储

区块链技术的出现为分布式存储系统的安全性和可靠性提供了新的解决方案。区块链存储系统是一种基于区块链技术的分布式存储解决方案，其核心概念是将数据分布式存储在网络中的多个节点上，并使用区块链作为数据的验证和记录机制。在区块链存储系统中，数据被分割成块，并以链式连接的方式进行存储，每个数据块包含了前一个块

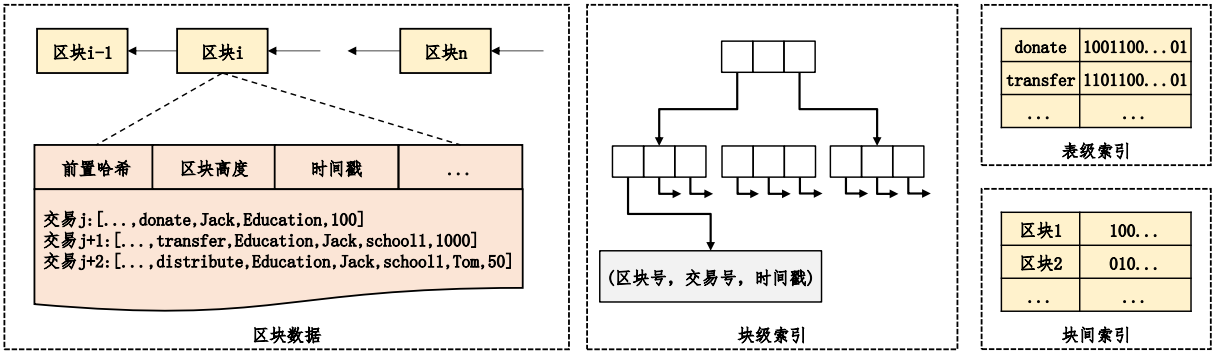


图 2.2 SEBDB 数据结构

的哈希值，从而构成了不可篡改的数据记录链。这种数据结构使得数据的完整性和安全性得到了极大的保障，任何尝试篡改数据的行为都会被系统所记录和拒绝。区块链存储系统的去中心化特性意味着数据的管理不再依赖于单一实体，而是由网络中的多个节点共同维护和验证。这种设计使得数据拥有者可以更加信任系统，因为数据的安全性和可靠性不再受限于单一实体的控制。目前，许多链上存储的研究都集中在链上数据存储的性能上，包括查询性能和存储负担。

部分工作关注于增强链上存储的查询支持。由于区块链的交易数据都是以日志的形式存储在区块链上，无法提供类似关系型数据库的高效查找。因此，一些工作提出了在区块链上建立索引的方法，以支持对区块链数据的高效查询。如图 2.2 所示，SEBDB^[3]使用 B+ 树建立了块级索引，将真实的数据组织成 B+ 树，以支持类 SQL 查询。SEBDB 也建立了表级索引和块间索引，便于快速定位区块位置。MSTDB^[21]则建立了基于语义的索引结构默克尔语义树（Merkle Semantic Trie, MST），以支持基于语义的多关键字查询，并对于 top-k 查询以及范围查询的验证方法做出优化。MSTDB 也利用索引压缩和查询预处理技术，进一步提高了存储和查询性能。

也有部分工作优化了区块上的索引和认证机制，以提高区块链状态的查询速度。LVMT^[6]采用了认证多点评估树（Authenticated Multipoint Evaluation Tree, AMT）在常数时间内更新完整性证明，并且采用多层设计来支持无限的键值对。LVMT 还通过存储版本号替代哈希的方式避免了昂贵的椭圆曲线乘法运算。COLE^[7]基于学习索引和列存储优化了区块链的存储索引结构。COLE 将数据按状态地址连续存储，通过学习索引模型减少存储索引开销；并针对区块链的存储磁盘特性引入了 LSM 树，以支持高效的读写性能。

部分工作关注链上存储的性能优化。FalconDB^[22]和 Chen 等人提出的仲裁模型^[2]都关注区块链节点之间数据交互的效率。为了减轻区块链账本数据存储的负担, Rapid-chain^[8]、SlimChain^[23]和 GriDB^[9]将账本的数据分配给其他分片存储, 以减轻存储压力。

在大规模数据生成迅速的物联网场景中, 上述解决方案存在两方面的挑战。

- 首先, 这些解决方案会给区块链带来额外的空间存储负担。由于物联网设备在短时间内生成的数据量庞大, 将这些数据全部存储在区块链上可能会导致链的存储需求急剧增加, 进而增加系统的复杂性和成本。
- 其次, 这些方案可能导致数据隐私泄露的风险增加, 因为在区块链上存储的数据通常是不可篡改和公开可见的, 这会导致物联网的数据泄露。

综上所述, 在数据密度低、生成速度快的物联网场景下直接使用传统的链上存储方案, 会产生较大的开销和风险。物联网场景需要更加注重数据存储效率和隐私保护, 因此需要针对这些特殊需求设计更为灵活和高效的数据管理方案。

2.2.2 面向文件数据的链下存储

基于区块链的文件系统因其较少的开销而备受广泛研究和关注。这些系统将文件本身存储在链下, 规避了原始数据直接存储在链上带来的巨大开销, 并通过在区块链上记录文件的哈希值和验证信息来确保文件的完整性和安全性。由于区块链的不可篡改性, 用户可以方便地验证文件的来源和完整性, 防止数据篡改和丢失。相较于传统的中心化存储系统, 基于区块链的文件系统通常不需要中心化的服务器和维护成本, 从而降低了整体的运营成本。

Filecoin^[24]是建立在星际文件系统 (InterPlanetary File System, IPFS) 的去中心化文件存储系统。IPFS 是一种点对点的分布式文件系统, 它通过内容寻址来存储和共享文件, 使得文件在没有单点故障的分布式网络上可用。通过激励机制鼓励用户提供存储服务, Filecoin 利用区块链技术确保了文件的完整性和可靠性, 同时为用户提供了一种去中心化的存储解决方案。在这个网络中, 用户可以通过提供存储空间来获取 Filecoin 代币奖励, 从而构建一个分布式的文件存储网络。这种模式不仅激励了更多的存储资源加入网络, 还提高了数据的持久性和可访问性, 因为文件被分散存储在全球各地的节点上, 减少了中心化存储的风险。

Storj^[11]和 Sia^[25]是两个典型的半去中心化的文件存储系统，它们为每个文件建立默克尔树以确保文件的完整性。这种方法通过分散存储文件块来提高可靠性和安全性，从而降低了数据丢失的风险。用户可以通过这些系统安全地存储和共享文件，同时保持数据的隐私和安全性。

FileDES^[26]专注于数据的加密存储，借助零知识证明等技术来保护数据的安全性。通过加密存储数据，FileDES 可以确保用户的数据在存储和传输过程中不会被泄露或篡改。这种方法为用户提供了更高级别的数据安全保障，尤其适用于对数据隐私和安全性要求较高的场景。

然而，尽管这些方法在确保文件完整性和安全性方面表现出色，它们并不适用于高效存储物联网数据的存储。这是因为物联网数据通常具有较低的价值密度，而将单个数据以文件形式存储在区块链文件系统中，将导致极高的成本。物联网数据的特点决定了需要更为轻量级和高效的数据存储方案，以更好地应对数据生成量大、频繁性高的特点，同时降低存储和处理成本。因此，对于物联网大规模数据存储的需求，寻找轻量、高效、可信的数据解决方案是一个挑战。

2.3 本章小结

在本章中，我们广泛探讨了面向物联网的分布式存储系统和基于区块链的存储系统的相关研究工作。我们分析了集中式数据存储解决方案的局限性，特别是它们在面对单点故障时的脆弱性，以及分布式数据库如何通过多个节点间分散存储数据来提高数据的可靠性和稳定性。我们详细讨论了 Apache Cassandra、Spanner、Ceph 和 CockroachDB 等分布式存储系统，它们通过不同的技术手段实现了数据的高可用性和容错性。然而，这些系统仍然存在单点故障的风险，并且由于数据管理的中心化，数据的完整性和安全性仍然受到威胁。

进一步地，我们探讨了基于区块链的存储系统，它们通过将数据分布式存储在多个节点上，并利用区块链作为数据验证和记录的机制，提供了一种新的数据存储解决方案。我们分析了链上存储和链下存储两种方法，以及它们在提高数据存储性能和确保数据完整性方面的研究进展。尽管这些基于区块链的存储系统在确保数据完整性和安全性方面具有优势，但它们在处理大规模、低价值密度的物联网数据时面临效率和成本的挑

战。

综上所述,现有的分布式存储系统和基于区块链的存储系统虽然在某些方面取得了进展,但它们在满足物联网数据存储需求方面仍存在不足。这些挑战包括处理大规模数据的效率、数据存储的成本以及数据隐私保护的需求。因此,开发一种轻量级、高效且可信的数据存储解决方案对于物联网领域来说是一个重要的研究方向。**TimeChain** 系统正是在这样的背景下应运而生,旨在结合区块链技术的优势,解决物联网时序数据存储中的关键问题。

3 基于区块链存储系统的测量与研究

在本章中，我们深入探讨了基于区块链的物联网时序数据存储系统的基础架构，并对该系统进行了全面的测量研究。该方案将数据分成几个聚合单元，将聚合单元的元数据记录上链，原始数据存储远端节点，以应对物联网数据的快速增长和对数据安全性的严格要求。通过对基本存储系统的性能测试，我们发现批处理存储能够显著降低存储延迟，但同时也带来了查询延迟的问题，特别是在执行范围查询等操作时。

我们进一步分析了导致查询性能不佳的根本原因，包括查询跨越多个聚合单元、存储节点选择不当以及庞大的数据完整性证明。这些发现为我们提供了针对性能瓶颈优化的机会。

为了解决这些问题，我们提出了 TimeChain 架构，这是一种新颖的、基于区块链的物联网时间序列数据存储系统。我们介绍了 TimeChain 系统的核心模块，包括数据批处理模块、存储节点选择模块和数据验证模块。具体来说，数据批处理模块通过构建自适应无向加权图和采用谱聚类算法来优化数据打包过程。存储节点选择模块则通过结合共识机制来确保节点选择的安全性和准确性。最后，数据验证模块通过引入基于 LSH 树的验证机制，有效减少了数据传输的大小，从而降低了网络传输延迟。这些模块共同作用，旨在减少查询延迟、提高数据存储的效率和安全性。

3.1 基于区块链存储系统的架构

为了提高基于区块链的分布式数据库的性能，我们提出了一种基本的链下存储系统，并对其进行了测量研究。如图 3.1 所示，一个基于区块链的基本分布式存储系统有四个主要角色，即**数据拥有者**、**网关**、**卫星**和**存储提供商**。

- **数据拥有者**：数据拥有者是数据的生成者和消费者，他们发起上传和查询数据的请求。
- **网关**：网关是数据拥有者与存储提供商之间的中间层，负责数据的上传和下载。
- **卫星**：与 Storj^[11]和 FileCoin^[24]类似，卫星是一个智能合约，负责协调数据拥有

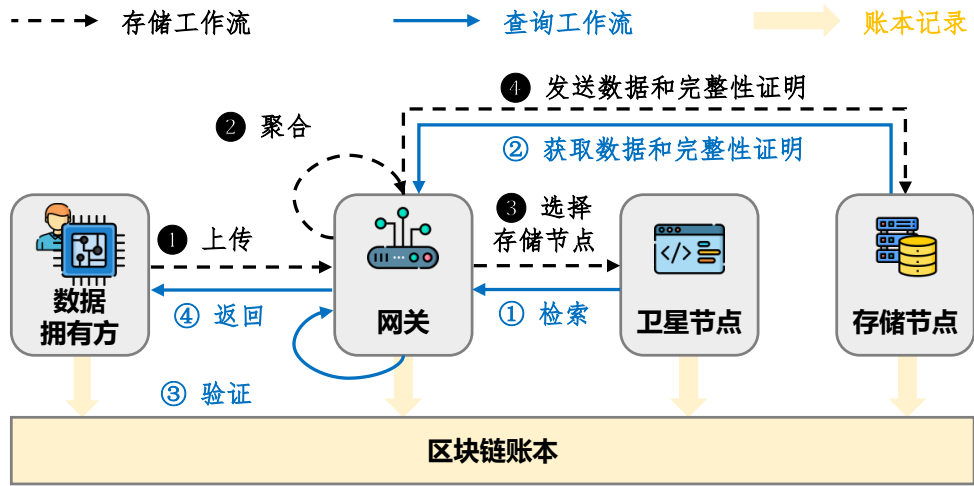


图 3.1 基于区块链的存储系统的工作流

者和存储提供商之间的交互。它们提供文件审计（检查）或可检索性（Proofs of Retrievability, POR）相关功能和对存储支付等操作的处理。

- **存储提供商**：存储提供商是存储数据的节点，通过提供存储服务获得奖励。为了使存储提供商能够通过灵活的查询快速向数据拥有者提供完整性证明，数据完整性证明也需要存储在存储提供商中，便于通过默克尔树的路径直接快速组装成完整性证明。存储提供商的服务信息，如剩余存储空间，将与交互记录一起记录在分布式账本中，以确保安全。

一般来说，数据存储和查询过程可以概括如下：

数据存储：

- ① 上传：数据拥有者通过网关接口上传数据。
- ② 聚合：网关接收到数据后，会对时间序列数据进行批处理。这一步骤涉及到将多个数据点聚合在一起，并为每批数据生成数据完整性证明，这对于确保数据在存储和后续查询过程中的完整性至关重要。
- ③ 选择存储节点：在聚合数据后，卫星节点介入，帮助网关识别和选择最佳的存储节点。这一步骤基于一系列选择标准，如节点的信誉、存储空间和地理位置等，以确保数据被存储在最可靠和最经济的节点上。
- ④ 发送数据和完整性证明：一旦确定了最佳存储节点，原始传感器数据及其完整性证明将被发送到这些节点。同时，数据批的元数据被记录在分布式账本中，利用区块链的不可篡改性确保数据的透明性和可追溯性。

数据查询：

- ① 检索：当数据拥有者需要访问他们的数据时，他们通过网关发起数据下载请求。网关随后与卫星节点交互，以检索存储提供商的位置信息。
- ② 获取数据和完整性证明：网关根据从卫星节点获得的位置信息，从存储提供商处获取所需的数据及其完整性证明。

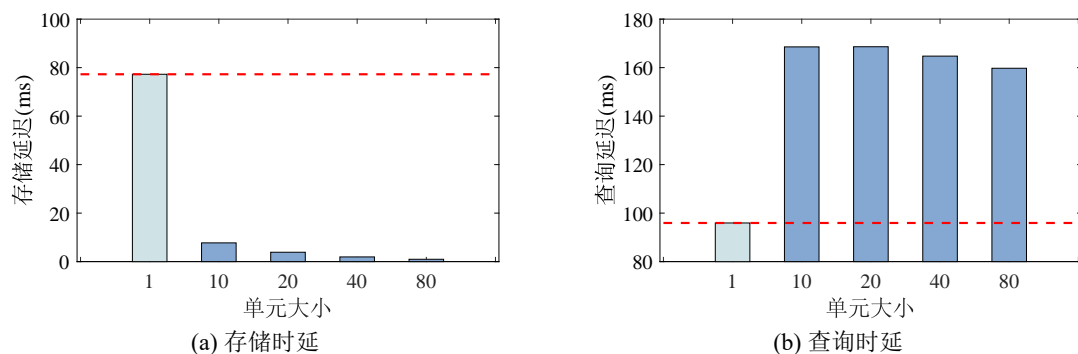


图 3.2 区块链存储系统的性能

完整性证明。这一步骤确保了数据在传输过程中的安全性和完整性。③验证：在获取数据后，网关将通过检查数据完整性证明来验证下载数据的完整性，确保数据未被篡改且准确无误。④返回：最后，经过验证的数据将通过网关安全地返回给数据拥有者，完成整个数据查询流程。

3.2 基于区块链存储系统的测量

在本节中，我们进行了一项初步测量，以评估基于区块链的基本存储系统的性能。我们基于 Hyperledger Fabric 实现了上述存储系统。该测试网络由 5 个节点组成，其中 1 个节点是网关，4 个节点是卫星。为了更准确地模拟真实世界中的存储网络环境，我们基于真实世界的数据存储网络模拟了一个包括 320 个存储节点的网络，其中包括 189 个真实存储提供商^[27]和 131 个模拟提供商（代表提供闲置存储的个人提供商）。为了确保实验结果的准确性，网关与这些模拟提供商之间的传输延迟是通过一个基于现有研究的线性回归模型^[28]进行预测的。

存储性能。在我们设置的系统中，数据拥有者每秒生成 20 个 56 字节的数据包，并在 20 秒内存储它们。存储性能结果如图 3.2a 所示。与单独存储每个数据相比，批存储将延迟减少到约 37.4 分之一，且随着聚合单元增大，存储延迟逐渐降低。这主要是因为较大的聚合单元规模减少了链上操作和数据传输的次数，从而降低了存储延迟。

查询性能。然后，我们测试范围查询的性能，如图 3.2b 所示。不幸的是，结果显示，相对于存储单个数据而言，批处理存储解决方案的查询性能相对较差。批量存储在不同聚合单元大小下的平均延迟为 165.4ms，无法满足许多物联网场景的需求。例如，自动驾驶应用程序的延迟小于 50ms^[29]，地震监测的延迟小于 100ms^[30]。

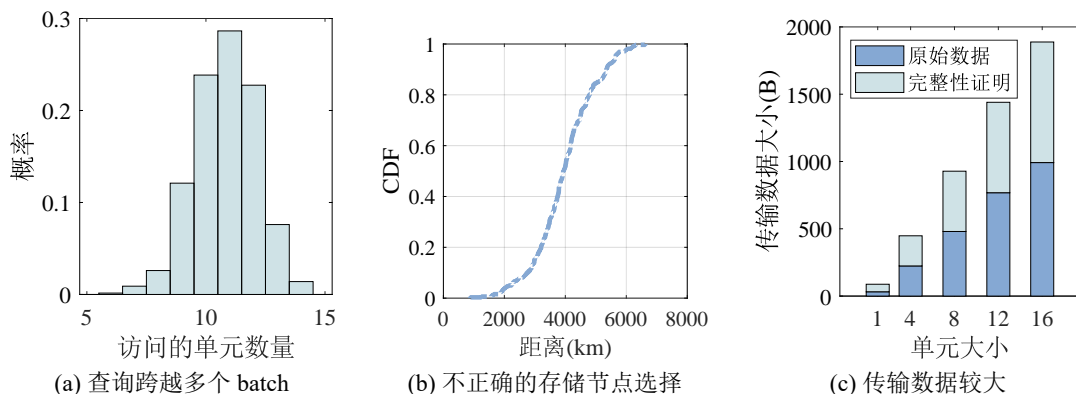


图 3.3 性能低下的根本原因

3.3 性能瓶颈的根本原因

为了找出该方案查询性能不佳的原因，我们进行了深入研究，并将原因总结为以下三点：

1. **范围查询跨越多个批处理单元。**时间序列数据的查询通常包含多个数据点，如范围查询、聚合查询、过滤查询等。如果批处理不当，单个查询可能会跨越多个批处理单元。我们使用现有的数据集 YCSB^[31]评估每个查询所跨越的批处理单元数量。如图 3.3a所示，超过 84.25% 的查询跨越了 10 个批处理单元。当这些批处理单元被存储在不同的节点上时，将引入额外的查询和传输延迟。

2. **存储节点选择不当。**在这个测量中，我们发现传输延迟占总查询延迟的很大一部分。如图 3.3b所示，世界各地存储节点的距离存在很大差异，导致节点之间的传输延迟存在很大差异。如果选择非常远的存储节点，传输延迟将会显著增加。此外，在整个存储网络中存在恶意节点的情况下，尽管最终选择的节点可能传输延迟较小，然而，由于恶意节点的存在，数据的完整性可能会受到威胁，从而无法提供完整的数据存储服务。

3. **完整性证明数据量大。**为了使存储提供商能够通过灵活的查询快速向数据所有者提供完整性证明，数据完整性证明也需要与存储提供商一起存储。因此，当存储提供商需要向数据所有者证明数据的完整性时，完整性证明也会被发送回数据所有者。图 3.3c显示了总传输数据量的各组成部分。从图中可以看出，数据完整性证明大小占接收数据的 48.8%，几乎是接收数据的一半。当网络繁忙时，大量的完整性证明数据传输会增加网络传输延迟。

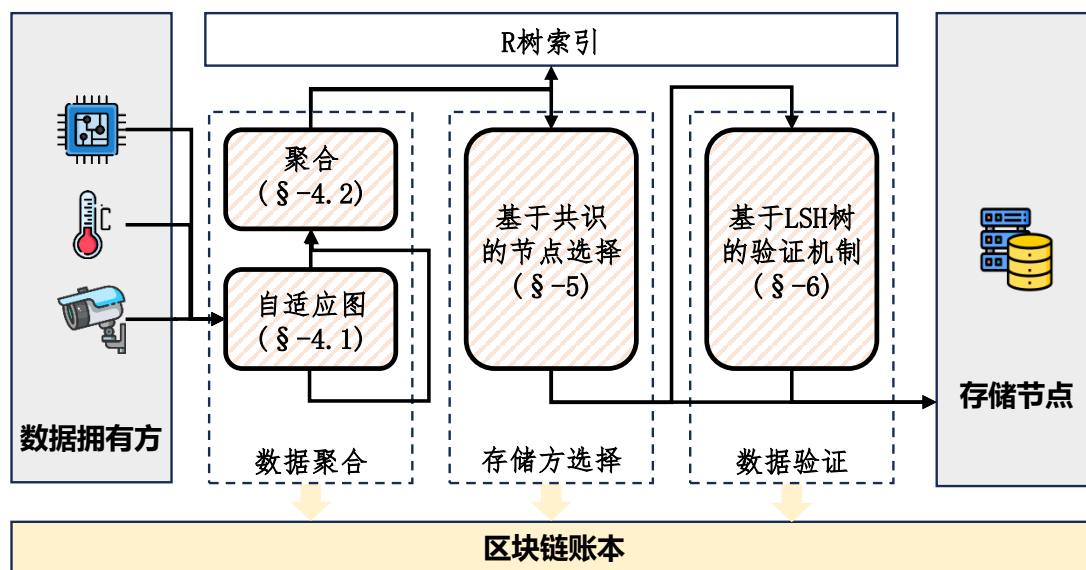


图 3.4 TimeChain 架构

3.4 高效的链下存储架构

为了提高区块链存储系统的查询性能，我们设计了一种基于区块链的新型物联网时间序列数据存储系统，TimeChain。图 3.4显示了 TimeChain 的架构。TimeChain 中的模块包括数据聚合、存储方选择、数据验证、R 树索引和区块链账本。TimeChain 建立在区块链平台上，所有操作都记录在分布式账本上，以方便数据的审计和追踪。TimeChain 的索引结构是 R 树，可以加速物联网场景中常用的时空聚合搜索。原始数据经过数据聚合模块，确定哪些数据被放在一个聚合单元中；经过存储方选择模块，确定聚合单元的存储位置。这些信息将被放在 R 树中，以便快速查询。接下来，我们介绍 TimeChain 的关键模块。

数据聚合模块：我们的测量研究表明，不正确的聚合方法会增加查询的跨单元数量，从而增加网络传输延迟。我们构建了一个自适应的无向加权图（Undirected Weighted Graph, UWG），以基于数据拥有者的历史查询准确捕获用户查询信息 (§-4.1)。基于我们构建的 UWG，我们将数据打包成批处理单元的问题转换为聚类问题^[32]，该问题根据用户的查询请求将所有原始数据划分为多个聚类。解决聚类问题的传统算法有很多，如 K-means^[33]、GMM^[34]等。然而，这种传统的聚类算法不适合分割物联网设备生成的数据。这是因为在 TimeChain 中，用户的查询没有遵循特定的特征，这可能会导致数据图的聚类形成复杂的形状，而不是常见的圆形。此外，传统的聚类算法需要将所有数据

划分为固定数量的集合，但并非所有集合都等于批大小，这将给索引查询带来额外的开销。因此，我们使用谱聚类算法来打包数据 (§-4.2)，这非常适合处理不规则和非固定数量的聚类。

存储方选择模块：存储节点的选择至关重要。正如我们之前在图 3.3b 中发现的那样，存储节点和网关之间的距离会影响数据访问延迟。此外，对于链下存储系统，存储空间不足的节点或恶意节点可能会导致数据丢失、篡改或服务中断，进而影响整个系统的安全性和稳定性。因此，我们根据距离和历史服务记录等信息对存储节点进行综合评估。存储节点选择过程的安全性也非常重要。Storj^[11]、CoopEdge^[35]和 PipeEdge^[36]通过一组固定的节点选择服务节点，并通过区块链确认决策。换句话说，他们集中决策，但仍然面临单点失败的威胁。然而，使用类似于 PBFT 的投票机制，节点选择过程通常需要多轮任务计算和消息广播。如果共识过程和节点选择过程完全解耦，系统安全将受到威胁。为了解决这个问题，我们将节点选择过程与共识相结合，提出了一种基于共识的节点选择机制 (§-5)。

数据验证模块：从之前的测量结果中我们可以发现，传输的数据中接近一半是数据完整性证明。数据完整性证明被组织为默克尔树的形式，它是由一系列哈希构建的。在默克尔树中，非叶子节点的哈希数几乎等于原始数据点的数量。由于物联网数据单元的大小大约等于哈希值，这意味着需要发送以验证数据的数据量几乎是原始数据的两倍。在这样的数据传输过程中，减小数据证明的大小是一个挑战。通过分析物联网数据，我们观察到物联网数据变化缓慢，在短时间内很少出现突然变化。对于这些相似的数据，LSH 算法可以从相似的原始数据中生成相似的哈希结果。LSH 算法确保类似的物联网数据即使在哈希后也保持相似。因此，我们提出了一种新的基于 LSH 树的验证机制 (§-6)，该机制采用 LSH 而不是传统默克尔树中使用的通用哈希。通过差分传输 LSH 哈希值，可以显著减小传输数据的大小。

3.5 本章小结

在本章中，受链下存储启发，我们先提出了一种基于区块链的物联网时序数据基本存储系统。对于这个基本系统，我们进行了一些性能测试，发现该系统确实可以降低存储延迟，但查询延迟较高。我们进一步分析了查询性能不佳的根本原因，并针对性能瓶

颈提出了高效链下存储系统 TimeChain 的架构。

4 面向链下存储的自适应聚合机制

我们的实验结果揭示了一个关键问题：不当的数据聚合策略会导致查询过程中需要跨越更多存储单元，进而增加了网络传输的延迟。在本章中，我们提出了一种自适应 UWG 的方法，它能够根据数据所有者的历史查询模式精确捕捉用户的查询行为。利用这个自适应 UWG，我们将数据分批的问题转化为一个聚类问题，即根据用户的查询需求将海量原始数据划分成多个有意义的聚类。

尽管存在多种传统的聚类算法，例如 K-means 和 GMM，它们不适合处理物联网设备产生的数据时。原因在于 TimeChain 系统中用户查询的模式多样化，不遵循统一特征，这可能导致聚类结果呈现出不规则的形状，而非典型的圆形。此外，这些传统算法通常要求将数据均匀分配到固定数量的聚类中，这与我们需求不符。因为不是所有的聚类都恰好匹配一个批次的大小，这无疑会增加索引查询的负担。

4.1 基于历史查询的自适应图构建过程

由于物联网的原始数据是孤立的点，我们在数据点之间创建加权边，表示被联合查询的概率。数据点 a 和 b 之间的边的权重 l_{ab} 表示为：

$$l_{ab} = \begin{cases} \sqrt{(id_a - id_b)^2 + (t_a - t_b)^2} & , k = 0 \\ \theta \cdot l_{ab} + (1 - \theta) \cdot x_{ab}^k & , k \geq 1 \end{cases} \quad (4-1)$$

其中 l_{ab} 被初始化为两个数据点设备 ID 和数据到达时间之间的欧氏距离。当用户的请求到达时，UWG 会根据请求中涉及的数据查询范围进行动态调整。为了避免查询图的过度存储开销，我们在更新图时忽略了数据的时间维度，只考虑数据的设备 ID 维度。我们使用变量 x_{ab}^k 来指示第 k 个查询的内容。当第 k 个查询包含设备 d_a 和设备 d_b 时， $x_{ab}^k = 1$ ，否则 $x_{ab}^k = 0$ 。然后，距离 l_{ab} 将根据 x_{ab}^k 进行更新。

由于用户请求可能非常随机，因此图的更新和变化可能会非常频繁。因此，我们设置了一个影响因子 θ 来确定用户请求对图更新的影响。当 θ 接近 1 时，权重受查询的影响更大。当 θ 接近 0 时，这意味着图尽可能保持初始状态。同时，TimeChain 使用滑

算法 4.1 聚合算法**Require:** 输入设备集 D , 数据集 S , 用户的历史查询集 Q^i **Ensure:** 最优聚合结果 P

```

1:  $S' \leftarrow \{s^a | a \in D \ \& \ s^a \subset S\}$  //根据设备 ID 梳理原始数据
2:  $L \leftarrow \left\{ \sqrt{(id_a - id_b)^2 + (t_a - t_b)^2} \mid \exists a, b \in D \right\}$  //根据式 4-1 初始化权重集  $L$ 
3:  $X^k \leftarrow \{0 \mid \exists a, b \in D\}$  //将用户查询请求信息  $Q^{i-1}$  打包成集合  $X^k$ 
4: for  $q^k \in Q^i$  do
5:   if  $a, b \in q^k$  then
6:     使用  $x_{ab}^k \leftarrow 1$  更新  $X^k$  //将用户查询请求信息  $Q^{i-1}$  打包成集合  $X^k$ 
7:   end if
8: end for
9:  $L \leftarrow \left\{ \theta \cdot l_{ab} + (1 - \theta) \cdot x_{ab}^k \mid \exists a, b \in D \exists l_{ab} \in L \right\}$  //更新 UWG 的权重集  $L$ 
10:  $D' \leftarrow cluster(D, L)$  //使用谱聚类算法获得聚合结果  $D'$ 
11:  $P \leftarrow \{\}$  //初始化聚合结果  $P$ 
12: for  $d^j \in D'$  do
13:   增加  $\{s^a \mid \exists a \in d^j\}$  到  $P$  中 //将设备数据合并到  $P$  中
14: end for
15: return  $P$ 

```

动窗口动态更新 UWG, 尽可能避免单个查询的对系统的鲁棒性影响。虽然查询模式的突然变化最初会降低打包准确率, 但系统会在多次查询后快速适应。通过自适应权重聚类算法, 我们可以根据节点之间的距离和查询的相关性动态调整节点之间的权重, 以更好地反映它们的相似性。这有助于在打包过程中更准确地确定哪些节点的数据应放置在同一批中, 以提高查询的效率和准确性。

4.2 基于谱聚类算法的封装机制

通过建立自适应的 UWG, 我们将数据分批的问题转化为聚类问题。在聚类算法的选择上, 我们发现传统的聚类算法如 K-means 和 GMM 并不适用于处理物联网设备产生的数据, 因为这些数据的聚类结果往往呈现出不规则的形状, 且传统算法要求将数据均

匀分配到固定数量的聚类中，这与我们的需求不符。由于谱聚类算法适用于处理不规则形状的分类问题，我们使用它来聚合数据。4.1显示 TimeChain 的整个打包过程。该算法的输入包括输入设备集 D 、数据集 S 和用户的历史查询集 Q^i 。我们首先根据设备 ID 和时间单位将原始数据 S 组织到一个集合 S' 中，该集合以时间单位表示设备的数据（第 1 行）。我们根据式 4-1 初始化权重集 L （第 2 行）。对于上一个滑动窗口中的历史查询记录集合 Q^{i-1} ，我们将用户查询请求信息打包成集合 X^k （第 4-8 行）。然后，我们根据收集到的查询信息 X^k 更新权重集 L （第 9 行）。对于 $UWG(D, L)$ ，我们使用谱聚类算法来获得聚合结果 D' （第 10 行）。根据聚合结果 D' ，我们将设备数据合并到 P 中，这是我们得到的聚合结果（第 12-14 行）。

4.3 本章小结

在本章中，我们深入研究了面向链下存储的自适应聚合机制，旨在解决物联网时序数据存储中的查询延迟问题。我们发现，不当的数据聚合策略会导致查询过程中需要跨越更多存储单元，从而增加了网络传输的延迟。为此，我们提出了一种基于历史查询的自适应无向加权图（UWG）方法，该方法能够精确捕捉用户的查询行为，并根据这些行为将数据分批处理，以优化查询效率。我们还提出了一种基于谱聚类算法的封装机制，以更好地处理不规则形状的数据聚类问题。通过这种方法，我们可以更准确地确定哪些节点的数据应放置在同一批中，以提高查询的效率和准确性。

5 基于共识协议的节点选择机制

确定最佳的存储节点选择是构建高效区块链存储系统的关键一环。根据我们的分析,如图 3.3b所示,存储节点与网关之间的距离对数据访问延迟有着直接的影响。此外,在链下存储环境中,节点存储空间不足或恶意行为可能会导致数据的丢失、被篡改或服务中断,这些都会对系统的安全性和稳定性造成严重影响。因此,我们综合考虑了节点间的距离和它们的历史服务记录,对存储节点进行了全面的评估。

同时,确存储节点选择过程的安全性同样不可忽视。现有的系统如 Storj、CoopEdge 和 PipeEdge,尽管通过区块链来确认节点选择的决策,却依然依赖于集中式的节点选择服务,这使得它们仍然面临着单点故障的风险。尽管采用类似 PBFT 的投票机制可以在一定程度上确保节点选择的安全性,但这通常涉及到多轮的计算和消息广播,增加了系统的复杂性。如果共识形成过程与节点选择过程完全分离,那么系统的安全性可能会受到威胁。

为了克服这些挑战,我们采取了一种创新的方法,将节点选择过程与共识机制紧密耦合,从而提出了一种基于共识的节点选择机制。这种方法不仅提高了节点选择的效率和安全性,而且增强了整个系统的抗攻击能力和可靠性。

5.1 结合共识的节点选择过程

为了安全地选择最佳存储节点,TimeChain 提出了一种基于共识机制的节点选择算法。整个选择过程包括**请求**、**准备**、**预提交**、**提交**和**回复**,其详细步骤如图 5.1所示,伪代码如 算法 5.1所示。与 PBFT 共识类似,在**请求**阶段,网关向系统中的所有节点发送请求,共识节点将在**回复**阶段将获得的结果返回给网关。同样类似于 PBFT 共识,我们假设拜占庭卫星的数量为 f ,卫星总数超过 $3f + 1$ 。

在**准备**阶段,每个节点通过考虑存储节点的距离、信誉等来计算分数,并将分数广播给所有其他节点。TimeChain 选择距离、剩余存储和存储质量作为指标,原因如下:根据图 3.3b,我们发现存储节点的距离会影响数据访问延迟。因此,我们选择距离作为一个重要的指标。此外,受到 FileCoin^[24]的启发,剩余存储空间和存储质量也是选择存储

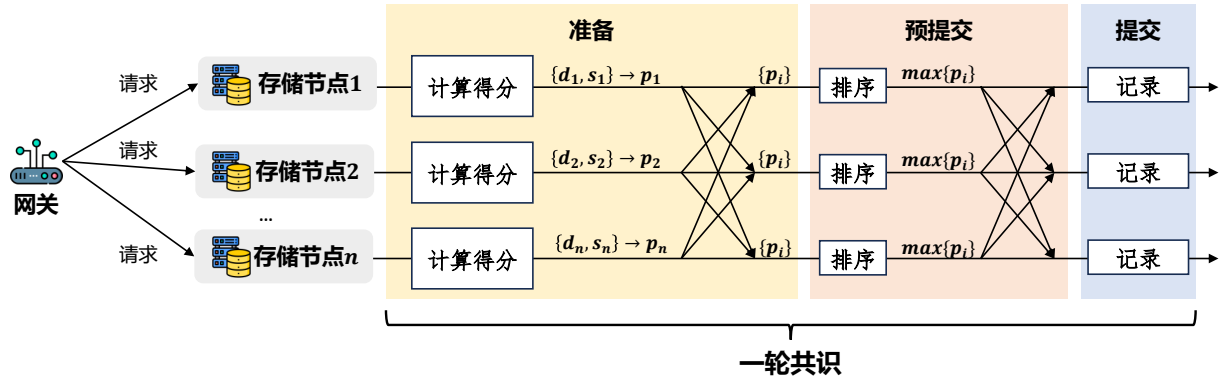


图 5.1 基于共识的节点选择机制 workflow

节点的重要因素。剩余存储空间可以直接影响存储提供商的服务能力，而存储质量可以影响数据的完整性。第 i 个节点的得分计算公式如下：

$$p_i = \alpha \cdot d_i + \beta \cdot s_i + \gamma \cdot q_i \quad (5-1)$$

其中 d_i 表示第 i 节点和客户端节点之间的距离； q_i 表示存储服务质量，可以从链上的服务记录中评估； s_i 表示节点的剩余存储空间。所有这些数据都可以在链上找到。 α 、 β 和 γ 是加权参数，这些系数可以根据特定的系统需求和性能要求进行调整。例如，可以增加的权重以优先考虑存储完整性，而可以调整的权重以降低延迟。

在**预提交**阶段，共识节点从其他节点接收准备好的消息集 $\{p_i\}$ 。当此节点的计时器超时并且收到超过 $2f+1$ 个 **PREPARE** 消息时，共识节点根据它们收到的信誉优先级 $\{p_i\}$ 决定最佳存储提供者。如果每一轮共识只返回最近的节点，由于距离对信誉计算机制的影响，一些较近节点的存储压力可能会非常高。为了平衡负载，共识节点将返回一组信誉最高的 n 节点，供网关随机选择，而不是信誉最高的节点。

在**提交**阶段，所有节点都将收到其他节点推荐的最佳存储决策。当相同的预提交消息的数量超过 $f+1$ 时，此节点将向客户端提交最佳存储节点。

5.2 共识过程的安全性分析

我们在这里考虑这一共识协议的安全性。由于 TimeChain 的共识协议只是基于 PBFT 添加了额外的信息，我们只考虑**准备**和**预提交**阶段额外信息带来的安全风险。

在**准备**阶段，如果一个节点伪造了自己的分数，网关可以很容易地检查分数的真实

性, 因为评估数据源都可以在链上找到。一旦节点伪造了其信誉, 该行为也将被记录在链上, 从而影响下一次的信誉评估。此外, 由于最后只选择了一个存储节点, 网关不关注所有节点得分的真实性, 只关注所选节点的得分。

在**预提交**阶段, 如果任何节点伪造了最终得分, 则不会影响最终结果。这是因为对于包含 f 拜占庭节点的 $3f+1$ 节点的网络, 必须在**提交**阶段获得相同的结果。

算法 5.1 共识过程

准备阶段

- 1: $request \leftarrow receive(REQUEST)$
- 2: **if** $request$ 有效 **then**
- 3: $d_i \leftarrow distance(i.pos, request.pos)$ //计算节点 i 和请求节点之间的距离
- 4: $s_i \leftarrow i.space$ //获取节点 i 的剩余存储空间
- 5: 通过式 5-1 计算 p_i
- 6: $broadcast(p_i, PREPARE)$ //广播 p_i
- 7: **end if**

预提交阶段

- 8: $\{p_i\} \leftarrow receive(\{p_i\}, PREPARE)$ //接收其他节点的 $PREPARE$ 消息
- 9: **if** $count(PREPARE) > 2 * f + 1$ 且超时 **then**
- 10: $P \leftarrow \max_n \{p_i\}$ //选择信誉最高的 n 个节点
- 11: $broadcast(P, PRECOMMIT)$ //广播最佳存储节点
- 12: **end if**

提交阶段

- 13: $P \leftarrow receive(P, PRECOMMIT)$ //接收其他节点的 $PRECOMMIT$ 消息
 - 14: **if** $count(P) > f + 1$ **then**
 - 15: $commit(P)$ //提交最佳存储节点
 - 16: **end if**
-

5.3 本章小结

在本章中，我们专注于 TimeChain 中存储节点选择模块。从前面的测试中，我们认识到，存储节点与网关之间的距离直接影响数据访问延迟，而存储节点的可靠性则关乎数据的完整性和安全性。因此，我们提出了一种基于共识协议的节点选择机制，该机制综合考虑了节点间的距离和历史服务记录，以全面评估存储节点的优劣。

我们分析了现有系统如 Storj、CoopEdge 和 PipeEdge 的局限性，指出它们依赖于集中式的节点选择服务，存在单点故障的风险。为了解决这一问题，我们将节点选择过程与共识机制紧密结合，提出了一种新的节点选择算法。该算法包括请求、准备、预提交、提交和回复五个阶段，通过这一流程，我们能够在确保安全性的同时，选择出最佳的存储节点。

在安全性分析中，我们证明了所提出的共识协议在 PBFT 基础上增加的信息并未引入额外的安全风险，因此该机制是安全的。我们的共识协议考虑了准备阶段和预提交阶段的安全性，确保即使在存在恶意节点的情况下，系统也能正确选择出最佳的存储节点。

6 基于 LSH 树的验证机制

我们的实验数据揭示了一个显著的现象：在传输的数据中，数据完整性证明几乎占据了一半的比例。这些证明通常以默克尔树的形式组织，由一连串的哈希值构成。在默克尔树结构中，非叶子节点的哈希数量与原始数据点的数量大致相同。鉴于物联网数据单元的大小与哈希值相近，这意味着为了验证数据的完整性，需要传输的数据量几乎是原始数据量的两倍。在数据传输过程中，如何减少完整性证明大小，成为了一个亟待解决的难题。

深入分析物联网数据的特性后，我们注意到物联网数据的变化通常较为缓慢，短期内不太可能发生剧烈变动。针对这些高度相似的数据点，LSH 算法能够从相似的原始数据中产生相似的哈希结果。LSH 算法的这一特性保证了即使在哈希处理之后，相似的物联网数据点仍然保持相似性。基于这一观察，我们设计了一种创新的基于 LSH 树的验证机制，它采用 LSH 算法替代了传统默克尔树中使用的通用哈希函数。通过仅传输 LSH 哈希值的差异部分，我们能够显著减少在验证过程中需要传输的数据量，从而提高了数据传输的效率。

6.1 LSH 树结构与工作流程

对位置敏感的哈希将类似的数据映射到类似的哈希值以进行重复数据删除。顾名思义，我们通常打包在物理空间或时间上接近的数据，这些数据往往具有很高的局部相似性。因此，通过在同一批上使用局部敏感哈希，我们可以得到相似的哈希值。当需要传输哈希值时，只传输哈希差部分，从而减少了要传输的数据量。

我们在图 6.1 中展示了 LSH 树的一个例子。具体来说，对于批处理中的数据，我们采取类似于默克尔树的步骤，首先对原始数据执行局部敏感哈希。对于哈希结果，我们将两个相近的哈希合并为一个字符串，并计算该字符串的局部敏感哈希值。然后，我们逐层向上重复这个过程，直到我们得到一个唯一的哈希值，即哈希根。在每一级哈希中，由于原始数据的高度相似性，哈希值的许多位是相同的。因此，在传输每一级哈希值时，我们只能传输不同的比特以减少传输延迟。同样，由于每一级哈希中存在局部相

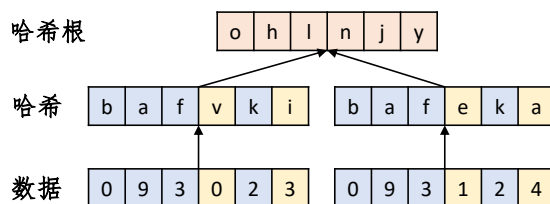


图 6.1 LSH 树结构

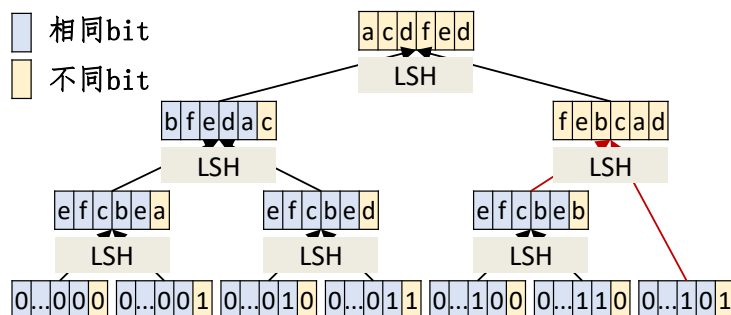


图 6.2 非满二叉 LSH 树

似性，第二级哈希中的许多比特也将相似。通过类比，对于每一层的哈希值，我们只需要传输哈希差比特，从而进一步减少了传输的数据量。

由于我们使用 LSH 树而不是原始的默克尔树，因此我们需要分析 LSH 树的安全性。这里我们主要考虑存储提供者篡改数据的情况，即仍然可以用不同的原始数据获得相同的哈希值的情况。我们测试了 LSH 树的不同层，发现在最接近数据源的哈希层中，哈希的平均差位数为 170 位。这已经超过了 MD5 和 SHA1 标准，这些标准现在在物联网场景中非常常用^[37-38]。对于靠近根节点的 LSH 树的层级，尽管哈希差位数较少，但这对篡改原始数据没有意义。

6.2 LSH 树的尾部合并机制

在一个完整的二叉树中，LSH 树可以通过成对地批量合并数据来执行局部敏感哈希。然而，如果批处理中的数据数量不足以形成完整的二叉树，那么像默克尔树一样构建哈希树将导致相似性的丧失。例如，在图 6.2 中，一批中有 7 个数据点，对于一个完整的二叉树来说，这个大小是不满足的。在第一轮哈希中，前 6 个数据成对执行 LSH。由于原始数据的相似性，这 6 个数据的哈希结果是相似的。在第二轮哈希中，由于数据 5-6 和第 7 个原始数据的第一轮哈希结果非常不同，这两个数据的哈希结果也与数据点 1-4 的哈希结果非常不一样。在执行完整性证明时，需要传输所有这些不同的哈希数据

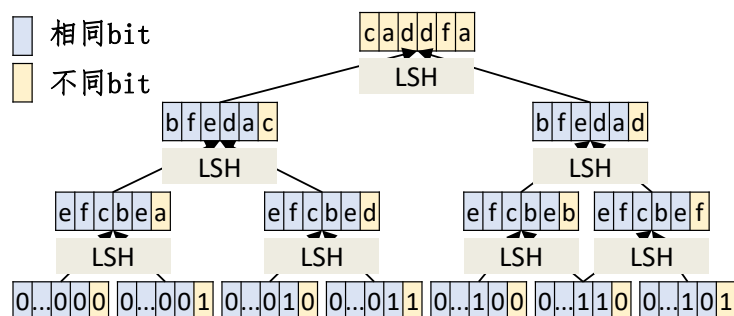


图 6.3 尾部合并下的非满二叉 LSH 树.

位，这增加了传输的数据量。

为了解决这个问题，我们引入了尾部合并策略，将非全二叉树的尾部节点与同一层的前节点合并。如图 6.3 所示，在第一轮哈希中，我们将第 7 个节点和第 6 个节点合并，以尽可能保持数据的相似性。数据 6-7 的哈希结果是 `efcbeb`，数据 5-6 的哈希值是 `efcbef`。显然，在第一轮哈希中存在很高的相似性，并且可以保持到下一级哈希。这将传输的哈希值大小从 12 位减少到 7 位，代价是在第一轮哈希中只传输了 1 个不同的比特。这样，在进行完整性证明时，我们只需要传输不同的哈希位，从而减少了传输的数据量。

6.3 本章小结

在本章中，我们针对物联网数据完整性验证过程中的数据传输延迟问题，提出了一种基于 LSH 树的新型验证机制。我们发现，在传输的数据中，数据完整性证明占据了相当大的比例，且通常以默克尔树的形式组织，这导致为了验证数据完整性需要传输的数据量几乎是原始数据量的两倍。为了解决这一问题，我们利用物联网数据变化缓慢且高度相似的特性，采用 LSH 算法替代了传统默克尔树中的通用哈希函数，通过仅传输 LSH 哈希值的差异部分，显著减少了验证过程中需要传输的数据量。

我们详细分析了 LSH 树的结构和工作流程，并针对非满二叉树的情况，引入了尾部合并策略以保持数据的相似性，进一步减少了传输的数据量。此外，我们还对 LSH 树的安全性进行了分析，确保了其在最接近数据源的哈希层中具有足够的安全性，超过了现有的 MD5 和 SHA1 标准，在物联网场景中足够安全。

7 实验结果与分析

在本章中，我们将对 TimeChain 系统进行全面的实验评估，从存储延迟和查询延迟两个关键维度来验证 TimeChain 设计的有效性。我们将分析 TimeChain 在不同查询负载和存储网络条件下的性能表现，并针对系统中的自适应聚合机制、基于共识的节点选择机制和基于 LSH 树的验证机制进行消融实验，以展示这些技术点对系统性能的具体提升。通过这些实验，我们旨在展示 TimeChain 相比于现有解决方案在区块链存储系统中的性能优势。

7.1 实验设置

我们基于一些开源项目（如 Hyperledger Fabric 和 IPFS）实现 TimeChain。我们使用阿里云的 50 台虚拟机构建了一个基于 Hyperledger Fabric 的底层区块链系统，每个节点配备 2 核 CPU 和 4GB 内存。块大小设置为 1500 个交易，块间隔为 1 秒。我们基于一个真实世界的存储网络^[27]，模拟了分布在世界各地的 320 个云服务器节点，并基于该集群进行了实验。每个存储节点配置 2 核 CPU 和 4GB 内存，每个存储节点的存储空间为 512GB。存储节点和网关之间的距离从 800 公里到 6000 公里不等，平均为 4000 公里。我们使用现有的线性回归模型^[28]模拟存储节点和网关之间的传输延迟。考虑到一些存储提供商存在欺诈行为，部分远程存储的数据将有概率无法被访问到，概率为 60%。我们使用电脑（Personal Computer, PC）作为物联网传感器的网关节点，它配备了 Intel(R)Core i7-13700K CPU@5.4GHz、32GB DRAM，并运行 Ubuntu 22.04。默认聚合存储单元大小和查询大小设置为 20。

7.1.1 基线

SEBDB^[3]是链上存储的典型代表。它通过将所有数据存储在区块链上，并使用 B+ 树创建时间戳和设备名称的快速索引实现了对链上区块的高效访问。在数据验证方面，SEBDB 使用传统的默克尔树进行数据验证。默克尔树通过计算数据块的哈希值，并将这些哈希值逐层组织成树结构，来实现数据完整性验证。尽管 SEBDB 不是专门为物联

网数据设计的链下存储解决方案，但是因为它的数据验证机制类似于 TimeChain，我们将它作为对比方案。为了保证实验的公平性，我们调整了 SEBDB 来记录单个数据项的存储位置并将其上传到链中，并随机选择存储节点。这些修改确保了实验的公平性，同时更好地反映了物联网数据的特点和需求。

FileDES^[26]是一个基于文件的链下存储系统。它通过在远程节点上存储文件并在链上记录文件的哈希值来实现文件的安全存储和可靠性。当客户端需要搜索文件时，FileDES 会遍历区块链上的所有块，以获取文件的存储位置。在数据验证方面，FileDES 也使用与 SEBDB 相同的默克尔树。由于 FileDES 是为文件存储设计的，为了确保实验的公平性，我们将 FileDES 改造成为适用于物联网数据的链下存储系统。对于到达网关的物联网数据，FileDES 将这些数据以固定大小、按照到达网关的顺序进行打包，并将这些数据上传到远程存储节点。

7.1.2 数据集和工作负载

我们使用了以下三个数据集进行实验：

港珠澳大桥 (Bridge)^[39]: Bridge 数据集包含 4M 条数据记录，这些数据是在 5 天内从港珠澳大桥上的 6 种传感器采集的。这 6 种传感器与桥梁健康监测有关，包括桥梁各个位置的振动加速度、挠度等，共计 53 个。由于数据生成频率为 10Hz，我们将平均数据查询范围设置为 40。

RT-IFTTT (RT)^[40]: RT 数据集包含 680K 条数据记录。这些数据是在 10 天内从 10 个真实传感器采集的值，包括温度、湿度、可见光和其他传感器。数据以秒为单位收集，因此查询的平均范围设置为 20。

天气 (WX)¹: WX 数据集包含 1.5M 条数据记录，其中包括来自五年内 30 多个城市的各种天气属性每小时测量的数据。鉴于数据收集频率相对较低，间隔最多一小时，我们假设查询范围为 10。

考虑到时间序列存储系统的存储特性^[41]，我们将这些数据集的格式转换为 $\langle measurement, field\ name, field\ value, timestamp \rangle$ 。measurement 类似于数据表，用于组织数据，例如“天气”；field name 用于标识不同的字段，例如“温度”；field value 存储的是实际数据，比如温度传感器的实际值；timestamp 是时间戳，记录了数据的时间

¹<https://www.kaggle.com/selfishgene/historical-hourly-weather-data/>

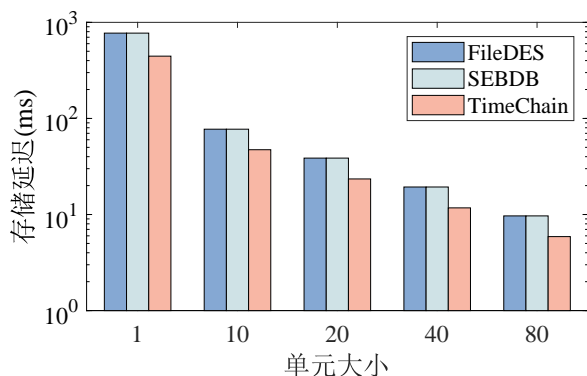


图 7.1 存储延迟

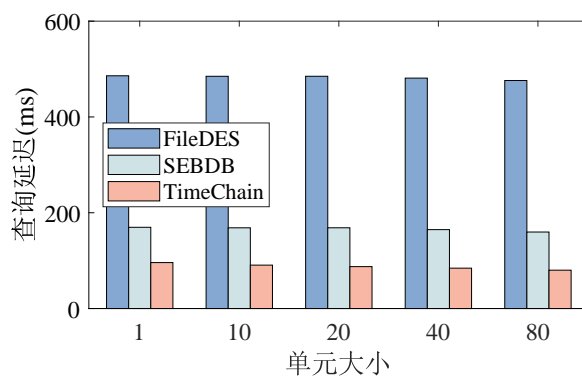


图 7.2 查询延迟

信息。考虑到数据集的特点，我们将这些数据的大小分别固定为 8B、32B、8B、8B。因此，一条数据总共占用的存储空间为 56B。

7.2 总体性能

7.2.1 存储延迟

我们比较了不同聚合单元大小下的存储延迟。由于物联网设备数量众多，数据生成速度快，我们将存储延迟定义为存储 10000 个数据的总延迟，而不是关注单个数据的细微延迟。如图 7.1 所示，对于不同的聚合单元大小，TimeChain 的存储延迟低于 SEBDB 和 FileDES。这是因为 TimeChain 的独特打包机制和节点选择机制减少了数据传输延迟。

此外，随着聚合单元大小变大，存储延迟变小。这是因为，对于相同数量的数据，当聚合单元较大时，数据在链中打包和记录的次数会减少。然而，较大的聚合单元可能会引入较大的存储等待延迟。因为一个聚合单元可能需要等待较长时间才可以被填满，才能被打包和记录。具体的聚合单元大小需要根据实际数据产生速度和用户的需求来确定。用户可以通过调整网关的聚合单元大小来控制实际的存储延迟。

7.2.2 查询延迟

我们比较了不同聚合单元大小的查询延迟，如图 7.2 所示。查询延迟是指基于固定查询范围随机查询数据集的平均延迟，查询范围设置为 20。从图 7.2 可以看出，TimeChain 的查询延迟低于其他两种方案。这是由于 TimeChain 减少了数据传输延迟，原因将在图 7.3 中解释。

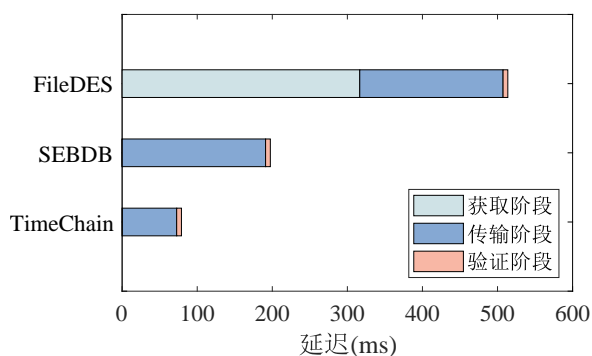


图 7.3 查询延迟分解

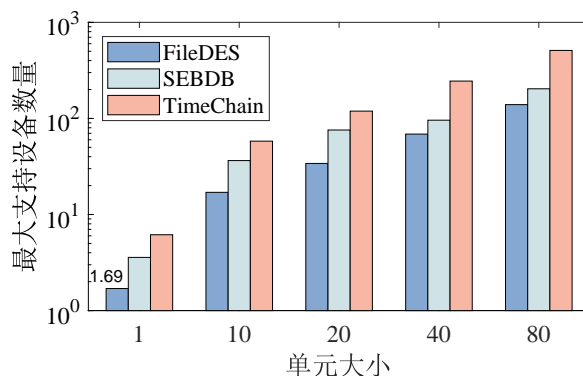


图 7.4 最大支持存储设备数

我们还可以发现，当聚合单元大小增加时，查询延迟会减少。这是因为当聚合单元大小增加时，同一查询中涉及的聚合单元数量会减少，用户的查询结果将大概率集中在一个存储节点上。然而，聚合单元越大，TimeChain 相对于其他解决方案的改进将随着聚合单元的增大而减少。这是因为当聚合单元非常大时，相当于将所有数据存储在一个聚合单元中。在这种情况下，对于数据进行谱聚类切分不会导致性能提高，反而会导致对多种范围查询的灵活性降低。此外，当大量数据集中在存储节点中时，存储系统的可扩展性和可靠性也会受到损害。

7.2.3 查询延迟分解

我们进一步分析了查询延迟的细分,以确定影响整体查询性能的主要因素,如图 7.3 所示。查询的延迟主要由 4 个阶段组成,检索、传输、验证和返回。考虑到传感器通常选择更近的网关,返回阶段的延迟几乎可以忽略不计。在验证阶段,三种方案的延迟相对接近,小于 1ms,也可以被忽略。

在传输和检索阶段,我们发现延迟占据了查询延迟的主要部分。特别是 TimeChain 的传输延迟明显低于 FileDES 和 SEBDB。这一优势归功于 TimeChain 的节点打包机制,该机制优化了数据的组织方式,减少了从存储节点获取数据的次数。同时,TimeChain 的节点选择机制通过选择地理位置更近的存储提供商进一步缩短了数据传输的距离,从而降低了传输延迟。

在检索阶段,FileDES 由于需要遍历区块链上的所有块来检索数据,导致了较高的检索延迟。这种机制在面对大量数据时效率较低,尤其是在数据量迅速增长的场景下。相比之下,SEBDB 和 TimeChain 通过使用 B+ 树和 R 树索引结构,显著提高了检索效

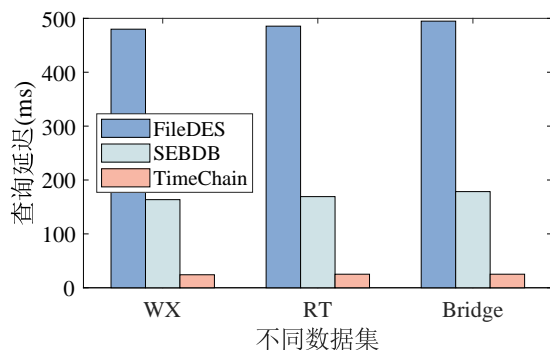


图 7.5 不同查询大小下的查询延迟

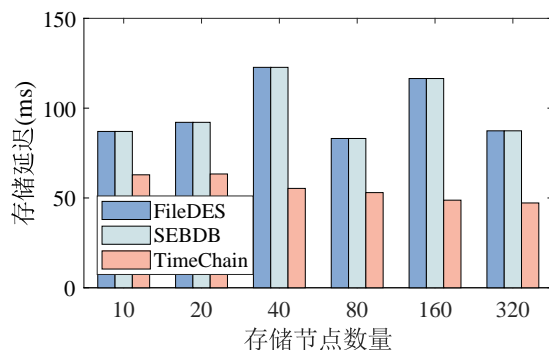


图 7.6 不同存储网络下的存储延迟

率，降低了延迟。这两种索引结构允许系统快速定位到存储数据的位置，从而加快了检索速度。

7.2.4 支持的最大存储设备数量

在本小节中，我们使用的支持设备最大数量指标，即存储系统每秒可以提供存储服务的设备数量。我们假设网关可以同时处理来自多个物联网设备的数据存储请求，并忽略网关本身的处理延迟。所有物联网设备同时以 1Hz 的频率生成数据，并要求在生成下一个数据之前必须存储这些数据。在这样的限制下，存储系统可以并行支持的最多设备数量即为支持的最大存储设备数量。如图 7.4 所示，TimeChain 支持最大设备数量分别是 SEBDB 和 FileDES 的 1.63 倍和 3.55 倍。这主要是由于 TimeChain 对数据传输延迟采取的优化技术，较小的数据传输延迟允许 TimeChain 以更快的速度存储数据。此外，对于所有测试的方案而言，随着聚合单元大小的增加，支持的最大设备数量也随之增加。这是因为较大的聚合单元意味着单个链上哈希可以代表更多的数据量，从而提高了单个存储单元的效率，使得系统能够支持更多的物联网设备。特别地，当聚合单元大小达到 80 时，TimeChain 支持的最大设备数已经达到了千级，这一结果充分展示了 TimeChain 在大规模物联网环境中的扩展能力和高效性。这一性能优势不仅证明了 TimeChain 在处理大规模数据时的可靠性，也为未来物联网应用的发展提供了强有力的技术支持。

7.3 性能影响因素评估

7.3.1 不同查询大小下的查询延迟

我们比较了三种不同查询大小数据集（Bridge、RT 和 WX）的查询性能，结果如图 7.5 所示。三个数据集中工作负载的平均查询大小分别为 40、20 和 10。我们可以发现，这三种解决方案的查询延迟通常随着查询大小的增加而增加，即 WX 最小，RT 次之，Bridge 最大。这是因为更大的查询大小涉及了更多的聚合单元，因此可能需要去更多的存储节点获取数据，从而增加了数据传输延迟。我们观察到，与 SEBDB 和 FileDES 相比，当查询大小变大时，TimeChain 的聚类算法带来的性能改进也会增加。这是因为当查询大小变大时，SEBDB 和 FileDES 通常需要从比 TimeChain 更多的节点获取数据，而 TimeChain 可以通过聚类算法减少数据获取的次数，从而减少数据传输延迟。

7.3.2 不同存储网络规模下的存储延迟

我们比较了不同存储网络规模下每种解决方案的存储延迟，如图 7.6 所示。随着存储节点数量的增加，TimeChain 的存储延迟呈下降趋势。这一现象的原因在于，TimeChain 的网关在选择存储提供商时拥有更大的灵活性，能够从更多的节点中选择，从而增加了选择到地理位置更接近的存储节点的机会，这自然降低了数据传输的时间。相比之下，FileDES 和 SEBDB 由于采用了随机选择存储节点的机制，它们在存储节点数量增加时，并没有表现出存储延迟的显著变化。这种随机性导致了这两种方案在存储延迟上缺乏可预测性，且在不同存储网络规模下，它们的性能波动较大。由于 FileDES 和 SEBDB 在选择存储节点时没有考虑节点的具体位置信息，因此它们无法充分利用增加的节点来优化存储延迟。这就导致了即使在存储节点数量增加的情况下，这两种方案的存储延迟仍然保持在一个相对较高的水平，且波动较大，这在一定程度上限制了它们在大规模存储网络中的应用潜力。

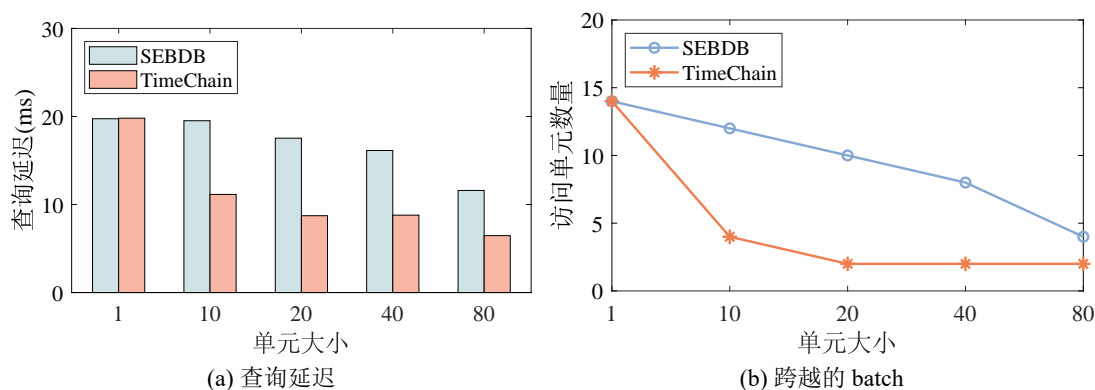


图 7.7 聚类算法消融实验

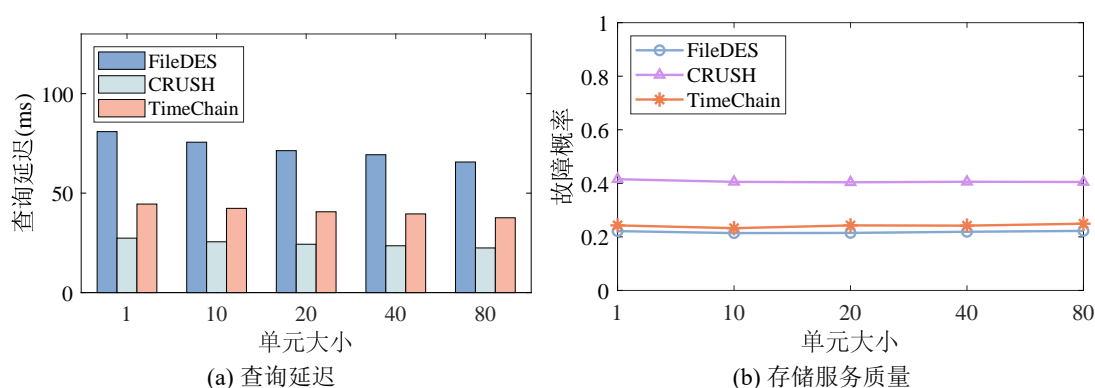


图 7.8 节点选择消融实验

7.4 消融实验

7.4.1 聚类算法

我们比较了聚类算法的效果，并将 TimeChain 与 SEBDB 进行了比较。如 7.7a 所示，与 SEBDB 相比，TimeChain 的网络传输延迟减少了 40.3%。这是因为 SEBDB 的打包节点未能充分考虑用户查询的规律性，只是将物联网数据根据到达网关的顺序进行打包。这导致在处理数据拥有者请求的数据时，SEBDB 不得不访问多个存储提供商所管理的聚合单元，增加了数据检索的复杂性和延迟。相比之下，TimeChain 采用了谱聚类算法，这种算法能够根据用户请求的特征对来自特定传感器的数据进行打包。这种方法不仅优化了数据在存储节点中的分布，还减少了数据访问时需要跨越的聚合单元数量。如图 7.7b 所示，TimeChain 访问的聚合单元数量比 SEBDB 少了 59.3%，这表明 TimeChain 在减少数据访问延迟和提高查询效率方面具有明显优势。

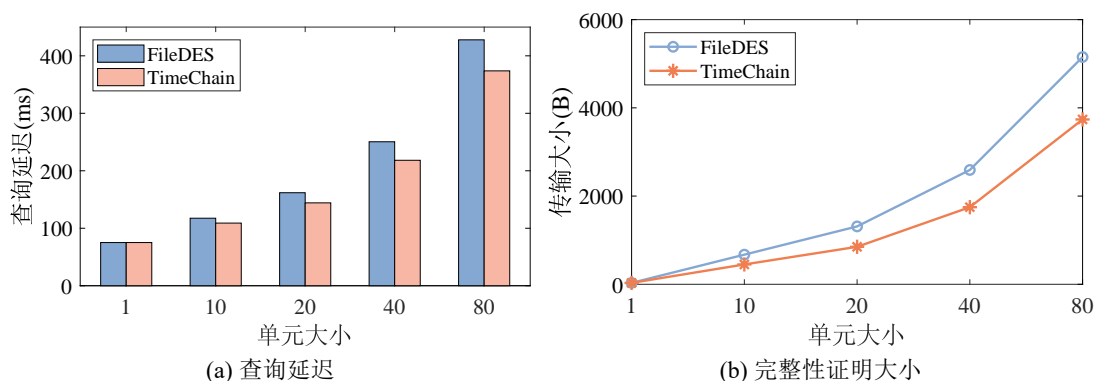


图 7.9 LSH 树消融实验

7.4.2 节点选择

我们在 7.8a 和 7.8b 中显示了 TimeChain、FileDES^[26] 和 CRUSH^[19] 在节点选择方面的性能差异。

FileDES 通过基于节点信誉的筛选机制, 确定了一组受信任的存储节点, 并在此基础上随机选择节点进行数据存储。这种方法的优点在于提高了数据存储的安全性, 因为只有信誉良好的节点才被考虑用于存储。然而, 由于缺乏对节点地理位置的考虑, FileDES 的随机选择可能导致选择了距离较远的节点, 从而增加了数据传输的延迟, 影响了整体性能。

CRUSH 则采取了一种基于物理位置的策略, 优先选择距离最近的节点进行数据存储。这种策略在不考虑节点故障和可靠性的情况下, 能够显著减少数据传输的时间。然而, CRUSH 方案的一个主要缺陷是它没有充分考虑节点的可靠性, 如 7.8b 所示, 这导致有 41% 的节点可能无法提供有效的存储服务, 这对于需要高可靠性的存储系统来说是一个严重的缺陷。

与 FileDES 和 CRUSH 相比, TimeChain 在节点选择上采取了一种综合考虑节点物理距离和信誉的策略。TimeChain 不仅考虑了节点的地理位置以减少传输延迟, 还考虑了节点的历史服务记录和信誉, 以确保所选节点的可靠性。这种综合考虑的方法使得 TimeChain 在响应时间和服务概率方面都展现出了最佳性能。尽管 TimeChain 选择的节点可能不是物理距离上最近的, 但通过平衡节点的距离和服务质量, TimeChain 能够提供更稳定和高效的存储服务。

7.4.3 LSH 树

我们比较了 FileDES 和 TimeChain 的网络传输延迟,如 7.9a所示。与 FileDES 相比,TimeChain 的数据传输延迟减少了 10.9%。这主要是来源于 TimeChain 对数据传输量的优化。在物联网中,局域网内传感器设备的数量众多,且这些设备通常以高频率持续生成数据。如 7.9b所示,由于 TimeChain 使用 LSH 作为哈希算法,通过网络传输的数据量大大减少。这种数据量的减少不仅降低了存储和查询操作的延迟,还显著减轻了存储提供商的存储负担。对于存储提供商而言,这意味着更低的存储成本和更高的数据处理效率。同时,对于整个存储系统来说,减少了数据传输量还有助于提高系统的吞吐量和响应速度,尤其是在高负载或资源受限的环境中。

7.5 本章小结

在本章中,我们对 TimeChain 系统进行了全面的实验评估,以验证其在存储性能方面的优势。通过在模拟的全球存储网络上进行测试,我们比较了 TimeChain 与现有解决方案,SEBDB 和 FileDES,在存储延迟和查询延迟方面的表现。实验结果表明,TimeChain 在减少查询延迟和存储延迟方面具有显著优势,平均查询延迟降低了 64.6%,存储延迟降低了 35.3%。

我们还分析了不同查询负载和存储网络规模对 TimeChain 性能的影响,发现随着聚合单元大小的增加,TimeChain 的存储和查询性能得到了进一步的提升。此外,TimeChain 在支持最大存储设备数量方面也展现出了其可扩展性,与 SEBDB 和 FileDES 相比,分别增加了 1.63 倍和 3.55 倍。

为了进一步验证 TimeChain 设计中关键技术点的有效性,我们进行了消融实验。实验结果证实了自适应聚合机制、基于共识的节点选择机制和基于 LSH 树的验证机制在提升系统性能方面的重要作用。特别是 LSH 树机制,通过减少传输的数据量,显著降低了网络传输延迟。

8 总结与展望

8.1 总结研究

本文提出了 TimeChain, 一个基于区块链的物联网时序数据高效可信存储系统。针对物联网场景下数据生成速度快、数据量大、需要高效率和高可信度的数据存储问题, TimeChain 仅将每批数据的哈希值存储在链上, 而完整的原始数据则保存在链外。TimeChain 针对范围查询性能的瓶颈, 分别提出了自适应聚合机制、基于共识的存储节点选择机制和基于 LSH 树的数据完整性验证机制。本文的主要创新点总结如下:

1. 提出了一种面向物联网数据的基本处理方案, 即把每批数据的哈希值存储在链上以减少数据开销并提高存储效率。本文也对这个基本方案进行了一系列测量, 发现了基本方案性能较差的核心原因, 为后续的优化提供了方向。

2. 设计了一种自适应聚合机制, 通过构建基于历史查询的无向加权图, 将数据打包问题转化为图划分问题, 并使用谱聚类算法优化数据的批处理存储, 以减少聚合查询期间的节点访问次数。

3. 提出了一种基于共识的存储节点选择机制, 结合存储节点信誉和传输距离, 通过将共识过程与节点选择相结合, 以减少传播延迟并提高节点选择的安全性。

4. 创新性地设计了基于 LSH 树的数据完整性验证机制, 利用物联网数据的局部相似性, 仅传输证明的非冗余部分, 显著减少了完整性验证所需的数据量。

我们基于开源组件实现了 TimeChain 系统, 并在一个模拟的存储网络中进行评估。实验结果表明, TimeChain 相比于现有的基于区块链的存储系统, 平均减少了 64.6% 的查询延迟和 35.3% 的存储延迟, 证明了其在存储性能方面的显著优势。

8.2 未来工作

在未来的工作中, 我们计划针对 TimeChain 系统进行以下几方面的研究和改进, 以进一步提升系统性能并适应不同的应用场景。

1. 我们将在真实的环境中实现并部署 TimeChain, 以验证其在实际应用中的性能和可靠性, 确保在实际部署中能够满足各种应用场景的需求。这包括在不同的网络条件

下，对算法的响应时间和吞吐量进行评估。

2. 由于 PBFT 算法存在吞吐量有限的问题，对于数据生成速度逐渐增加和设备逐渐扩张的物联网场景，PBFT 可能无法满足系统的性能需求。我们计划探索其他轻量化或异步共识算法，如 Raft 和 Tendermint，将这些高效算法与 TimeChain 结合。这些算法能够在不牺牲安全性的前提下，减少通信开销，提高 TimeChain 的响应速度和吞吐量，特别适合资源受限的物联网设备。我们也将考虑其他提升区块链性能的方法，如分片、侧链等，以提高系统的可扩展性和性能，并进一步分析可能带来的安全风险。

3. 在引入新的共识算法的同时，我们也将重点关注 TimeChain 的安全性和稳定性。这包括对我们所提共识协议和 LSH 树的安全性进行实验验证。我们将在一个真实、存在欺诈节点的物联网环境中进行实验，分析 TimeChain 对恶意网络的容忍性，以及对数据完整性和隐私性的保护能力。

参考文献

- [1] HUNG M. Leading the IoT, Gartner insights on how to lead in a connected world[J]. Gartner Research, 2017, 1: 1-5.
- [2] CHEN F, WANG J, JIANG C, et al. Blockchain based non-repudiable IoT data trading: Simpler, faster, and cheaper[C]//IEEE INFOCOM 2022-IEEE Conference on Computer Communications. 2022: 1958-1967.
- [3] ZHU Y, ZHANG Z, JIN C, et al. SEBDB: Semantics empowered blockchain database[C]//2019 IEEE 35th international conference on data engineering (ICDE). 2019: 1820-1831.
- [4] XU C, ZHANG C, XU J. vchain: Enabling verifiable boolean range queries over blockchain databases [C]//Proceedings of the 2019 international conference on management of data. 2019: 141-158.
- [5] WANG H, XU C, ZHANG C, et al. vchain+: Optimizing verifiable blockchain boolean range queries [C]//2022 IEEE 38th International Conference on Data Engineering (ICDE). 2022: 1927-1940.
- [6] LI C, BEILLAH S M, YANG G, et al. {LVMT}: An efficient authenticated storage for blockchain [C]//17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23). 2023: 135-153.
- [7] ZHANG C, XU C, HU H, et al. {COLE}: A Column-based Learned Storage for Blockchain Systems [C]//22nd USENIX Conference on File and Storage Technologies (FAST 24). 2024: 329-345.
- [8] ZAMANI M, MOVAHEDI M, RAYKOVA M. Rapidchain: Scaling blockchain via full sharding[C]//Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. 2018: 931-948.
- [9] HONG Z, GUO S, ZHOU E, et al. Gridb: scaling blockchain database via sharding and off-chain cross-shard mechanism[J]. Proceedings of the VLDB Endowment, 2023, 16(7): 1685-1698.
- [10] EL-HINDI M, BINNIG C, ARASU A, et al. BlockchainDB: A shared database on blockchains[J]. Proceedings of the VLDB Endowment, 2019, 12(11): 1597-1609.
- [11] LABS S. Storj: A decentralized cloud storage network framework[Z]. 2018.
- [12] MCCONAGHY T, MARQUES R, MÜLLER A, et al. Bigchaindb: a scalable blockchain database[J]. white paper, BigChainDB, 2016: 53-72.
- [13] Sia[Z]. <https://sia.tech/>. Accessed: [2024.4].
- [14] LAKSHMAN A, MALIK P. Cassandra: a decentralized structured storage system[J]. ACM SIGOPS operating systems review, 2010, 44(2): 35-40.
- [15] DECANDIA G, HASTORUN D, JAMPANI M, et al. Dynamo: Amazon's highly available key-value store[J]. ACM SIGOPS operating systems review, 2007, 41(6): 205-220.
- [16] CHANG F, DEAN J, GHEMAWAT S, et al. Bigtable: A distributed storage system for structured data [J]. ACM Transactions on Computer Systems (TOCS), 2008, 26(2): 1-26.
- [17] CORBETT J C, DEAN J, EPSTEIN M, et al. Spanner: Google's globally distributed database[J]. ACM Transactions on Computer Systems (TOCS), 2013, 31(3): 1-22.
- [18] CERVIN A, HENRIKSSON D, OHLIN M. TrueTime 2.0—reference manual[J]. Department of Automatic Control, Lund University, 2016.
- [19] WEIL S, BRANDT S A, MILLER E L, et al. Ceph: A scalable, high-performance distributed file system[C]//Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06). 2006: 307-320.
- [20] TAFT R, SHARIF I, MATEI A, et al. Cockroachdb: The resilient geo-distributed SQL database[C]//Proceedings of the 2020 ACM SIGMOD international conference on management of data. 2020: 1493-1509.
- [21] ZHOU E, HONG Z, XIAO Y, et al. MSTDB: a hybrid storage-empowered scalable semantic blockchain database[J]. IEEE Transactions on Knowledge and Data Engineering, 2022.
- [22] PENG Y, DU M, LI F, et al. FalconDB: Blockchain-based collaborative database[C]//Proceedings of the 2020 ACM SIGMOD international conference on management of data. 2020: 637-652.
- [23] XU C, ZHANG C, XU J, et al. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing[J]. Proceedings of the VLDB Endowment, 2021, 14(11): 2314-2326.

- [24] BAUER D P. Filecoin[G]//Getting Started with Ethereum: A Step-by-Step Guide to Becoming a Blockchain Developer. Springer, 2022: 97-101.
- [25] VORICK D, CHAMPINE L. Sia: Simple decentralized storage[J]. Retrieved May, 2014, 8: 2018.
- [26] XU M, ZHANG J, GUO H, et al. FileDES: A Secure, Scalable and Succinct Decentralized Encrypted Storage Network[C]//IEEE INFOCOM 2024-IEEE Conference on Computer Communications. 2024: 1-10.
- [27] CORNEO L, EDER M, MOHAN N, et al. Surrounded by the clouds: A comprehensive cloud reachability study[C]//Proceedings of the Web Conference 2021. 2021: 295-304.
- [28] ZIVIANI A, FDIDA S, DE REZENDE J F, et al. Improving the accuracy of measurement-based geographic location of Internet hosts[J]. Computer Networks, 2005, 47(4): 503-523.
- [29] CAESAR H, BANKITI V, LANG A H, et al. nusenes: A multimodal dataset for autonomous driving [C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 11621-11631.
- [30] BHATIA M, AHANGER T A, MANOCHA A. Artificial intelligence based real-time earthquake prediction[J]. Engineering Applications of Artificial Intelligence, 2023, 120: 105856.
- [31] BARATA M, BERNARDINO J, FURTADO P. Ycsb and tpc-h: Big data and decision support benchmarks[C]//2014 IEEE International Congress on Big Data. 2014: 800-801.
- [32] XU R, WUNSCH D. Survey of clustering algorithms[J]. IEEE Transactions on neural networks, 2005, 16(3): 645-678.
- [33] KANUNGO T, MOUNT D M, NETANYAHU N S, et al. An efficient k-means clustering algorithm: Analysis and implementation[J]. IEEE transactions on pattern analysis and machine intelligence, 2002, 24(7): 881-892.
- [34] HE X, CAI D, SHAO Y, et al. Laplacian regularized Gaussian mixture model for data clustering[J]. IEEE transactions on knowledge and data engineering, 2010, 23(9): 1406-1418.
- [35] YUAN L, HE Q, TAN S, et al. Coopedge: A decentralized blockchain-based platform for cooperative edge computing[C]//Proceedings of the Web Conference 2021. 2021: 2245-2257.
- [36] YUAN L, HE Q, CHEN F, et al. PipeEdge: A trusted pipelining collaborative edge training based on blockchain[C]//Proceedings of the ACM Web Conference 2023. 2023: 3033-3043.
- [37] CHI L, ZHU X. Hashing techniques: A survey and taxonomy[J]. ACM Computing Surveys (Csur), 2017, 50(1): 1-36.
- [38] LANDGE I A, SATOPAY H. Secured IoT through hashing using MD5[C]//2018 fourth international conference on advances in electrical, electronics, information, communication and bio-informatics (AEEICB). 2018: 1-5.
- [39] ZHANG W, GUO C, GAO Y, et al. Edge Cloud Collaborative Stream Computing for Real-Time Structural Health Monitoring[J]. arXiv preprint arXiv:2310.07130, 2023.
- [40] HEO S, SONG S, KIM J, et al. Rt-ifttt: Real-time iot framework with trigger condition-aware flexible polling intervals[C]//2017 IEEE Real-Time Systems Symposium (RTSS). 2017: 266-276.
- [41] NAQVI S N Z, YFANTIDOU S, ZIMÁNYI E. Time series databases and influxdb[J]. Studienarbeit, Université Libre de Bruxelles, 2017, 12: 1-44.

作者简历

教育经历

2018.09 2022.06 本科 南京理工大学 软件工程
2022.09 2025.03 硕士 浙江大学 计算机技术

发表论文

- [1] Tong Sun, Borui Li, **Yixiao Teng**, Yi Gao, Wei Dong. dTEE: A Declarative Approach to Secure IoT Applications Using TrustZone. Proc. of the 23rd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'24). [CCF B 类会议]
- [2] Wenzhao Zhang, **Yixiao Teng**, Yi Gao, Wei Dong. Reducing End-to-End Latency of Trigger-Action IoT Programs on Containerized Edge Platforms. IEEE Transactions on Mobile Computing (TMC), 2024. [CCF A 类期刊]

投稿中论文

- [1] **Yixiao Teng**, Jiamei Lv, Yi Gao, Wei Dong, MEChain: MEC-aided Blockchain Network with Joint Storage Computation Offloading, Submitted to IEEE Transactions on Services Computing (TDSC), under review. [CCF A 类期刊]
- [2] **Yixiao Teng**, Jiamei Lv, Ziping Wang, Yi Gao, Wei Dong, TimeChain: A Secure and Decentralized Off-chain Storage System for IoT Time Series Data, Submitted to ACM Web Conference, 2025. [CCF A 类会议]
- [3] Tong Sun, Bowen Jiang, Hailong Lin, **Yixiao Teng**, Borui Li, Yi Gao, Wei Dong, TensorShield: Safeguarding On-Device Inference without Model Modifications by Shielding Critical DNN Tensors with TEE, Submitted to ACM MobiCom 2025. [CCF A 类会议]
- [4] Wenzhao Zhang, Renjie Li, Yongqi Chen, **Yixiao Teng**, Yi Gao, Wei Dong, Elink: Enabling Realtime Processing of Massive IoT Data by Edge-Cloud Collaborative Stream Comput-

ing, Submitted to IEEE ICDE 2025. [CCF A 类会议]

申请中专利

- [1] 董玮, 滕依筱, 高艺, 张文照. 一种基于边缘云原生的实时规则引擎构建方法. 申请号: 202310738034.X. 申请日期: 2023 年 6 月 21 日.
- [2] 董玮, 滕依筱, 吕嘉美, 高艺. 一种面向移动区块链的云边端计算存储卸载方法和装置. 申请号: 202311636363.X. 申请日期: 2023 年 12 月 1 日.