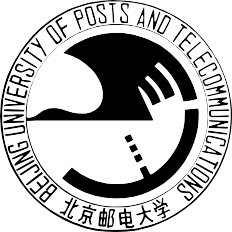


博士学位论文（学术学位）



题目： 基于区块链的边缘分布式计算卸

载关键技术研究

学 号： **2020010357**

#### 姓 名： 吴忠辉 专 业： 计算机科学与技术培养方式： 全日制 导 师： 许长桥 学 院： 计算机学院

**BEIJING UNIVERSITY OF POSTS AND**

**TELECOMMUNICATIONS**

**Doctoral Dissertation**

**Title : Research on Key Technologies of**

**Blockchain-Based Edge Distributed**

**Computing Offloading**

**Student ID: 2020010357**

**Candidate: Zhonghui Wu**

**Subject: Computer Science and Technology**

**Training Mode: Full-Time**

**Supervisor: Changqiao Xu**

**Institute: School of Computing**

# 答辩委员会名单

|  |  |  |  |
| --- | --- | --- | --- |
| 职务 | 姓 名 | 职 称 | 工 作 单 位 |
| 主席 |  |  |  |
| 委员 |  |  |  |
| 委员 |  |  |  |
| 委员 |  |  |  |
| 委员 |  |  |  |
| 秘书 |  |  |  |
| 答辩日期 |  | | |

**独创性（或创新性）声明**

本人声明所呈交的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。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电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了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名： 日期：

**关于论文使用授权的说明**

本人完全了解并同意北京邮电大学有关保留、使用学位论文的规定，即：北京邮电大学拥有以下关于学位论文的无偿使用权，具体包括：学校有权保留并向国家有关部门或机构送交学位论文，有权允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，有权允许采用影印、缩印或其它复制手段保存、汇编学位论文，将学位论文的全部或部分内容编入有关数据库进行检索。（保密的学位论文在解密后遵守此规定）

本人签名： 日期：

导师签名： 日期：

## 基于区块链的边缘分布式计算卸载关键技术研究

## 摘要

#### 在当今信息化、数字化日益深入的时代背景下，智能设备的广泛

普及与互联网技术的不断进步，共同促进了诸多科学技术和新兴产业的蓬勃发展。从社交媒体的实时交互到智慧城市的构建，再到虚拟现实技术的广泛应用，每一项技术进步都充分证明了强大算力的核心作用。高效、低延迟的计算不仅关乎着用户的优质体验，也是推进技术革新与社会发展的关键。然而，随着对算力需求的不断增长， 传统的集中式计算架构逐渐显露出局限性，包括处理延迟长、能源消耗大、带宽需求增加以及数据处理集中等问题，这些问题使得它难以满足现代社会对计算高效率和高安全性的需求。鉴于此，边缘分布式计算引起了广泛关注，它通过在网络边缘部署大量计算设备进行任务分散处理，能够优化数据流程、提升系统效率与数据安全性。

然而，这种计算范式的特性具有多源性、异构性和动态性的特征。多源性体现在设备的来源广泛，从个人的智能设备到企业级服务器皆可成为边缘计算节点；异构性强调边缘设备在计算能力、传输速率等关键性能参数上的差异；动态性反映了网络环境及计算需求与资源供给的快速、持续变化的特点。这些特性使得边缘分布式计算场景下的计算卸载在实际部署和运行中面临以下挑战：（1）边缘设备的异构性强、来源多样化，导致节点的可信度难以确保，处理流程难以追溯，影响到计算任务的安全性与高效性；（2）网络环境的动态多变性，当前的计算卸载策略显得静态且僵化，导致计算卸载效率低下，进而影响系统性能和用户体验；（3）多设备协作的计算场景中，计算卸载的不公平可能导致任务分配不均，造成“强者通吃” 的局面，进而损害节点的积极性，影响系统的生态稳定性；（4）在边缘分布式计算环境中，计算模型及结果的可靠性问题成为关键，它直接关系到整个系统的性能表现。因此，如何将计算任务卸载到可

靠的节点上，从而保证计算的安全性成为一个挑战。

在解决这些挑战的过程中，区块链技术以其去中心化、不可篡改和透明性的特点展现出巨大的潜力。分布式账本技术能够有效地保障边缘分布式计算任务的安全性与可靠性，同时，智能合约的引入有望实现对计算任务的自动化管理。因此，研究人员正积极探索利用区块链技术优化边缘分布式计算的任务调度、增强节点间的信任度，并提高计算结果的可靠性。然而，目前的研究往往偏重于探索区块链在提升系统透明度和可追溯性方面的应用，却忽视了区块链技术本身在性能、可扩展性和能耗等方面的挑战，其性能瓶颈和高能耗问题严重制约了其在边缘分布式计算应用中的广泛部署。

为此，本文围绕基于区块链的边缘分布式计算卸载中的信任基础、弹性卸载、公平化调度及安全性保障关键技术，开展了研究，旨在提升计算效率、确保计算安全、优化用户体验。具体包括：

1. 第三章围绕信任问题，提出多链协同的可信计算卸载模型， 实现了计算全流程的数据留痕，而且构建了多链协作的计算卸载方法，并规范了交互流程，为后续研究提供了坚实的可信基础。
2. 第四章考虑网络环境的异构和时变特性，设计了智能合约驱动的自适应卸载策略，将任务分配问题转化为随机优化问题，从而在降低总成本和提高任务完成率方面实现了长期优化。通过部署奖励和惩罚机制的智能合约，既激励了工人的贡献，又保证了任务的顺利完成，极大提升了系统的鲁棒性和调度效率。
3. 围绕新兴应用服务场景下的计算卸载公平性问题，第五章提出了一种基于工作量证明的公平化卸载方法，通过探索子任务的强耦合关系，构建了通用任务有向无环图，设计了工作量证明驱动的计算卸载策略，实现了基于多智能体强化学习的动态挖掘策略调整机制，确保了计算任务的公平分配和低时延服务。
4. 针对分布式计算场景中计算模型的安全可靠性问题，第六章设计了一种基于链上信息的安全卸载方案。提出了损失感知的信用评估机制和基于洗牌的隔离机制，并通过动态评分及隔离机制， 实

现了短时间内对恶意客户端的有效识别，提高了系统的安全可靠性。本文以边缘分布式场景下的计算卸载为主要研究用例，以区块

链为代表的新兴科学技术作为主要研究手段，从可信、弹性、公平性、安全四个方面开展了全面的分析与研究。研究方法涵盖了问题引入、数学建模、算法设计、性能分析和实验验证等环节，增强了研究的科学性和准确性。本文所取得的成果对未来边缘分布式计算的研究与发展具有一定的借鉴意义。

关键词：区块链 边缘计算 计算卸载 强化学习

# RESEARCH ON KEY TECHNOLOGIES OF BLOCKCHAIN-BASED EDGE DISTRIBUTED

# COMPUTING OFFLOADING

# ABSTRACT

#### In an era marked by increasingly deepening informatization and dig- italization, the widespread popularity of smart devices and the continuous progress of Internet technology have jointly promoted the vigorous devel- opment of many science technology and emerging industries. These ad- vancements range from real-time social media interactions to the construc- tion of intelligent urban environments and the broad implementation of vir- tual reality technologies, each underscoring the pivotal importance of ro- bust computing power. The demand for efficient, low-latency computing is not merely vital for ensuring a superior user experience but also plays a critical role in fostering technological innovation and societal progress. Nonetheless, the escalating need for computational resources is revealing the limitations of traditional centralized computing architectures, includ- ing protracted processing times, heightened energy consumption, increased bandwidth requirements, and centralized data handling, which collectively make it struggle to meet contemporary societal requirements for efficiency and security. Consequently, edge distributed computing has emerged as a focal area of interest, advocating for the strategic placement of numerous computing devices at the network’s edge to facilitate distributed task pro- cessing, thereby aiming to refine data workflows, boost system efficiency, and enhance data security.

However, this computing paradigm is characterized by its multi-sourced, heterogeneous, and dynamic nature. ”Multi-sourcedness” denotes the di- verse origins of devices, ranging from personal smart devices to enterprise- level servers, all potential nodes for edge computing. ”Heterogeneity” highlights the variances in critical performance metrics, such as computing

power and transmission speeds among edge devices. ”Dynamism” captures the swift and ongoing shifts in network conditions, computing demands, and resource availability. These traits introduce several challenges to com- puting offloading within edge distributed computing settings, including:

1. The pronounced heterogeneity and varied origins of edge devices com- plicate the assurance of node trustworthiness and the tracking of processing flows, thereby impacting the security and efficiency of computing tasks.
2. The dynamic and fluctuating nature of network environments renders existing computing offloading strategies as static and inflexible, leading to suboptimal computing offloading, which in turn, detracts from system performance and user experience. (3) In scenarios involving collaborative computing across multiple devices, the inequity of computing offloading may lead to disproportionate task distribution, resulting in the ”strong take all” situation, which diminishs nodes’ willingness to participate and fur- ther affects the ecological stability of the system. (4) Within distributed computing environments, the dependability of computing models and out- comes is paramount, directly influencing the performance of the entire sys- tem. Therefore, how to offload computing tasks to reliable nodes to ensure the security of computing becomes a challenge.

Addressing these challenges, blockchain technology, with its decen- tralized, immutable, and transparent attributes, has shown great potential. The application of distributed ledger technology can effectively safeguard the security and dependability of edge distributed computing tasks, while smart contracts offer a pathway to automate the management of comput- ing tasks. Researchers are actively exploring the integration of blockchain technology to optimize task scheduling in edge distributed computing, fos- ter node trust, and bolster the reliability of computing outcomes. How- ever, much of the existing research primarily concentrates on harnessing blockchain to enhance system transparency and traceability, somewhat ne- glecting the challenges blockchain technology faces regarding performance, scalability, and energy consumption. These constraints severely limit its

broad applicability in edge distributed computing scenarios.

This thesis delves into essential technologies such as trust mecha- nisms, elastic offloading, equitable scheduling, and security enhancement in blockchain-based edge distributed computing offloading. It aims to aug- ment computing efficiency, ensure data security, and elevate the user ex- perience. The research includes:

* 1. To address the issue of trust mechanism, Chapter 3 proposes Multi- chain Collaborative Trusted Computing Offloading Model. It achieves data traceability throughout the computing process, establishes a multi-chain co- operative task scheduling mechanism, and standards the interaction work- flows, laying a robust trust foundation for further research.
  2. Considering the heterogeneous and time-varying characteristics of network environments, Chapter 4 designs Smart Contract-driven Adap- tive Offloading strategy that transforms the task allocation problem into a stochastic optimization problem, thereby achieving long-term optimiza- tion in reducing total costs and improving task completion rates. The de- ployment of smart contracts with reward and penalty mechanisms not only incentivizes worker contribution but also ensures task completion, signifi- cantly enhancing system robustness and scheduling efficiency.
  3. For the fairness issue of computing offloading in emerging appli- cation service scenarios, Chapter 5 proposes Proof of Work-based Fair Of- floading Method. It explores the strong coupling relationships of subtasks, constructs a general task directed acyclic graph, and designs a workload proof-driven computing offloading strategy, along with a dynamic mining strategy adjustment mechanism based on multi-agent reinforcement learn- ing, ensuring fair task distribution and low-latency service.
  4. Considering the security issue of computing models in distributed computing scenarios, Chapter 6 develops Secure Offloading Scheme based on On-chain Information. A loss-aware credit evaluation mechanism and a shuffling-based isolation mechanism are proposed. Through dynamic scor- ing and isolation mechanism, it effectively identifies malicious clients in

the short term, thereby enhancing system security and reliability.

This thesis, focusing on computing offloading in edge distributed sce- narios with blockchain as the main research tool, conducts a comprehen- sive analysis and research from the aspects of trustworthiness, elasticity, fairness, and security. The research methodology encompasses problem delineation, mathematical modeling, algorithmic design, performance as- sessment, and experimental validation, thereby bolstering scientificity and rigor of research. The achievements of the thesis have certain reference significance for the future research and development of edge distributed computing.

**KEY WORDS:** Blockchain Edge computing Computing Offloading Reinforcement learning

目 录

[摘 要](#_bookmark0) I

[ABSTRACT](#_bookmark1) V

[第一章 绪论](#_bookmark2) 1

* 1. [论文研究背景](#_bookmark3) 1
     1. [边缘分布式计算发展概述](#_bookmark4) 1
     2. [边缘分布式计算卸载挑战](#_bookmark10) 2
     3. [区块链在边缘分布式计算卸载下的应用潜力](#_bookmark12) 3
  2. [论文选题依据](#_bookmark13) 4
  3. [主要研究内容](#_bookmark17) 5
  4. [主要贡献和创新](#_bookmark18) 7
  5. [论文组织架构](#_bookmark19) 8

[第二章 相关工作综述](#_bookmark21) 11

* 1. [边缘分布式计算概述](#_bookmark22) 11
     1. [边缘分布式计算场景](#_bookmark23) 11
     2. [边缘分布式计算卸载](#_bookmark25) 13
  2. [相关研究现状](#_bookmark26) 14
     1. [边缘分布式计算可信卸载相关研究](#_bookmark27) 14
     2. [边缘分布式计算高效卸载相关研究](#_bookmark28) 15
     3. [边缘分布式计算安全卸载相关研究](#_bookmark29) 17
     4. [区块链赋能的计算卸载相关研究](#_bookmark30) 17
  3. [本章小结](#_bookmark31) 18

[第三章 多链协同的可信计算卸载模型](#_bookmark32) 21

[3.1 引言](#_bookmark33) 21

* 1. [基本系统模型](#_bookmark34) 22
  2. [基于多链的区块链架构设计](#_bookmark36) 23
     1. [架构基本介绍](#_bookmark37) 24
     2. [基于链的计算卸载交互流程](#_bookmark38) 24
  3. [计算卸载算法设计](#_bookmark41) 27
  4. [实验与性能分析](#_bookmark42) 27
  5. [本章小结](#_bookmark47) 30

[第四章 智能合约驱动的自适应卸载策略](#_bookmark48) 31

[4.1 引言](#_bookmark49) 31

* 1. [系统模型概述](#_bookmark51) 33
  2. [区块链和智能合约设计](#_bookmark52) 34
     1. [区块链结构设计](#_bookmark53) 34
     2. [智能合约设计](#_bookmark54) 35
  3. [自适应卸载策略](#_bookmark55) 35
     1. [问题建模](#_bookmark56) 35
     2. [算法设计](#_bookmark58) 38
     3. [奖励/惩罚机制设计](#_bookmark60) 38
  4. [理论分析](#_bookmark64) 40
  5. [实验与性能分析](#_bookmark68) 42
  6. [本章小结](#_bookmark78) 48

[第五章 基于工作量证明的公平化卸载方法](#_bookmark79) 49

[5.1 引言](#_bookmark80) 49

* 1. [系统架构和网络模型](#_bookmark82) 52
     1. [系统概述](#_bookmark83) 52
     2. [网络模型](#_bookmark86) 55
  2. [区块链架构设计](#_bookmark88) 56
     1. [基于工作量证明的区块链设计](#_bookmark89) 56
     2. [安全性分析](#_bookmark92) 59
  3. [优化问题建模](#_bookmark94) 61
  4. [任务调度算法设计](#_bookmark96) 62
     1. [学习问题的表述](#_bookmark97) 62
     2. [分散式多智能体强化学习算法](#_bookmark99) 64
  5. [实验与性能分析](#_bookmark102) 67
     1. [实验设置](#_bookmark103) 67
     2. [对比方案](#_bookmark106) 68
     3. [实验结果](#_bookmark107) 68
  6. [本章小结](#_bookmark114) 72

[第六章 基于链上信息的安全卸载方案](#_bookmark115) 73

[6.1 引言](#_bookmark116) 73

* 1. [系统模型和攻击模型](#_bookmark117) 75
     1. [系统模型](#_bookmark118) 75
     2. [攻击模型](#_bookmark120) 77
     3. [基于区块链的声誉机制](#_bookmark121) 77
  2. [恶意客户端检测策略](#_bookmark124) 79
     1. [基于损失的信用评估](#_bookmark125) 79
     2. [基于洗牌的隔离机制](#_bookmark130) 81
     3. [理论分析](#_bookmark132) 83
     4. [拜占庭容错聚合](#_bookmark134) 85
  3. [实验与性能分析](#_bookmark135) 85
     1. [实验设置](#_bookmark136) 85
     2. [结果和分析](#_bookmark139) 87
  4. [本章小结](#_bookmark144) 90

[第七章 结束语](#_bookmark145) 91

* 1. [论文总结](#_bookmark146) 91
  2. [未来研究工作](#_bookmark147) 93

[参考文献](#_bookmark148) 95

[附录 **A** 第四章中相关定理证明](#_bookmark284) 105

* 1. [定理 4.1 的证明](#_bookmark285) 105
  2. [定理 4.2 的证明](#_bookmark287) 105
  3. [定理 4.3 的证明](#_bookmark289) 106

[附录 **B** 缩略语表](#_bookmark290) 107

[致 谢](#_bookmark291) 109

[攻读学位期间取得的“创新成果”目录](#_bookmark292) 111

表格索引

[5-1 符号](#_bookmark84) 53

[5-2 章节五实验参数](#_bookmark105) 68

[6-1 章节六实验参数](#_bookmark138) 86

## 插图索引

[1-1 论文组织架构及各研究点的逻辑关系](#_bookmark20) 9

[2-1 边缘分布式计算网络示意图](#_bookmark24) 12

* 1. [系统架构总览](#_bookmark35) 23
  2. [MCTM 中的智能合约设计](#_bookmark39) 25
  3. [MCTM 工作流程](#_bookmark40) 26
  4. [收敛性](#_bookmark43) 29
  5. [不同恶意节点比例下的服务质量](#_bookmark44) 29
  6. [以太坊 gas 开销性能](#_bookmark45) 29
  7. [不同总任务数下的服务质量](#_bookmark46) 30
  8. [系统总览](#_bookmark50) 33
  9. [基于智能合约的计算卸载流程](#_bookmark57) 36
  10. [总成本随仿真时间变化](#_bookmark69) 43
  11. [总队列长度随仿真时间变化](#_bookmark70) 43
  12. [不同比例的不成实节点下奖励与押金变化情况](#_bookmark71) 44
  13. [不同移动速度范围下奖励与押金变化情况](#_bookmark72) 44

[4-7 不同请求到达率下（λ = 50*,* 100*,* 150*,* 200），四种解决方案的总成本](#_bookmark73)45

* 1. [PoW 共识下即时吞吐量与时间的关系](#_bookmark74) 46
  2. [DPoS 共识下即时吞吐量 TPS 与时间的关系](#_bookmark75) 46
  3. [区块批准时延与区块数的关系](#_bookmark76) 47
  4. [累计吞吐量与时间的关系](#_bookmark77) 47

[5-1 具有依赖性任务的边缘分布式计算卸载示意图。上半部分展示了具有依赖性特征的任务，它可以分为多个子任务。下半部分表示边](#_bookmark81)

[缘网络。其中，蓝色线表示数据流，红色线表示计算任务卸载过程。](#_bookmark81)50 [5-2 系统网络架构](#_bookmark85) 54

* 1. [任务链以及 DAG 区块链设计](#_bookmark87) 56
  2. [区块链扩展流程图](#_bookmark90) 58
  3. [DMA 算法结构示意图](#_bookmark100) 64
  4. [实验环境](#_bookmark104) 67
  5. [不同 NCP 数（6, 9, 12, 15）下的奖励值收敛性](#_bookmark108) 69
  6. [不同子任务数（6, 9, 12）下的奖励值收敛性](#_bookmark109) 69
  7. [不同结构的任务链奖励值的收敛情况](#_bookmark110) 70
  8. [不同 NCP 数量下的时间消耗对比](#_bookmark111) 71
  9. [不同不可靠节点比例下的时间消耗对比](#_bookmark112) 71
  10. [不同 NCP 数量下的 Jain 公平性评估](#_bookmark113) 72
  11. [系统模型](#_bookmark119) 76
  12. [不同攻击下分布式 DNN 学习的损失表现（MSE）](#_bookmark126) 80
  13. [在不同模型和数据集下的损失性能](#_bookmark140) 87
  14. [不同恶意客户端比例下的 MSE](#_bookmark141) 88
  15. [不同客户端数量下的 MSE](#_bookmark142) 89
  16. [不同攻击方式下的 MSE 情况](#_bookmark143) 90

# 第一章 绪论

#### 论文研究背景

* + 1. 边缘分布式计算发展概述

在当今信息时代，人们正见证着数字化应用的爆炸式增长。从 4K/8K 高清视频[[1](#_bookmark149)] 到虚拟现实（Virtual Reality，VR）/ 增强现实（Extended Reality，XR）体验[[2](#_bookmark150)]，再到远程医疗[[3](#_bookmark151)] 和元宇宙[[4](#_bookmark152)] 的探索，新型应用的涌现不仅极大丰富了人们的生活，也促进了数据流量的剧增。根据诺基亚发布的《2030 年全球网络流量》（Global Network Traffic 2030）[①](#_bookmark5)，全球网络流量预计将从 2022 年的每月507EB 增加到 2026 年的每月 1014EB 和 2030 年的每月 2443EB。为处理庞大的数据以及充分挖掘、利用数据的价值，全球对算力需求亦呈现爆发式增长。在华为发布的《泛在算力：智能社会的基石》[②](#_bookmark6)中，预计到 2030 年，人工智能对计算能力的需求将增长 390 倍，区块链将增长 2000 倍，虚拟现实市场将增长 300 倍。工业和信息化部等六部门联合印发《算力基础设施高质量发展行动计划》[③](#_bookmark7)指 出：算力已经成为智慧社会的重要基石。如何健全算力网、解决算力难、用好算力券、充分激活算力脑，成为当下数字时代产业经济发展的关键。

过去的十多年间，基于云的计算卸载允许用户将计算密集型的任务卸载到资源丰富且计算能力强大的远程云数据中心，实现数据从边缘到云中心再到边缘的流转。而面对上述数据流量和算力需求激增的趋势，传统云计算因其集中化架构面临多重问题：数据处理延迟高，能源消耗大，算力提升与算力需求增长匹配失衡等。如：4K 超高清直播需要超低时延[④](#_bookmark8)，给核心网的带宽带来巨大压力； 深度学习模型的层数和参数数量持续增长导致了对计算资源的需求呈指数级增长[⑤](#_bookmark9)，而云计算能力的提升速率难以匹配。此外，由于数据需要经过远距离传输， 在远离用户侧计算、存储，因此。隐私保护与数据安全方面也存在风险。在此背景下，随着物联网设备的激增和第五代移动通信技术（Fifth-Generation Mobile

①  <https://onestore.nokia.com/asset/213660?_ga=2.158548031.402744148.1698805358-591053280.1697698055>

② [https://www-file.huawei.com/-/media/corporate/pdf/public-policy/ubiquitous\_computing\_power\_the\_](https://www-file.huawei.com/-/media/corporate/pdf/public-policy/ubiquitous_computing_power_the_cornerstone_intelligent_society_en.pdf?la=it-it) [cornerstone\_intelligent\_society\_en.pdf?la=it-it](https://www-file.huawei.com/-/media/corporate/pdf/public-policy/ubiquitous_computing_power_the_cornerstone_intelligent_society_en.pdf?la=it-it)

③ <https://www.gov.cn/zhengce/zhengceku/202310/P020231009520949915888.pdf>

④ <https://www-file.huawei.com/-/media/corporate/pdf/news/5g-4k-sla-whitepaper.pdf>

⑤ <https://roadshow.h3c.com/zl/pdf/2023zhinengsuanli.pdf>

Communication Technology，5G）通信技术的发展，云计算能力下沉至网络边缘， 边端设备的算力被充分挖掘。边缘分布式计算，作为一种新兴的计算范式，被视为缓解当前网络压力、提升计算效率、保障用户体验质量的有效手段。这种模式允许数据在产生地点附近的多个节点上进行分散处理，而不是依赖单一的边缘节点或远程云中心，从而进一步减少了延迟，提高了计算效率和数据处理的可靠性。根据大视野研究有限公司（Grand View Research, Inc.）的最新报告[⑥](#_bookmark11)， 2023 年全球边缘分布式计算市场规模价值 164.5 亿美元，预计 2030 年，其市场规模预计将达到 1559 亿美元，年增长率高达 37.9%。

在数字化浪潮推动下，我们站在一个新的历史起点。面对未来超大规模数据流量、超密集算力和超低时延服务的需求，边缘分布式计算技术正承载着新时代的梦想和挑战，不仅是技术进步的产物，更是推动社会向更智能、更互联的新纪元迈进的重要力量。它代表了对于更加高效、安全和可持续计算模式的探索和实践，是未来智能化社会基础设施建设的关键一环。

* + 1. 边缘分布式计算卸载挑战

在边缘分布式计算中，计算卸载是一项关键核心技术，它的主要目的是在不同的计算任务中选择最合适的计算处理节点，以满足低延迟和高可靠性的需求。通过合理的卸载策略，可以有效利用边缘网络的计算资源，从而优化整个系统的性能和用户体验。然而，边缘分布式计算具有以下三大显著特征：多源性、异构性和动态性。多源性体现在设备来源的广泛性，从个人的智能设备到企业级服务器，皆可成为边缘计算节点，为计算卸载提供了丰富的资源。异构性则指出边缘设备在计算能力、存储容量和传输速率等方面的差异性，这要求计算卸载策略能够适应各种设备的特点，以实现资源的最优配置。动态性反映了网络环境的复杂多变以及计算需求与资源供给的快速变化，这使得计算卸载策略需要具备高度的适应性和灵活性。因此，边缘分布式计算卸载在实际部署与运行过程将面临以下四个主要挑战：

* + - 1. 节点的可信度难以保证：边缘分布式计算涉及的算力来源广泛且分散， 涵盖了属于不同组织和个人的设备，难以对每个计算节点的安全性和信任度进行充分验证。节点可信性难以保障。同时，这些节点的计算过程和中间数据往往没有有效的记录机制，导致计算过程缺乏透明度和可追溯性。此外，当前边缘分布式计算系统中的决策往往依赖于中央决策器，这不仅增加了系统的单点

⑥ <https://www.grandviewresearch.com/industry-analysis/edge-computing-market>

失效风险，还可能成为攻击者的目标，一旦被侵害，整个系统的可用性和鲁棒性都将受到严重影响。因此，确保边缘节点的可信性，从而保障计算任务在可信任的环境下执行成为了边缘分布式计算卸载面临的一大难点。

* + - 1. 计算卸载策略的僵化：边缘分布式计算环境中的计算设备的计算能力、存储容量以及网络连接状态等各不相同，且网络状态和计算需求具有高度的动态性。例如，某些设备可能仅在特定时间段内可用，或其性能随时间和使用情况而波动；设备之间地理分布分散，它们之间的网络连接往往不稳定，这进一步增加了计算任务高效卸载的难度。当前的计算卸载策略往往显得静态且僵化，难以快速适应环境的变化，导致计算卸载效率低下。不仅延长了计算任务的处理时间，还可能导致计算资源浪费，进而影响整体的计算效率和用户体验。因此， 为了充分发挥边缘分布式计算的潜力，亟需开展计算任务弹性卸载方案的研究。
      2. 计算卸载的不公平性：技术进步与各类应用涌现使得计算任务日益复杂化，数据处理流程往往可以被细化为众多相互耦合、依存的子任务。在此场景下，多设备协同串行/并行工作是常见的计算模式。然而，任务卸载过程中的不 公平性可能导致任务分配不均，某些节点可能承担过多的计算负载，而另一些节点则处于闲置状态。这种不平衡的资源利用不仅降低了计算效率，还会造成了“强者通吃”局面（任务奖励将集中于强算力节点，弱算力节点无法提供自身算力，从而无法获得奖励），损害节点参与者的积极性，影响系统的生态稳定性和可持续发展。因此需计算卸载方法应优化系统计算效率的同时，公平分配计算任务，确保各类设备按能力贡献计算力，实现边缘分布式计算环境的高效公平。
      3. 计算卸载的可靠性问题：随边缘分布式计算技术在大模型训练、智慧城市等关键领域的深入应用，确保计算模型和计算结果的可靠性至关重要。由于计算任务在多个节点上分布式执行，每个节点的计算结果都需要被汇总并验证。恶意节点可以通过提供错误的计算结果或篡改数据，对整个计算过程的准确性和可靠性构成威胁。因此，开发有效的机制以识别恶意行为，将计算任务卸载到安全的节点上，并确保计算模型及结果的准确无误，成为了边缘分布式计算卸载面临的一个关键挑战。
    1. 区块链在边缘分布式计算卸载下的应用潜力

受到密码学和分布式技术的启发，自 2008 年比特币的诞生以来，区块链技术逐渐从一个支持加密货币的底层技术发展成为一种能够提供透明、不可篡改、去中心化信任的全新计算模型。在这个模型中，每一笔交易或数据都会被加密并永久记录在一个公开的分布式账本上，这个账本被网络中的每一个节点共同

维护。正是这些特性，让区块链成为了解决边缘分布式计算卸载中面临的可信、安全等挑战的有力工具。

边缘分布式计算卸载过程中，区块链的应用潜力主要体现在以下几个方面： 数据审计化：区块链上的每一笔交易都是不可变的，因为数据块是用哈希和一对密钥链接起来的，并记录有时间戳，这为交易提供了更好的可审计性。一个可追溯、不可篡改的数据记录使得系统参与节点能够验证服务的执行情况，从而增强系统透明度和可信度。公开透明化：通过智能合约，区块链可以自动化地执行合约条款，实现资源和任务在边缘节点间的自动、透明和公平分配。这不仅提高了资源利用率，还降低了管理成本和潜在的人为错误。去中心化：区块链上的所有参与者都存储了数据库文件的副本。它不再需要一个可信的中央机构来验证每笔交易。同时，智能合约的诞生也为去中心化的决策方案提供了突破口。

然而，尽管区块链作为一种革命性技术，在边缘分布式计算卸载领域展现出巨大应用潜力，但其在直接应用于边缘分布式计算时仍面临一系列挑战。现有研究往往集中于探索区块链在系统透明度和可追溯性提升方面的应用，而忽略了区块链技术本身存在的性能、可扩展性和能耗问题。区块链技术在处理大规模交易时遇到的性能瓶颈问题，尤其是低吞吐量和高延迟问题，对于需要迅速处理海量数据的边缘分布式计算场景可能构成严重的制约。此外，区块链网络的可扩展性问题亦限制了其在广泛的边缘计算环境中的应用。能耗问题则是另一个关键挑战，特别是在采用工作量证明（Proof of Work，PoW）等共识机制的区块链中，高能耗计算对环境造成的影响及其运营成本是不能忽视的问题。因此，将区块链技术应用于边缘分布式计算以优化计算任务卸载的同时，必须全面评估并解决这些技术挑战。

#### 论文选题依据

边缘分布式计算领域是近年来学术界和工业界的研究热点，也被列为我国的重点发展领域。国内的《“十四五”国家信息化规划》[⑦](#_bookmark14)和《“十四五”数字经济发展规划》[⑧](#_bookmark15)都强调了提升特定场景下的边缘计算能力、算力统筹及智能调度的重要性，旨在实现低时延、高可靠、强安全的边缘计算服务。同时，美国也将边缘分布式计算纳入了《关键和新兴技术清单（2024）》（Critical and Emerging Technologies List Update 2024）[⑨](#_bookmark16)。这些政策的实施彰显了边缘分布式计算在构

⑦ <https://www.gov.cn/xinwen/2021-12/28/5664873/files/1760823a103e4d75ac681564fe481af4.pdf>

⑧  <https://www.gov.cn/zhengce/content/2022-01/12/content_5667817.htm>

⑨ <https://www.gov.cn/zhengce/content/2022-01/12/content_5667817.htm>

建未来信息社会中的核心地位。此外，区块链也是国家级战略技术之一，《十四个五年规划和 2035 年远景目标纲要》中提出推动智能合约、共识算法、加密算法、分布式系统等区块链技术创新，强调了区块链等新兴技术在未来社会经济发展中的重要作用。这一战略部署指出了区块链在加强数据安全性、提高交易效率和促进技术融合方面的潜力，特别是在与边缘分布式计算结合时，可望实现更加可信、高效、安全的计算环境。

本文的选题依托于依托于国家自然科学基金杰出青年科学基金项目“异构网络融合的传输理论与方法”（项目编号：62225105）、以及十三五预研项目“区块链军事技术应用”（项目编号：JZX6Y202001010155）等。其中，杰出青年科学基金的主要工作之一是研究自主协作的多维资源调度机制，针对多业态异构网络下网络、计算、存储多维资源协作问题，探索机器学习等技术在异构网络资源调配中的应用，做出智能计算卸载决策。十三五预研项目主要面向边缘无人机群协作场景下，通过区块链技术构建可信、高效的通信平台。

结合上述项目的主要研究目标，本文对边缘分布式计算卸载进行了深入的探讨和研究，旨在解决边缘分布式计算卸载在实际部署中遇到的可信、弹性、公平性、安全等方面的问题，完成了包括区块链增强的可信卸载模型、自适应卸载策略、公平化卸载方法、安全卸载方案等一系列相关研究内容。

综上所述，本文的研究响应了国家对新型基础设施建设和数字经济发展的战略需求，针对边缘分布式计算卸载领域中存在的关键技术问题，提出了创新性的解决方案，旨在为未来高质量、低时延的边缘服务提供理论基础和技术支撑，进而推动我国边缘计算技术的发展和应用，为国家信息化建设和数字化转型贡献力量。

#### 主要研究内容

本文围绕边缘分布式计算卸载领域内存在的挑战，特别是在引入区块链技术以提高系统的可信度、弹性、公平性及安全性方面展开深入研究，从而提升计算效率、保障计算安全、优化用户体验。

研究的主要内容包括以下几个方面：（1）围绕边缘节点的可信性难以保证，处理流程难以追溯问题，提出了“多链协同的可信计算卸载模型”。建立了去中心化的计算卸载模型，利用区块链的不可篡改性和透明性特征，增强了边缘节点的可信度，保障计算任务的可靠执行。（2）针对通信环境动态多变，计算卸载策略静态僵化问题，提出了“智能合约驱动的自适应卸载策略”。考虑资源动态特

征，设计了基于李亚普诺夫的在线决策方法，采用智能合约自动执行任务分配和奖励发放，优化任务调度的效率。（3）针对计算卸载公平性问题，提出了“基于工作量证明的公平化卸载方法”。利用子任务之间的耦合特性，引入PoW 和基于多智能体强化学习的动态策略调整，确保了计算任务在各个节点之间的公平分配，增强节点参与积极性，减少了服务时延。（4）应对分布式计算场景中计算模型和结果可靠性问题，提出了“基于链上信息的安全卸载方案”。基于历史计算表现，设计评估机制和隔离机制，提高系统对恶意行为的识别和防御能力。

具体研究内容如下：

1. 信任基础：边缘分布式计算应用场景下，由于计算资源的来源广泛且分散， 节点的可信性成为一个难以保障的问题。特别是在没有可靠记录机制的环境下，计算过程与中间数据的透明度和可追溯性显得尤其欠缺。此外，现有的边缘分布式计算系统多依赖中心化决策机制，这种架构下系统的可用性和鲁棒性容易遭受攻击。针对上述问题，本文提出了一种多链协同的可信计算卸载模型。首先通过构建一个多链协同工作的任务调度机制，实现了在不同边缘计算节点间的高效计算任务分配. 然后依托分布式账本特性， 确保了数据的完整性与可靠性。同时，对计算任务的执行和验证流程进行了规范化管理，显著提升了边缘计算节点的可信度。该模型采用去中心化的设计，有效减少了对单一中心决策实体的依赖，降低了系统遭受攻击的可能性，从而显著增强了系统的整体鲁棒性，为后续研究构建可信基础。
2. 弹性卸载：在边缘计算领域，边缘设备的庞大规模和资源供需的高度动态性，给计算任务的卸载带来了显著的不确定性。此外，由于缺乏公开且透明的激励机制，资源提供者参与的积极性不足，容易导致资源供应受限， 进而影响了整体的计算效率。为了解决这些问题，本文设计了一种智能合约驱动的自适应卸载策略。该策略将任务分配问题构建为一个随机优化问题，并提出了一种基于李亚普诺夫优化理论的在线决策方法，旨在从长期角度出发，优化系统的总成本及任务完成率。同时，通过部署智能合约实现的奖励与惩罚机制，有效提升了系统的鲁棒性和任务调度的效率。
3. 公平化调度：随着计算任务的日益复杂化，计算流程常常被细分为多个相互依赖且耦合的子任务，这些子任务在计算模式上展现出了串行性性与并行性的不同需求。当这些复杂任务被卸载至边缘计算环境中的众多设备时， 如何确保任务分配的公平性与效率性便显得尤为关键。针对此问题，本文

提出了一种基于工作量证明的公平化卸载方法。本方法构建了一个通用的有向无环图（Directed Acyclic Graph，DAG）任务模型，将计算卸载和区块链“挖矿”相结合，旨在实现公平的计算资源卸载。此外，本文还开发了一种基于多智能体强化学习的动态策略调整机制，以适应设备性能的多样性，确保不同性能的设备能根据其实时计算能力贡献相应的价值，从而在边缘分布式计算环境中实现高效与公平的并存。

1. 安全性保障：在分布式深度学习应用领域，特别是在对准确性和可靠性要求极高的场景下，恶意节点通过故意提供错误的计算结果或在计算过程中篡改数据，构成了对系统计算准确性和可靠性的重大安全威胁。针对上述挑战，本文提出了基于链上信息的安全卸载方案。此机制首先利用损失感知型信用评估法对各计算节点进行动态评分，并不断更新其信用值。并将相应数据里记录上链。基于此，进一步设计了一种基于洗牌策略的隔离机制，有效地实现了对恶意节点的快速识别与隔离。此外，为了最大程度地降低恶意客户端的潜在影响，本系统还引入了拜占庭容错机制进行增强。通过一系列实验验证，本方案在多种攻击场景下展现了出色的性能，保障了计算卸载的安全。

#### 主要贡献和创新

本文深入探讨了边缘分布式计算卸载的一些关键理论及技术，提出了一系列创新性解决方案。主要贡献和创新点可总结如下：

1. 提出了多链协同的可信计算卸载模型：本研究提出了一种多链协同的可信计算卸载模型，旨在强化边缘分布式计算环境中节点的信任度。通过结合多链协同技术和分布式账本特性实现了计算任务的高效分配和透明化管理，提升了计算效率和系统的鲁棒性。此外，采用的去中心化架构设计显著降低了系统遭受攻击的可能性，确保了数据完整性和可靠性。这一工作为后续的研究开展提供了一个互信的可靠环境。
2. 设计了智能合约驱动的自适应卸载策略：本研究通过设计一种智能合约驱动的自适应卸载策略，针对性应对计算环境与资源供需的高度动态性所带来的不确定性等问题。该策略通过构建随机优化问题并应用李亚普诺夫优化理论，实现了长期成本和任务完成率的优化。利用智能合约实施的奖励与惩罚机制显著提高了资源提供者的积极性，从而增强了系统的鲁棒性和

任务调度效率。这一研究不仅解决了资源供应不稳定的问题，还为边缘分布式计算任务卸载和资源管理开辟了新的高效可靠路径。

1. 提出了基于工作量证明的公平化卸载方法: 本研究提出了一种基于工作量证明的公平化卸载方法。本研究在确保边缘计算环境中的计算任务公平分配方面取得了显著进步。创新性地结合了DAG 任务模型与工作量证明机制，确保了不同性能设备按其实际计算能力公平贡献。同时，通过多智能体强化学习动态调整策略，适应设备自身性能的变化。这不仅提高了任务调度的公平性和效率，也为复杂计算任务在边缘环境中的高效执行提供了新的解决路径。
2. 构建了基于链上信息的安全卸载方案: 本研究构建了一种基于链上信息的安全卸载方案，该方案通过链上信息进行支持。利用损失感知型信用评估法动态评分计算节点并更新信用值，结合洗牌策略的隔离机制，快速有效地识别和隔离恶意节点。引入的拜占庭容错机制进一步增强了系统在面对攻击时的韧性，确保了计算卸载的安全性和可靠性。这一方案的实施，对于提高边缘分布式深计算的安全防护能力具有重要意义。

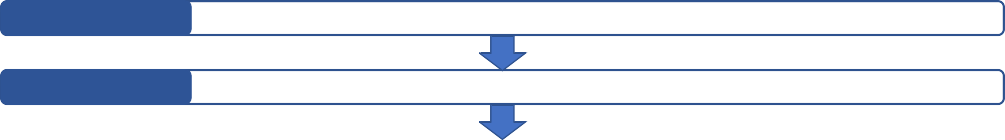
通过上述研究，本文不仅针对边缘分布式计算卸载领域的关键技术挑战提出了有效的解决方案，而且还通过结合区块链技术，为实现可信、高效、公平和安全的计算卸载提供了新的思路和方法。这些成果对于推动边缘分布式计算和区块链技术的深入融合与应用，具有重要的理论和实践价值。

#### 论文组织架构

论文针对边缘分布式计算卸载中所面临的一系列关键问题进行研究，以提高计算效率，保障计算安全、提升用户体验为目标，分别从可信基础、弹性卸载、公平化调度以及安全性保障四个角度入手，提出了一套行之有效的解决方案，为边缘分布式计算卸载提供了理论和技术支撑。本文的组织架构及各研究点的逻辑关系如图[1-1](#_bookmark20)所示：

第一章首先对论文的研究背景进行总体论述，总结出当前边缘分布式计算卸载所存在的挑战。随后阐述了论文选题依据，其次，分别介绍了本文主要研究内容、贡献和创新点。最后说明了论文的组织架构。

第二章系统地描述了边缘分布式计算卸载和区块链领域的相关工作。首先， 介绍了边缘分布式计算场景以及计算卸载的相关概念。随后分别从可信卸载、



**基于区块链的边缘分布式计算卸载关键技术研究**

**第一章**

**绪论**

**第二章**

**相关工作综述**

**现状特征**

**多源性**

**异构性**

**动态性**

**问题挑战**

**可信**

**弹性**

**公平性**

**安全**

**研究内容**

**第三章**

**多链协同的可信计算卸载模型**

**第四章**

**智能合约驱动的自适应卸载策略**

**第五章**

**基于工作量证明的公平化卸载方法**

**第六章**

**基于链上信息的安全卸载方案**

（基础）

（支撑）

构建可信基础 提高卸载效率 保障计算安全

**第七章**

**总结与展望**

图 1-1 论文组织架构及各研究点的逻辑关系

高效卸载、安全卸载三个层面讨论了相关研究。最后本章对区块链技术在计算卸载应用中的相关研究进行了详细介绍。

第三章围绕计算卸载的可信问题，提出了多链协同的可信计算卸载模型。首先以典型的边缘分布式计算场景为例，讨论了潜在的问题。随后，设计多链并行的模型架构，对全流程进行可靠的数据留痕并规范了交互流程。结果表明该方案可以有效提高节点可信度，在提供高质量服务方面具有更好的性能。该章节内容为后续章节提供了可信环境，是研究开展的基础。

第四章针对动态资源适配导致的计算效率问题，提出智能合约驱动的自适应卸载策略。首先将计算任务分配问题建模成一个随机优化问题，在此基础上提出基于李雅普诺夫的在线决策方法，以实时适应资源供给侧波动。此外，利用智能合约将奖励分配机制公开化，执行透明化，有效地激励了计算节点的参与度。最后从理论以及实验上证明了算法的收敛性以及方案的有效性。

第五章面向日趋复杂的具有依赖性、强耦合性特征的边缘计算业务，提出了基于工作量证明的公平化卸载方法。首先建立了一个通用DAG 模型来描述通用的计算业务，进而设计了一个自适应的DAG 区块链结构，将执行计算与产生区块的“挖矿”过程相关联，以激励节点的广泛参与，确保公平性。随后构建了一

个分散的多智能体环境，节点自主决定是否承担子任务。进一步设计了一种基于演员-评论家方法的分散多智能体强化学习算法，以适应复杂的计算卸载场景。实验部分与四种先进方案进行对比，在公平性、扩展性等方面均展示出优势。

第六章从安全角度出发，针对边缘分布式计算中的中毒攻击问题，提出基于链上信息的安全卸载方案。首先通过预实验得到该场景下模型中毒攻击的影响效果，在此基础上设计损失感知信用评估机制，更新节点信誉度评分。进一步采用基于洗牌的隔离机制，筛选出恶意节点。在此过程中，采用智能合约进行透明和公平的绩效评估和任务卸载，确保公平的奖励分配、声誉更新和基于贡献度的工人选择。通过与现有方法的综合实验对比，揭示了所提方案在攻击场景下的优越性能。

第七章对本文的研究工作进行了归纳总结，指出了当前研究工作的局限性， 并对未来工作的开展进行规划与展望。

## 第二章 相关工作综述

本章第一部分对边缘分布式计算进行概述。从边缘分布式计算的典型场景和应用等方面展开，详细阐述了边缘分布式计算关键场景、特征。随后，围绕计算卸载的关键影响因素和步骤，细致阐述了边缘分布式计算卸载的核心要点。本章第二部分从可信卸载、高效卸载、安全卸载以及区块链在计算卸载中的应用四个维度，综述了这四个方面的相关研究进展和主流思路。

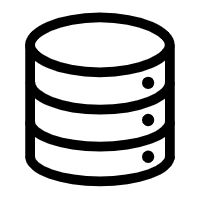
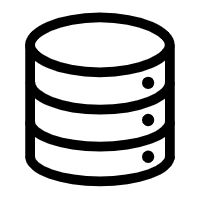
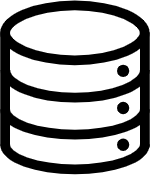
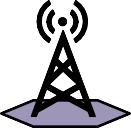
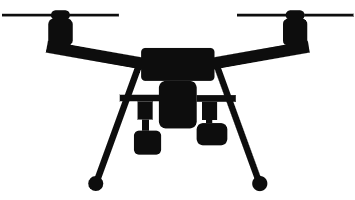
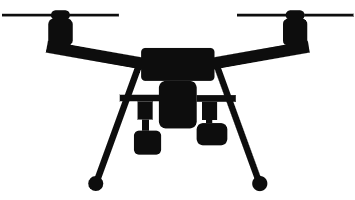
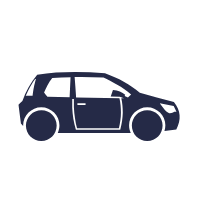
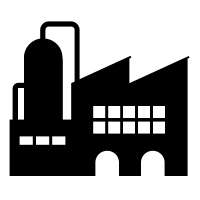
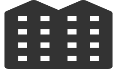
#### 边缘分布式计算概述

* + 1. 边缘分布式计算场景

在数字化时代的推动下，移动互联网、物联网及智能设备的广泛普及，不仅极大地丰富了人们的生活，也引领了数据量和计算需求的急剧增长。面对这一现状，传统的集中式计算模式因其在处理速度、延迟以及能源效率等方面的局限性而面临到前所未有的难题。值得注意的是，由智能设备产生的数据常常地理分布广泛，尤其是那些跨境的数据源与服务器，它们的数据传输往往会受到如欧盟《通用数据保护条例》[[5](#_bookmark153)] 等国家或地区政策的严格限制。鉴于这些挑战， 分布式计算——一种创新的计算范式——应时而生。它的核心理念是通过网络连接的众多计算节点的协同工作来完成计算任务，旨在优化计算资源的配置并提升其利用效率。

分布式计算依托于物理上分散的计算单元之间的紧密通信与协作，共同完成复杂的计算任务，具有以下显著优势：优化资源利用：通过整合分散的计算资源，实现超越单个大型计算机的处理能力；提升处理速度：得益于并行处理机制，分布式计算允许多个任务在多个计算节点上同时进行，大幅提高了数据处理的速度；增强系统可扩展性：随着任务量的增长，系统可以通过简单地增加更多的计算节点来轻松扩展，这使得分布式计算极具弹性和扩展性；提高系统可靠性：通过数据的冗余存储和任务的备份执行，分布式计算极大增强了系统对各种故障的容忍度，保证了计算任务的连续性和数据的安全性。近年来，无论是在工业界还是学术界，分布式计算都受到了广泛关注。在工业界，Apache Spark[[6](#_bookmark154)] 和 Hadoop MapReduce[[7](#_bookmark155)] 代表了两大核心技术框架；在学术界，基于分布式计算的理念，学者们提出了联邦学习（Federated Learning[[8](#_bookmark156)]）、分散计算（Dispersed

数据中心层



云服务器

边缘服务器

基站

基站

边缘设备

以及边缘数据

**边缘服务器层**

**边缘设备层**

图 2-1 边缘分布式计算网络示意图

Computing[[9](#_bookmark157)]）等概念，进一步推动了分布式计算成为研究的热门话题之一。 随着数据产生地点的日益分散以及对实时数据处理需求的与日俱增，将计

算资源布局靠近数据源变得尤为重要。这种需求促进了边缘分布式计算概念的兴起——即在网络的边缘部署大量计算节点，协同工作，以分布式的方式进行数据的处理和分析。边缘分布式计算通过充分利用网络边缘的计算资源，有效地减轻了中央处理中心的负担，降低了数据传输过程中的延迟，从而显著提升了数据处理的速度和效率。

典型的边缘分布式计算网络包括三层结构：边缘设备层、边缘服务器层和数据中心层，如图[2-1](#_bookmark24)所示。边缘设备层：该层直接与数据源相连，主要承担数 据的初步收集及部分数据处理和分析工作。此层涵盖了众多设备，如智能手机、传感器、个人电脑等。边缘服务器层：此层包括部署在靠近边缘设备的计算资源，例如基站、微数据中心或专用边缘服务器等。它们的主要任务是从边缘设备收集数据，执行更为复杂的数据处理与分析任务，进一步加工数据。数据中心层：尽管边缘分布式计算侧重于在网络边缘进行数据处理，数据中心层仍发挥着至关重要的角色。特别是在需要进行大规模数据分析、长期数据存储或复杂决策过程时，这一层能够向边缘设备和节点提供额外的计算资源、高级分析服务及全球数据管理功能。这种分层架构不仅使数据处理速度更快、延迟更低， 而且通过在地理位置相近的节点上进行数据计算，有效减少了数据传输过程中的潜在暴露风险，从而增强了数据的安全性与隐私保护。

边缘分布式计算尤其适合于需要快速响应的应用场景，如工业物联网[[10](#_bookmark158)]、

智慧城市[[11](#_bookmark159)] 等，通过在数据产生地点或其近邻处理数据，显著优化了整个系统的计算性能与响应能力。以一个布满摄像头、距离传感器等设备的智能交通系统[[12](#_bookmark160)] 为例，此系统依赖于实时数据的快速处理，以实现交通流量控制和事故预警等关键功能。在边缘分布式计算模式下，这些从传感器上捕获的实时数据不只是在临近的边缘车辆或路边计算设备上进行预处理，而且还能够通过边缘节点间的密切协作，完成更为复杂的数据分析与决策支持任务。这种计算模式极大提高了数据处理的效率和系统的响应速度，同时，也增强了系统的可扩展性与容错能力，为智能交通系统的高效运行提供了坚实的技术支撑。

边缘分布式计算不仅解决了传统集中式计算模式所面临的效率、安全等挑战，也为未来计算模式的发展指明了新方向。随着技术的不断进步和应用场景的持续扩展，其重要性与应用价值预计将不断提升。

* + 1. 边缘分布式计算卸载

在探索边缘分布式计算的深层次应用和优化策略时，计算卸载技术扮演着不可或缺的角色。计算卸载指的是在不同的计算单元之间转移计算任务的过程， 目的是减轻单个设备上的计算负载、缩短任务的处理时间，从而提升整体的计算服务质量。在边缘分布式计算环境中，计算卸载是实现计算任务靠近数据源的基石。通过智能分配边缘节点间的计算任务，这种技术不仅能够充分利用靠近用户的计算资源，减少数据在网络中的传输距离及时间，还能显著增强系统的响应速度与计算效率。这种方法在提高数据处理效能的同时，还能有效降低能耗，进一步优化整个计算环境的性能表现。

* + - 1. 计算卸载的关键考量因素

在计算卸载过程中，正确综合考量多项关键因素对于确保计算任务有效、高效地完成至关重要。这些因素包括：

计算任务特性：首先要评估任务的独立性，即任务是否可以独立执行，不需要频繁的数据交换。此外，还需要考虑计算与数据强度，因为计算密集型任务更适合进行卸载，而对于数据密集型任务，则必须考虑数据传输的成本。网络环境：网络的可用带宽直接影响数据传输的速度和延迟，而低延迟有助于减少任务响应时间，从而提升整体体验。网络稳定性也是一个至关重要的考虑因素，因为它直接关系到计算卸载的连续性和可靠性。边缘节点能力：包括计算能力（如 CPU 速度和内存大小）、能源供应（尤其是对于电池供电的设备而言， 能源是一个重要的限制因素）以及存储空间（对于执行某些计算任务而言是前提条件）。用户需求与服务质量：根据任务的优先级（紧急或高优先级任务可能

需要优先考虑卸载）和服务质量要求（包括处理时间、可用性和容错性等）来制定计算卸载策略。安全性与隐私：保护数据免受未经授权的访问或篡改，以及保障用户隐私是进行计算卸载时必须严格考虑的因素。

* + - 1. 计算卸载执行步骤

计算卸载的执行过程涉及到一系列精细化的步骤，包括：

任务识别与切割：首先，需要识别出那些适合进行卸载的计算任务，并评估这些任务是否能够被有效地分割成更小的子任务。这一步是整个计算卸载过程的基础；资源发现与评估：接下来，通过识别网络中可用的计算资源，并评估这些资源的计算能力、存储空间及网络状态等关键参数。这一步骤对于后续的卸载决策至关重要；卸载决策制定：基于对任务特性和可用资源状况的综合考量， 利用算法来决定哪些计算任务应当卸载到哪些目标资源上。这一步骤中，可能需要考虑实时数据，以动态调整卸载决策以适应网络和资源的即时变化；任务迁移与执行：在确定卸载目标后，接下来需要安全地传输任务所需的数据和代码到目标设备，并配置目标设备的执行环境以启动任务；结果收集与反馈：一旦计算任务完成，便将结果反馈给请求设备。根据执行结果，优化未来的卸载策略，以提高整体的计算效率和响应速度。

#### 相关研究现状

边缘分布式计算卸载的实现面临着多方面的挑战，包括信任度管理、多节点环境下高效的卸载策略以及整个过程的安全保障。本节旨在从这三个关键维度出发，梳理该领域内的研究现状，揭示各项研究如何互补，共同促进边缘分布式计算卸载技术的发展。同时，也指出了现有研究在理论与实践应用中存在的问题与不足，为后续研究提供了改进方向。最后，本节就关于区块链在边缘分布式计算卸载中的研究进行介绍。

* + 1. 边缘分布式计算可信卸载相关研究

在边缘分布式计算环境中，参与节点的多样性与设备的异构性为计算卸载的可信性提出了重大挑战。确保节点间信任，不仅是实现高效和安全计算卸载的基础，更是维持整个系统运行稳定性的关键。近年来，众多研究集中于信任度建模与信誉管理机制的开发，旨在探寻解决此类问题的有效方法。

首先是信任度建模。在文献 [[13](#_bookmark161)] 中，作者提出了一种针对雾计算环境的安全服务选择的信任建模方法。通过整合模糊逻辑和最佳最差方法的多标准决策，

对服务质量、社交关系以及历史经验等多维信任度量指标的贡献度进行了综合评估。相比现有技术，该方法在信任值的收敛性和准确性方面分别提高了 30% 和 5%。文章[[14](#_bookmark162)] 设计 SC-TRUST，旨在轻量级计算平台上微服务组合中实现透明信任的构建与解析的挑战。通过深入分析服务组合的信任属性及服务工作流如何影响信任的建立与解析，本研究成功推导出一系列信任评估函数，为微服务组合的信任管理提供理论指导和实践参考。

信誉管理机制也是提升系统可信度的重要策略。在文献 [[15](#_bookmark163)] 中，尽管加密技术能够防御外部攻击，但对于已获得认证并加入网络的雾节点的内部威胁却显得力不从心，因此，作者介绍了一种针对雾计算环境的信任管理策略，旨在解决雾节点在协作和数据共享过程中遇到的安全性及隐私问题。通过分析雾节点之间直接与间接互动的相关历史数据，该策略有效管理了节点间的信任等级， 成功减少了约 66% 的恶意互动，并将服务响应时间缩短了大约 15 秒。Liang 等

人在文献 [[16](#_bookmark164)] 中，面向社交网络、无线传感器网络、云计算和雾计算构成的新型物联网中的信任缺失问题，提出基于多源反馈和雾计算的可靠信任计算机制， 通过收集感知层的信任反馈信息，进行推荐信任计算，建立新的信任度量标准， 提高系统的计算效率和可靠性。

尽管已有多种尝试和方法旨在解决相关问题，但应对不诚实个体的挑战依然存在，例如“搭便车问题”（Free-Rider Problem）。为此，Wang 等人[[17](#_bookmark165)] 将任务分配转换为拍卖模型，这防止了资源卖家在支付决策上撒谎。同样地，Alamer 等人在文献 [[18](#_bookmark166)] 中提出了一个可靠的投标方案，以防止车辆通过提交不诚实的投标来获利。此外，Xu 等人在文章 [[19](#_bookmark167)] 中提出了一种面向智能城市边缘计算的信任导向服务放置方法。该方法采用强化的帕累托优化算法，在执行性能与隐私保护之间寻求最佳平衡。进一步地，利用基于理想解相似度顺序偏好技术和多准则决策制定技术，得到所获得服务放置策略中的最优解。

然而，现有方案往往是针对特定问题而设计，这限制了它们在广泛应对多种恶意行为时的有效性。在不损害系统性能的基础上，如何进一步确保计算环境的可信，仍然是一个亟待深入探讨的研究领域。

* + 1. 边缘分布式计算高效卸载相关研究

本节将重点讨论如何在边缘环境中有效地实现资源协同与任务卸载，以优化整体系统性能和提高资源利用效率。

Xu 等人在文献 [[20](#_bookmark168)] 中针对车联网场景，提出了一种将计算任务从云端卸载至边缘服务器的策略，并进一步设计了一套优化边缘服务器的量化部署方法。

Song 等人在 [[21](#_bookmark169)] 中探讨了边缘计算环境下计算服务器部署与用户计算任务卸载策略的联合优化问题。随着智能设备的不断发展，边缘分布式计算任务不再局限于在边缘的高性能服务器上。这种趋势催生了如基于众包的移动设备云[[22](#_bookmark170)] 等新概念的出现。Zhu 等人在文章 [[23](#_bookmark171)] 中将相关问题公式化，在系统状态被假定为静态或完全已知的条件下进行研究。

然而，这些研究往往忽略了边缘分布式环境的随机性和不可预测性，特别是考虑到不同设备的计算资源和网络环境的动态性[[24](#_bookmark172),[25](#_bookmark173)]，这要求设计出更为灵活和主动的计算卸载策略。Chen 等人在文献 [[26](#_bookmark174)] 中针对车联网中移动边缘计算服务器覆盖不足的问题，通过引入多跳概念并结合贪心算法及离散蝙蝠算法，提出了一个旨在提高任务执行效率的分布式多跳任务卸载决策模型，以充分利用路边闲置的车辆计算资源。尽管这一方法取得了一定的进展，但在卸载效率和任务分配优化方面仍有较大的提升空间。

除此之外，目前绝大多数研究仅将每个计算服务视作一个最小的计算单元[[27](#_bookmark175),[28](#_bookmark176)]，而忽略了子任务之间的耦合和依赖关系。子任务间的这种相互依赖性 将直接影响计算效率和计算卸载的可用性，应用程序内部组件间的复杂联系应当给予重视。PipeEdge[[29](#_bookmark177)] 尝试通过将分布式深度神经网络（Deep Neural Network， DNN）训练任务划分为不同的子任务，并在边缘节点上进行流式处理来解决这一问题。然而，此类方案仅限于串行依赖性任务，对于具有复杂依赖性的任务则显得力不从心。值得注意的是，Al-Habob 等人[[30](#_bookmark178)] 以及Rahimzadeh 等人[[31](#_bookmark179)] 的研究开始关注向多个移动边缘计算服务器并行和顺序卸载任务的问题，并探讨了基于遗传算法和冲突图模型等的调度算法。这些方法能够在考虑任务间依赖性的同时，优化资源分配和计算效率。尽管上述方案在提升计算卸载效率方面取得了进展，但在计算节点选择上往往偏向于拥有充足算力的设备，忽略了算力相对较弱的设备的参与权，从而可能导致资源的不公平分配。虽然“强者恒强”的策略表面上看似高效，但实际上，这并不利于建立一个具有更高包容性和公平性的边缘计算环境。特别是在未来强调去中心化和用户参与度的 Web3.0

（The Third Generation of The World Wide Web）时代，这种策略中的偏见可能会成为阻碍边缘分布式计算技术广泛应用和发展的一大障碍。

总之，尽管当前关于计算卸载效率的研究已经取得了一定程度的成果，但在确保边缘分布式计算卸载的计算效率与公平性方面，仍面临着众多挑战与机遇。

* + 1. 边缘分布式计算安全卸载相关研究

在边缘分布式计算环境中，委派计算任务给边缘节点所带来的安全挑战已经成为了重要的研究议题。这些挑战包括分布式拒绝服务（Distributed Denial of Service，DDoS）攻击、数据篡改和隐私泄露等风险[[32](#_bookmark180),[33](#_bookmark181)]。随着DNN 的广泛应用，特别是在边缘设备进行模型的分布式训练逐渐受到青睐[[34](#_bookmark182),[35](#_bookmark183)]，开放且不可信的网络环境对系统安全性的挑战也随之增大。这种环境下，训练过程更易遭受外部攻击，从而影响训练结果的可靠性和模型的完整性，其中模型中毒攻击尤其值得关注。

模型中毒攻击是指在模型训练阶段通过引入对抗性修改，以操纵或降低模型性能的一种攻击方式。这类攻击通过故意修改机器学习模型的内部参数，破坏模型的完整性和功能。根据攻击者的具体目标，模型中毒攻击分为无目标[[36](#_bookmark184)] 和有目标[[37](#_bookmark185)] 两种形式。无目标模型中毒攻击旨在通过操纵模型参数，降低模型对所有输入的准确性。相比之下，有目标模型中毒攻击专注于特定输入上的准确性降低，这种攻击形式要求攻击者掌握整个模型的参数架构。

为防御模型中毒攻击，分布式DNN 训练中通常依赖拜占庭容错机制，利用大多数节点的可靠性来减轻恶意行为的影响，保持最终模型的准确性。此外，文章[[38](#_bookmark186)] 尝试结合修剪和微调技术对抗特定攻击，如引入精细修剪（Fine-Pruning） 来大幅降低后门攻击的成功率。然而，该方案在处理模型并行性问题时效果欠佳。攻击者可以通过前向计算影响后续的客户端，并在梯度反向传播过程中影响之前的客户端，从而对整个计算群体的结果造成影响。Shubha 等人在文献 [[39](#_bookmark187)] 中尝试解决模型并行性的问题，但这种方法需要提前依赖于可靠的客户端进行预训练。

鉴于上述背景，显然需要设计出更为通用和健壮的安全解决方案来应对边缘分布式计算卸载中的DNN 模型中毒攻击。这不仅需要有效地识别和隔离恶意节点，以确保计算任务能够安全地卸载至可靠节点，还涉及到开发能在更广泛的攻击场景下保护模型训练过程完整性的机制。

* + 1. 区块链赋能的计算卸载相关研究

自 2008 年中本聪提出比特币以来，区块链技术凭借其数据审计、公开透明化及去中心化的特性在计算卸载领域展现了独特的价值。

在数据审计方面，区块链的不可篡改性为记录节点历史行为提供了强有力的保障。文献 [[40](#_bookmark188),[41](#_bookmark189)] 基于区块链记录的历史行为进行计算卸载，不仅增强了数

据处理的安全性，也提高了处理效率。Turgay 等学者在文献 [[42](#_bookmark190)] 中，利用区块链账本的分布式存储特性，加强了可审计性和安全性，从而促进了医疗系统间信息的共享，为分布式医疗数据训练提供了支持。

在公开透明化方面，利用区块链技术，所有用户操作都可以被记录在链上， 确保了系统操作的高透明度和可靠性。这一特性在雾计算中的应用，尤其是在安全区块链辅助的访问控制方案中，显示了其强大的潜力[[43](#_bookmark191)]。区块链技术还被 应用于无线网络虚拟化，有效地解决了资源分配的双重支付问题，减少了业务摩擦，并增强了行业的信任度和透明度[[44](#_bookmark192)]。

在去中心化方面，区块链技术为计算卸载提供了一种全新的解决思路。传统的计算卸载中，集中式处理方案[[45](#_bookmark193)] 容易受到 DDoS 攻击或单点失效问题， 临时波动或服务暂停会导致计算效率大幅下降。文章 [[46](#_bookmark194)] 提出了一个完全去中心化的计算资源共享框架，支持占优势策略激励相容（Dominant-Strategy Incentive-Compatible，DSIC）的双边拍卖。通过智能合约支持的分布式账本，极大地简化了交易各方的行为并减少了市场操纵的机会。Qu 等人在 [[47](#_bookmark195)] 中，将区块链整合工业 4.0 网络的认知计算中，并开发了一个去中心化的认知计算模型。在此模型中，区块链作为底层架构部署，所有本地设备均视为区块链的矿工和用户，借助区块链的分布式特性来选择高质量和可信的边缘设备执行计算任务。Chen 等人[[48](#_bookmark196)] 提出了一种基于智能合约的可信资源分配机制，旨在解决边缘服务器资源定价和服务质量评估中的问题。在此机制中，终端用户可以基于对延迟和价格的实际需求，从四种定价方案中选择一种购买模式，利用智能合约自动匹配高性能的边缘服务器给终端用户。

尽管区块链技术在计算卸载领域展现了巨大的应用潜力，但其自身也伴随着新的挑战和问题。例如，区块链操作的高能耗、交易处理的低效率等问题，都是需要进一步研究和解决的课题。

#### 本章小结

本章首先对边缘分布式计算进行概述，介绍了包括边缘分布式计算的典型应用场景以及在智能设备和物联网中的应用前景。接下来，本章详细介绍了计算卸载的必要性，特别是在边缘计算环境中，如何通过计算卸载提高计算效率和资源利用率，降低延迟，解释了边缘分布式计算卸载需要考虑的因素及其实施步骤。随后，深入探讨了可信卸载、高效卸载和安全卸载三个关键维度，分别从这些维度综述了当前的研究进展和主流思路。最后，本章强调了区块链技术

在计算卸载过程中的关键作用，并对区块链技术在边缘分布式计算中的应用进行了介绍。

## 第三章 多链协同的可信计算卸载模型

在数字化时代，对算力的需求日益迫切。尽管传统云计算中心具有强大的计算能力，但在面临高实时性要求和大规模并发任务时，由于数据传输延迟和中心化处理的限制，其性能往往不尽如人意。边缘计算技术随之诞生，通过将计算资源部署在网络边缘，近源地处理数据，显著缩短了数据传输的路径，从而加快了数据处理速度并提高了实时性，成为解决此类挑战的关键技术。然而，边缘分布式计算的节点来源多样性，给计算任务的可信卸载带来了新的挑战。鉴于此，本章提出了一种基于区块链技术的创新解决方案——多链协同的可信计算卸载模型，旨在提供一种既可信、高效、去中心化的计算卸载模型。通过引入多链并行的混合区块链架构，并规范化计算交互流程，MCTM 模型在系统的可扩展性、去中心化程度及可信度等方面取得了显著提升，为后续研究的开展提供了一个互信的可靠环境。

#### 引言

随着数字化转型加速，对高效算力的需求日渐增长。尽管传统云计算中心具备强大的计算力，但在处理高实时性要求和大规模并发任务时，因数据传输延迟和中心化处理的局限，往往显得力不从心。这一挑战促使边缘计算技术的兴起。边缘计算通过在网络边缘层近源地部署计算资源，有效地缩短数据传输路径，显著提升数据处理速度和实时性，成为满足高效算力需求的关键技术。

边缘计算的优势不仅体现在其低延迟处理能力上，还包括优化带宽使用和加强用户隐私保护。依托边缘服务器的解决方案[[49](#_bookmark197)]，将计算任务分散至接近数据源的边缘服务器执行，既解决了云计算中心远距离传输的延迟问题，也减少了数据在网络中的传输，降低了对中心化数据中心的依赖，从而提升了数据处理的效率和安全性。然而，在高峰时段满足巨大的计算需求可能需要部署大量边缘服务器，这对运营商而言可能是一笔不小的经济负担。因此，一些学者提出了基于众包的计算资源解决方案[[23](#_bookmark171),[45](#_bookmark193),[50](#_bookmark198)]，利用网络边缘的弹性资源来应对高峰期计算需求，通过动员社区中的闲置计算资源，不仅提升了资源利用率，也进一步降低了延迟，展示了边缘计算在资源调度和管理上的独特优势。

然而，边缘分布式计算的多源性特征引入了一系列安全和信任方面的挑战。鉴于计算资源来源广泛，包括个人电脑、智能手机及企业级服务器等多种设备，

这一多样性使得在边缘计算环境中实现计算任务的可信卸载变得尤为复杂。例如，存在“免费搭车”[[51](#_bookmark199)] 行为的风险，即参与者可能从系统中获益而不真正贡献其计算资源。此外，传统计算卸载方案依赖于集中式协调机构[[45](#_bookmark193)]，这使得系统容易遭受分布式拒绝服务攻击等网络安全威胁，从而可能导致计算服务的不可用。此外，系统可扩展性也是一个关键问题。考虑到边缘大规模并发的计算任务，一个中心化的调度代理可能成为计算处理性能的瓶颈。

区块链技术，凭借其分布式的特性，提供了解决上述挑战的途径。然而，当前区块链平台的有限吞吐量难以满足边缘场景中频繁且多变的计算任务需求。以比特币[[52](#_bookmark200)] 为例，其网络每秒只能处理 6 至 7 笔交易，显著限制了其在边缘分布式计算中的应用。Wang 等人[[53](#_bookmark201)] 尝试通过增加区块大小和交易数量来提高区块链的吞吐量，但这种方法没有考虑到每个参与区块链设备的网络带宽和计算能力的限制。

针对这一挑战，本章将深入探讨边缘分布式计算的特点对计算卸载可信性的影响，并提出能够保证系统环境可信的计算卸载模型，以支持边缘分布式计算在更广泛的应用场景中的部署和发展。具体来说，本章提出了多链协同的可信计算卸载模型（Multi-chain Collaborative Trusted Computing Offloading Model， MCTM），旨在为缘分布式计算提供一种可信、高效和去中心化的计算卸载解决方案。本章首先介绍了 MCTM 框架的设计思路，该框架采用了多链并行的混合区块链技术，并规范了交互流程，以提升系统的可扩展性、去中心化程度及可信度。随后，对于计算卸载算法的设计，给出了一个建模示例。最后，在实验部分，本章以基于强化学习的卸载算法为例，展示了 MCTM 的可信表现。

本章的其余部分安排如下：第 3.2 节描述了基本模型。第 3.3 节展示了多链

协同的区块链设计。第 3.4 节描述了计算卸载算法设计。第 3.5 节分析了实验结果。最后，在第 3.6 节中对本章进行总结。

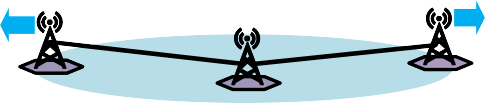
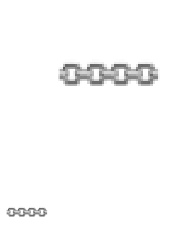
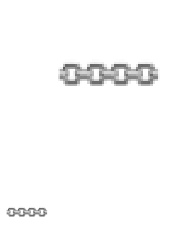
#### 基本系统模型

如图[3-1](#_bookmark35)所示（图中以直播转码计算任务为示意），从宏观角度看，系统进行了区域划分，每个集群包含一个边缘服务器和多个用户/工作者。计算发起者通 过边缘网络将计算任务（在直播转码场景中，计算任务包括直播流所需的处理， 包括转码、物理仿真、图形渲染等）传递给每个集群。

然后，具有合适计算资源的用户有资格申请任务。这些用户被定义为工作者。工作者处理计算任务，并将其发送给对计算结果有需求的其他用户或计算

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **区块链设计** | **侧链结构：** | **智能合约：** | PRSC | TPSC | TESC |

 **原始计算任务 ** **边缘进行处理**



**集群 1: 用户/工作者**

**视频流分段**

**...**

**任务分配**

**奖励-惩罚模型**

**...**

**边缘网络**

**不同分辨率**

**集群 N: 用户/工作者**

**计算发起者**

 **用户进行处理**

 **与区块链交互**

图 3-1 系统架构总览

发起者。在上述过程中，工作者和边缘服务器不断与区块链层进行交互。

区块链层为工作者信息、发布任务的细节等提供高吞吐量、抗篡改和去中心化的数据存储。智能合约提供工作者注册、任务发布和任务/奖励分配的自我管理。因此， 本模型设计了三种类型的智能合约， 分别是工人注册智能合约（Processor Registration Smart Contract, PRSC）、任务发布智能合约

（Task Posting Smart Contract, TPSC）和任务执行智能合约（Task Execution Smart Contract, TESC）。

值得一提的是，系统可以通过利用边缘辅助方法来增强模型的可用性。以转码过程为例，当一个集群的工作者不足时，边缘可以承担未执行的任务（转码过程中，通常提供最低质量的转码形式，以保证一定的服务质量）。通过这种方式，即使所有指定的处理工作者都失败了，系统仍然可以继续运行，保证最低的服务质量。

#### 基于多链的区块链架构设计

本章将区块链整合进了 MCTM 中。区块链是一种基于点对点（Peer to Peer， P2P）网络的去中心化数字账本，其关键理念包括：（1）交易被发起并在区块链网络上广播；（2）矿工验证交易，并将经过批准的交易存储在一个区块中，这些区块相互连接形成链式数据结构；（3）应用共识机制以确保不同矿工维护的区

块的一致性；（4）智能合约是由代码执行的自动协议，它使参与者无需第三方即可遵守规则。模型中定义了三种不同的个体角色：计算发起者、工人和矿工。计算发起者在系统平台上发布处理请求。工人是执行任务的工作者，而矿工则验证关于发布任务和任务完成状态等的交易。

* + 1. 架构基本介绍

为了处理传统区块链中极高的吞吐量，本节构建了由一个根链和多个侧链组成的根侧混合链。根链负责工人注册、数字身份、签名以及与跨链操作管理相关的其他功能。根链可以是任何人都可以加入的公共区块链（例如，以太坊），也可以是具有授权访问权限的私有链。对于每个计算任务，将建立一个新的侧链。该特定任务的所有相关交互，如任务发布、卸载和奖励分配，都在这条链上进行。每个侧链与一个集群相关联，同一侧链上的矿工验证相应集群的处理任务完成情况。

验证任务完成也非常重要，因为它需要持续评估计算的质量。如果在根链上应用矿工来评估众多工人的内容，那么传送内容的大量流量将消耗巨大带宽。因此，验证放在侧链上进行，为了实现这一设计目的，每个集群里面的工人都配备了监控器和区块链客户端。计算执行时，监控器通过多个维度估算接收的计算结果的质量（例如在直播场景下，包括分辨率、延迟和冻结时间等维度参数）。区块链客户端提供根据监控器的评估结果验证完成状态的功能。此外，系统中侧链采用了权威证明（proof of Authority，PoA）[[54](#_bookmark202)] 共识机制，第一个区块 由发布计算任务的节点生成。矿工们（可以由边缘服务器和工人共同组成）作为验证者，共同验证和签署交易。

* + 1. 基于链的计算卸载交互流程

MCTM 的功能通过三种不同的区块链智能合约实现：PRSC、TPSC 和TESC。每个智能合约的设计及其与区块链的交互方式如图[3-2](#_bookmark39)所示。

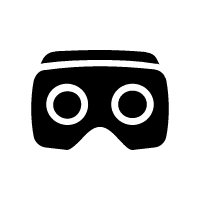
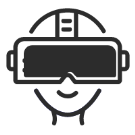
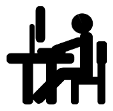
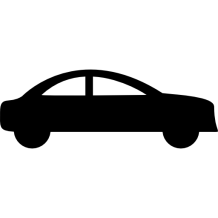
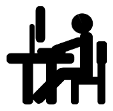
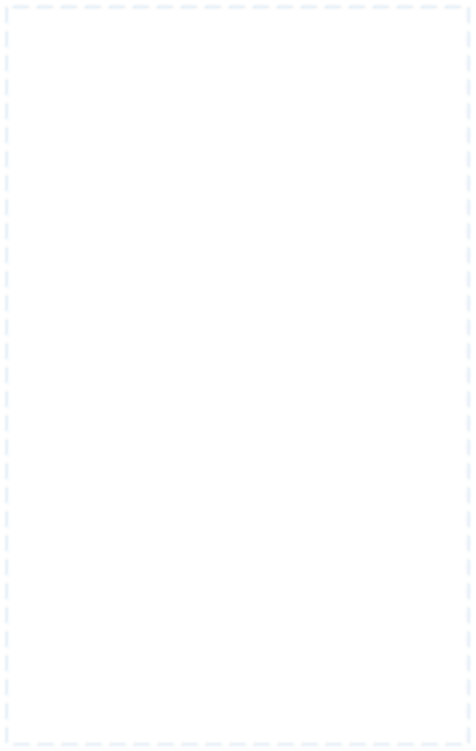
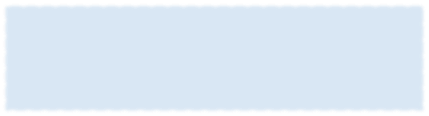
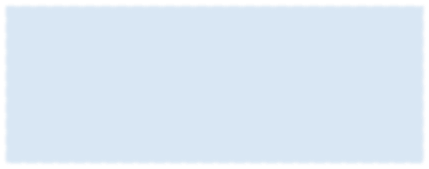
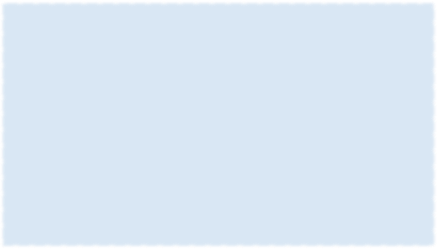
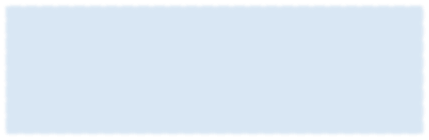
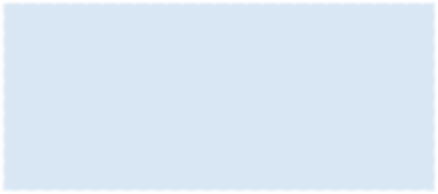
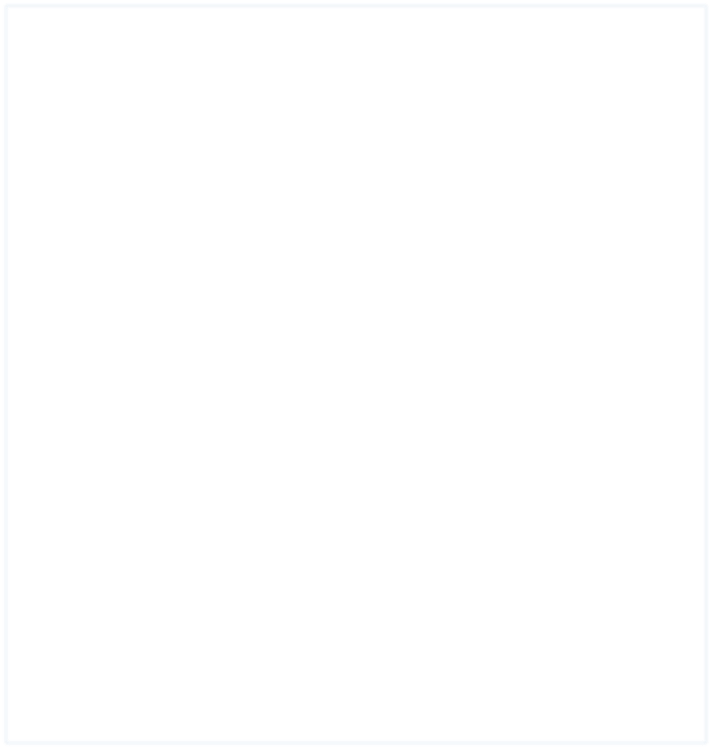
**PRSC** 在工人加入系统时被调用。然后，一个 ID 将与工人关联并记录在属性工作者 *ID* 中，其中工人的公钥哈希值作为其值。资源表示他们可以贡献的计算/存储资源量。在线时长表示激活时间的长度, 从上次上线时间开始，不累计计算。总任务记录了接受的任务数量。完成率维护了成功完成任务的比率。对于每个工人，完成率的初始值设为 1，并可以通过调用函数更新 (.) 进行更新。 **TPSC** 描述了发布的任务。任务发布 (.) 在计算发起者将计算任务发布到

系统平台时被调用。由于边缘计算任务往往不是原子化任务，能够被拆解为多

**[]**

**[]**

图 3-2 MCTM 中的智能合约设计



**用户**

**/工作者**

**处理器注册智能合约（PRSC）**

**根链**

**计算发起者**

**侧链**

*满意度*

*状态*

*押金*

*最终奖励*

*资源提供量*

*工作者 ID*

**工人信息**

**用户**

**/工作者**

**任务执行智能合约（TESC）**

*租赁期限*

*子任务 ID*

**更新(.)**

**奖励/惩罚(.)**

**任务分配(.)**

**TESC**

**TPSC**

**PRSC**

**任务发布智能合约（TPSC）**

*资源需求量*

*完成状态*

*持续时间*

*子任务ID*

*购买者 ID*

**任务**

*在线时长*

*总任务*

*完成率*

*资源*

*工作者 ID*

**工作者**

**注册(.)**

*保证金*

*总任务ID*

**更新(.)**

**保证金(.)**

**任务发布(.)**

**更新(.)**

**[]**

个子任务，每个子任务可以独立被写在到不同的计算节点上。因此，特别规定每一次计算任务的序号为总任务 *ID*，其中各个子任务的序号为子任务 *ID*。资源需求量和保证金分别指定了发布任务所需的资源和安全押金。计算持续时间由持续时间维护。完成状态指示任务是否完成。更新 (.) 也可以更新它们。

**TESC** 管理每个任务的任务分配和奖惩过程。函数任务分配 (.) 确定如何将任务分配给每个工人。子任务 *ID* 和租赁期限分别表示处理的计算的子任务序号和任务完成期限。工作者 *ID* 和资源提供量分别指代工人的 ID 和提供的资源。未完成任务的安全押金将在押金中支付。满意度（范围从 0 到 10）是从监控器获得的值，描述了平均任务执行质量。状态根据任务的完成状态指示，根据该状态，奖励/惩罚 (.) 将被调用以奖励或惩罚工人。最终奖励记录了最终的奖励。

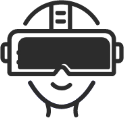
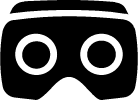
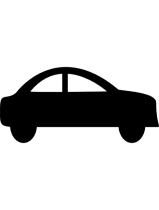
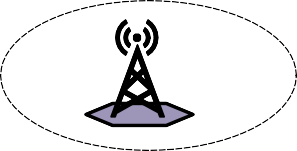
基于上述设计，MCTM 的工作流程如图[3-3](#_bookmark40)所示。

* + - 1. 工人注册：工人一旦加入系统就会向区块链注册。注册后，将为其分配一个唯一标识符。
      2. 任务发布：计算发起者通过调用 TPSC 向系统发布处理请求。TPSC 首先确认子任务 ID 和持续时间。然后估算资源需求并记录下来。此外，TPSC 还会通过保证金指定向计算发起者收取保证金。
      3. 任务分配：在每个处理周期 *t* 期间，TESC 将自动调用并执行任务分配。

TESC 中的任务分配 (.) 函数首先扫描 PRSC 和 TPSC 中的数据，以查询资源提

|  |  |
| --- | --- |
| **监控器** | |
| **工作者 ID** | **满意度** |
| A | 8 |
| **...** | **...** |

图 3-3 MCTM 工作流程



**用户/工作者**

**计算发起者**

**侧链 1**

**根链**

**侧链 2**

**步骤 3: 任务**

**用户/工作者**

**用户/工作者**

**...**

**...**

6

B

**罚**

**满意度**

**工作者 ID**

**步骤 6: 奖励/惩**

**监控器**

**分配**

**步骤 5: 任务**

**验证**

**步骤 1:**

**注册**

**步骤 2: 任务**

**发布**

**步骤 4: 任务传递**

供和需求。然后，它执行任务分配。任务/奖励分配的结果将发送给工人进行确认。确认结果后，每个加入的工人将根据 TESC 保存一个安全押金，直到他们成功完成任务才能提取。

* + - 1. 源内容传递：边缘服务器首先通过检索 TESC 生成的工人信息来确定工人，然后将计算任务分发给这些预分配的工人。
      2. 任务验证：工人将计算后的结果进行交付。收到其他计算结果的工人维护并更新一个监控器表。该表包括两个属性：工作者 *ID* 和满意度。满意度由监控器获取。通过计算所有工人的监控器表的平均满意度值，TESC 中的满意度将被更新。如果此值低于阈值，则任务应视为失败，状态设置为未完成。
      3. 奖励/惩罚：如果所有子任务满足要求，相应总任务的 TPSC 中的完成状态将设置为完成，计算发起者的押金将自动转给工人作为奖励。工人完成任务后也将提取他们的押金。否则，如果未完成，计算发起者将收回部分保证金。系统将以同一总任务内其他失败者的押金补偿完成分配任务的工人。此外，完成结果将与 PRSC 同步以更新完成率。

在上述过程中，区块链的分布式和不可逆特性防止工人和计算发起者为了自身利益操纵任务规格。此外，由于智能合约的存在，资源使用的支付可以在计算发起者和工人之间自动转移，无需第三方参与。对工人和计算发起者都要

求保证金/押金，使得对区块链的 DDoS 攻击成本非常高。由于区块链的去中心化特性，MCTM 也具有抗单点故障的能力。

#### 计算卸载算法设计

在当前的研究工作中，本章提出了一种多链协同的可信计算卸载模型，该模型的核心在于利用区块链技术的去中心化、不可篡改和透明性特点，以实现在边缘计算环境中的计算任务可信卸载。基于此模型，可以进一步探索如何根据不同场景和需求设计高效的计算卸载算法。本节旨在展示在可信模型的基础上，通过建模和优化来实现计算卸载，以适应不同的应用需求。

为了简化问题，本节考虑边缘计算环境中一个集群内中的任务分配。假设系统时间是分时的，网络状态在相同时隙内保持不变，但在时隙之间可能变化。在边缘计算场景中，系统效用通常需要考虑多个维度，如任务处理延迟、数据传输速率和计算资源的利用率等。因此，算法的目标是确保高效且平衡的资源利用。

本节定义资源积压 **Q** (*t*) 为在任何给定时间未能满足的资源请求。根据稳态

原理[[55](#_bookmark203)]，当 **Q** (*t*) 稳定时，意味着系统能够平衡资源供需。这要求参与计算的边缘节点（工人）提供的资源量需大于或等于所需资源量。此外，考虑到任务是

在边缘节点上本地执行的，能源成本也是一个重要考量。

因此，为了在确保任务高效完成的同时最小化资源成本，定义边缘节点 *i* 的效用函数为 *Ci* (*xi* (*t*)) = *hi* (*t*) · *xi* (*t*) −β · *Fi* (*xi* (*t*))。其中 *hi* (*t*) 表示节点历史上成功完成任务的比例，*Fi* (**x***i* (*t*)) 表示节点 *i* 为提供资源 *xi* (*t*) 所需承担的能源成本。β 是一个权重因子，用于平衡成本和可靠性之间的关系。高历史完成率体现了节点的可靠性，而低能源成本的优化则有助于节约能源。优化目标是从长期视角出发，最大化所有参与节点的总效用，同时确保系统的稳定性和高效性。

基于此优化目标，可以采取多种方法来解决计算卸载的优化问题。如：使用随机优化技术，通过构建数学模型并应用随机过程理论来找到最优或近似最优的资源分配策略。另一种方法是基于强化学习的决策过程，该方法不需要事先对系统模型有完整的了解，而是通过与环境的交互学习来逐步优化决策策略。

#### 实验与性能分析

本节旨在通过一系列数值仿真实验评估提出的多链协同的可信计算卸载模型。为此，实验使用 Go Ethereum 和 Torch 平台分别在配置为 3.2GHzx8，32GB

内存，GeForce GTX 1080 的工作站上实现了混合区块链框架。在模拟的边缘计算场景中，设置了 1000 个用户和 5 个边缘服务器（即 5 个计算集群），模拟真实世界中边缘计算资源的分布和用户请求的负载。在这个场景中，最多 20% 的用户充当计算资源提供者（即工作者）。同时考虑了恶意节点的存在，这些节点接受任务但不贡献任何计算资源，以测试算法对于不诚实行为的鲁棒性。为了实验的进行，本节在上一节的优化目标基础上，以基于深度 Q 网络（Deep Q-Network， DQN）算法的卸载策略为例，进行实验。

首先工人选择过程形式化为一个马尔可夫决策过程（Markov Decision Process， MDP），并将任务分配和资源管理问题描述为连续的、高维的状态空间，采用无模型的MDP 方法进行处理。状态空间：定义的系统状态包括工作者数量、可用计算资源、历史完成率和每单位资源的能源消耗。动作空间：动作空间由工人选择策略构成，这决定了计算任务与工人之间的匹配。考虑到队列稳定性的需求， 资源不足的节点在当前动作中的选择权被设置为 0。奖励机制：为了实现成本和

可靠性的最优平衡，将 *Ci* (*xi* (*t*)) 定义为 DRL 代理的奖励函数，依据此奖励机制

进行深度强化学习算法的训练和策略优化。在此基础上，进一步使用了DQN 算法来更新模型网络和调整任务调度策略。

为了进行比较，考虑了以下五种基准方案：

* 服务器：每个边缘服务器自行执行所有任务。
* 随机：从工作者中随机选择工人。
* 没有区块链的 MCTM（无区块链）：没有区块链结构，工作者的信誉不会被记录。卸载算法在边缘服务器上执行。
* 没有侧链的 MCTM（无侧链）：所有区块链交互直接在根链上进行。

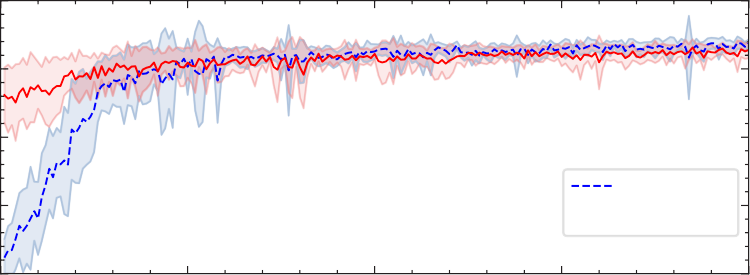
需要注意的是，除非另有说明，实验是基于 10% 的恶意节点比和 16 个总任务进行的。

首先，本节验证基于DQN 的算法在离线和在线阶段的收敛性。为了更清楚、准确地展示数据波动，本节重复进行了十次实验，并将同一回合（Episode）的数据分成一组。然后，在图[3-4](#_bookmark43)中阴影部分的线条是通过连接每组的平均值绘制的。阴影的上下限分别代表平均值加标准差和平均值减标准差。离线训练在 *t* = 60

时收敛，并保持稳定。然后，成熟的策略被采用为在线部署训练的初始参数，因

此，在在线阶段开始时就实现了相对较高的奖励。然后红线也收敛，表明它从长期来看找到了最优的任务分配策略。

0



训练

部署

−10

−20

奖励

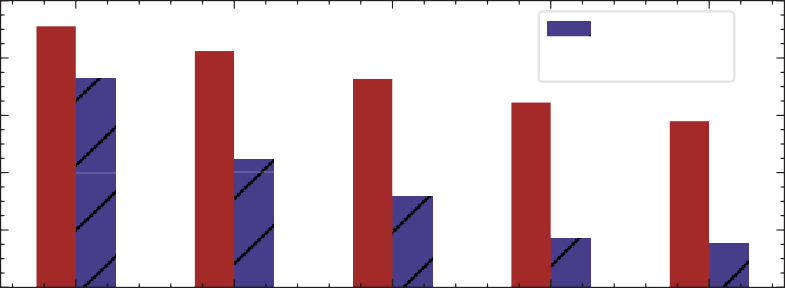
−30

−400 50 100 150 200

回合（Episode）

图 3-4 收敛性

10



无区块链

***MCTC***

8

6

满意度

4

2

0 10% 20%

30%

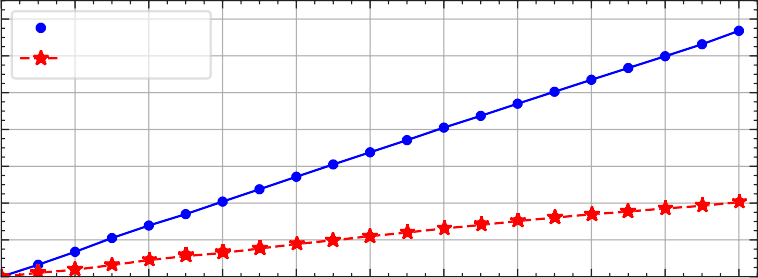
恶意节点比

40% 50%

图 3-5 不同恶意节点比例下的服务质量

接下来，检验不同恶意节点比下的性能，如图[3-5](#_bookmark44)所示。MCTM 得分最高， 与无区块链方案相比其下降趋势较慢。

1*.*4



×108

无侧链

***MCTM***

***Gas*** 消耗（gas）

1*.*2

1*.*0

0*.*8

0*.*6

0*.*4

0*.*2

0*.*00 4 8 12 16 20 24 28 32 36 40

任务数量

图 3-6 以太坊 gas 开销性能

在图[3-6](#_bookmark45)中，实验测试了 MCTM 和无侧链的 Gas 消耗。Gas 表示执行交易所需的计算资源。在以太坊中，一个区块中所有交易的总 Gas 不得超过 Gaslimit

10



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |  |  |  |  |
|  |  | 服务器  随机 |  |  |  |  |  |  |  |  |  |
|  |  | ***MCT*** | ***M*** |  |  |  |  |  |  |  |  |
|  |  |  |  |

8

6

满意度

4

2

0 4 8 12 16 20 24 28 32 36 40

任务数量

图 3-7 不同总任务数下的服务质量

的值，这限制了交易数量，进一步限制了每秒交易数（Transaction Per Second， TPS）。在 MCTM 中，大部分计算和交互迁移到了根链之外。因此，MCTM 的总 Gas 消耗较小。

最后，在图[3-7](#_bookmark46)中比较了不同总任务数量下的 MCTM、服务器方案和随机方案。当总任务数量较小时，MCTM 和服务器方案实现了相似的满意度。然而，一旦总任务超过 16 个，服务器方案的线条急剧下降，因为仅依靠服务器无法满足不断增加的计算任务。此外，随着总任务数量的增加，MCTM 的优势变得更加明显。

#### 本章小结

本章深入探讨了边缘分布式计算在实现计算任务的可信卸载方面的关键技术和挑战。通过提出的多链协同的可信计算卸载模型），本研究不仅解决了传统边缘计算中的延迟和安全性问题，还有效应对了计算节点多源性带来的挑战。实验结果证明，MCTM 模型能够有效提升边缘计算环境中的计算效率和系统的整体可信度，为边缘分布式计算的广泛应用奠定了基础。

## 第四章 智能合约驱动的自适应卸载策略

边缘分布式计算的动态性主要表现在计算资源、网络环境以及计算需求等的迅速变化上，这些变化对计算卸载策略的设计与执行提出了对高度灵活性和适应性的要求。在第三章的可信模型基础上，本章进一步探讨了边缘分布式计算动态性对计算卸载效率的影响，并提出了一种基于智能合约的自适应计算卸载策略。该策略充分考虑了边缘计算环境的多变性，包括设备频繁的加入与退出、网络条件的波动以及计算需求的动态变化等因素，实现了对资源供应与需求之间不断演变的高效管理。通过引入智能合约技术，该策略自动化地执行任务分配与奖励发放，进而确保了激励机制的公平性与高效性。同时，策略采纳随机优化方法来适应系统动态变化，确保了策略的长期优化性能。本章的研究不仅增强了边缘计算资源的利用效率，也为系统的稳定运行和可持续发展提供了新的解决方案，突显了在边缘环境下智能合约技术的应用价值和潜力。

#### 引言

边缘分布式计算作为应对大数据处理挑战的关键技术途径，在加速数据处理、降低延迟、提高系统响应速度等多个方面发挥着不可或缺的作用。然而，在实际应用过程中，其固有的动态性特征在现代计算场景中显得格外重要。动态性不仅体现在计算资源的变化多端和网络环境的不稳定性上，而且还涉及到计算需求的迅速变化和时效性问题。动态性的核心挑战在于如何高效地管理和利用资源，以适应资源供给与需求之间不断的变化。为应对这一挑战，不仅需要深入考虑计算任务的时空分布特征，还要兼顾设备的计算能力、网络状况以及用户需求的多样性和动态性。此外，边缘计算环境下设备的频繁加入与退出，以及网络条件的波动，进一步增加了系统的不确定性，从而对计算卸载策略的灵活性和适应性提出了更高的要求。例如，在工业物联网场景[[56](#_bookmark204),[57](#_bookmark205)] 中，亿万计的机器和控制系统的相互连接构建了一个复杂且多变的生态系统。在此环境下，边缘分布式计算的动态性表现在对设备状态的实时监测、工业生产流程的即时调整以及市场需求的快速响应上——根据当前的网络条件、设备的可用性以及数据处理需求的实时变化，动态地调整任务分配策略和计算资源优先级，以确保系统的稳定运行和高响应速度[[58](#_bookmark206)–[60](#_bookmark207)]。

针对现有研究在应对边缘分布式计算挑战时，普遍基于静态或已完全知晓

的系统状态条件进行问题的确定性表述[[61](#_bookmark208)]，这一方法未能充分考虑到诸如资源提供者地理位置的分布异质性、网络环境的不稳定性以及移动设备资源提供的不连续性等动态因素。实际情况中，为了适应系统动态性的变化并确保长期优化，采用随机优化策略进行在线任务分配显得至关重要。此外，设计有效的激励机制也是实现高效计算卸载的关键之一。鉴于处理任务时工作节点需消耗自身资源，系统需设计合理的奖励机制以促进资源共享。现有研究主要聚焦于如何防止工作节点为了获得更多奖励而虚报执行成本的问题[[46](#_bookmark194),[62](#_bookmark209)]，对防范免费搭车[[51](#_bookmark199)] 等其他不诚实行为则相对欠缺。这类行为不仅会降低激励机制的效率，还可能对系统的信任度和稳定运行造成严重影响。因此，构建一个既能激发参与者积极性又能有效防范不诚实行为的激励系统显得尤其重要。

为了解决上述问题，本章从边缘分布式计算的动态性特征着手，提出了一种基于智能合约的自适应卸载策略。该策略不仅充分考虑边缘计算环境的多变性，实现更加灵活和可靠的计算卸载方案，还通过引入智能合约技术，自动化地管理任务分配和奖励发放，保障了激励机制的完整性和公正性。通过这些措施，不仅提升了边缘计算资源的利用效率，也为保障系统的可持续发展和健康运行提供了强有力的支撑。

本章的主要贡献概述如下：

* + 1. 提出了一种基于智能合约的自适应卸载策略，采用基于 Plasma[[63](#_bookmark210)] 的高吞吐量区块链，以支持系统核心功能。部署四种不同类型的智能合约于区块链上，用于管理工作节点注册、任务发布、任务分配以及奖励/惩罚。
    2. 将任务分配问题建模为随机优化问题[[55](#_bookmark203),[64](#_bookmark211)]，提出了基于 Lyapunov 的在线决策方法，通过随机优化方法应对边缘计算环境的动态变化，实现了计算卸载的长期最优性。同时，还设计了一个奖励/惩罚算法，激励工作节点完成任务，确保资源的有效利用。
    3. 通过理论分析和实验验证，证明了所提出计算卸载策略的有效性，包括最优收敛性和对抗恶意行为等能力。

本章的其余部分安排如下：第 4.2 节将对系统模型进行详细介绍。第 4.3 节

阐述区块链和智能合约的设计。第 4.4 节描述自适应卸载策略。第 4.5 节进行理论分析。第 4.6 节和 4.7 节分别展示实验结果和总结本章内容。

**任务分配 注册 奖励 任务发布 根链区块 子链区块**

**控制层**



**工作节点**

**应用**

**智能合约**

**基于Plasma的区块链结构**

R/PSC

TASC

TPSC

RSC

**设备层**

图 4-1 系统总览

#### 系统模型概述

考虑一个边端设备作为计算节点的系统，如图[4-1](#_bookmark50)所示。移动设备，如车载设备、平板电脑和笔记本电脑，作为其中的工作节点。系统的控制层主要依赖于基于 Plasma 的区块链来维护有关工作节点、客户端和处理任务的信息。应用程序的客户端，如视频处理、计算、自动驾驶，充当资源购买者。系统还包括四种不同类型的智能合约，即注册智能合约（Registration Smart Contract, RSC）、任务发布智能合约（Task Posting Smart Contract, TPSC）、任务分配智能合约（Task Allocation Smart Contract, TASC）和奖励/惩罚智能合约（Rewarding/ Penalizing Smart Contract, R/PSC）。这些智能合约完成了系统的主要功能：工作节点注册、任务发布/分配和激励。

和第三章描述的智能合约类似，工作节点通过调用 RSC 来注册到系统中， 资源购买者通过调用 TPSC 来指定他们的需求。在每个处理间隔期间，TASC 将自动激活。它首先估算到达任务的资源需求，然后安排任务给工作节点，以最小化整体处理成本。R/PSC 包含一个奖励/惩罚方案，旨在激励资源提供的同时也实现某些属性，即个体理性、可靠性和防止免费搭车。

#### 区块链和智能合约设计

* + 1. 区块链结构设计

区块链是一个分布式账本，其运作方式如下：1）客户端向区块链参与者

（称为矿工）广播交易以进行交易验证；2）矿工将批准的交易存储在基于链的数据结构中；3）共识算法同步不同矿工存储的区块。在本章系统中，区块链有三个目的：1）维护工作节点的信息，包括可用资源、移动性、账户余额和完成率；2）管理工作节点注册、任务发布/分配以及奖励/惩罚；3）根据资源使用结果、任务完成/未完成的奖励/惩罚更新账户余额。

由于系统的任务处理主要依赖区块链，其吞吐量决定了系统的性能。现有解决方案[[46](#_bookmark194)] 依赖于基于链的区块链逐个批准任务分配的拍卖，这对于 IIoT 中频繁的任务调度来说是不足够的。因此，本章在第三章的基础上考虑多链协同的方法，将一种基于 Plasma 的结构[[63](#_bookmark210)] 融入系统中。这种新型结构包括一个根链和多个子链。多个子链可以同时处理智能合约和生成区块，从而提高了整体吞吐量。根链包括 RSC 并维护工作节点的账户信息。任何公共链，如以太坊，都可以用作根链，以增强可信度并简化用户访问。子链应用基于链的结构，在其中区块逻辑上一个接一个地链接，就像传统区块链一样。每个子链与一种类型的任务相关联，只通过 Plasma 合约与根链联系。该合约用于帮助子链将任务发布和分配结果等信息提交到根链生成的区块中，并且它还包括更新区块链参与者账户信息的功能。由于子链彼此独立，只需与根链交换信息，因此系统的可扩展性显著增强。相应链的矿工调用智能合约进行任务分配和奖励/惩罚。

由于不同矿工维护的区块可能因去中心化验证而变得不一致，因此需要一个同步矿工区块的共识算法。现有的解决方案，如PoW[[65](#_bookmark212)]，耗时且计算密集， 因为它要求每个矿工解决哈希谜题。其他如权益证明权益证明（Proof of Stake， PoS）[[66](#_bookmark213)] 减少了计算开销，但导致中心化，这违背了区块链的设计目的。为了减 少共识开销并减轻中心化的影响，本章把委托权益证明 DPoS[[67](#_bookmark214)] 作为共识机制。在权益委托证明（Delegated Proof of Stake，DPoS）中，由代表投票选出的一组见证人生成并批准区块，这消除了在每个不受信任的矿工处解决谜题的等待延迟和计算成本。由于见证人是通过去中心化的投票过程选出的，因此在去中心化和证明效率之间保持了良好的平衡。每个子链选出 10 名矿工作为代表进行见证

人投票。这条链中得票最多的前 5 名矿工成为见证人，他们每个人依次生成一个区块。

* + 1. 智能合约设计

系统通过没有集中化协调的自动执行代码——智能合约来实现其功能。因此，参考第三章的模型，本节设计了四种类型的智能合约：

注册智能合约（RSC）部署在根链上，当工作节点加入时被调用。它在根链上创建并存储一个 5 元组(*vi, li, bi, oti, ci*)，其中 *vi* 表示工作节点 *i* 的移动速度，*li* 表示位置，*bi* 表示历史完成率，*oti* 表示在线时间，*ci* 表示工作节点 *i* 的成本函

数。这些参数将根据 *i* 在处理过程中的行动进行更新。

任务发布智能合约（TPSC）部署在子链上，当客户端发布任务时被调用。它为每个发布的任务 *j* 生成以下 4 元组 (*Qj, Dj, Rj, S j*)，其中 *Qj* 和 *Dj* 分别指定 *j* 的资源需求和处理持续时间。*Rj* 是为 *j* 提供资源的报酬，*Sj* 是一个布尔参数， 1 表示 *j* 已完成，0 则相反。

任务分配智能合约（TASC）在每个处理周期开始时自动调用。它主要包括一个函数 *TaAlloc(.)*。该函数首先访问根链和所在子链以确认该周期的可用资源和资源需求。然后它输出选定的工作节点，并确定它们应提供的资源量。一个带有 3 个属性的表格维护任务分配结果。在此表中，***W*** 表示选定的工作节点集合，***R*** 记录每个节点提供的资源量，***F*** 跟踪他们的完成状态。

奖励/惩罚智能合约（R/PSC）部署在子链上，包含两个函数：*Re/PeAss(.)* 和 *Verify(.)*。*Re/PeAss(.)* 为每个选定的工作节点计算奖励和安全押金，并将结果写入区块链。*Verify(.)* 包括该类型任务的验证方法。这些方法可以由客户端预先定义，并随应用变化。如果任务的处理持续时间结束，*Verify(.)* 首先通过验证方法确认每个工作的完成状态。在批准任务完成后，此函数将自动根据 *Re/PeAss(.)* 的输出在工作节点和客户端之间转移报酬。

基于智能合约的计算卸载流程在图[4-2](#_bookmark57)中详细说明。它主要包括三个步骤：

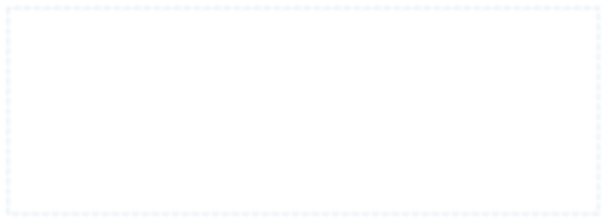
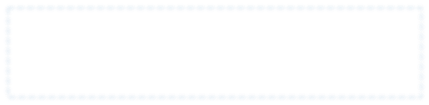
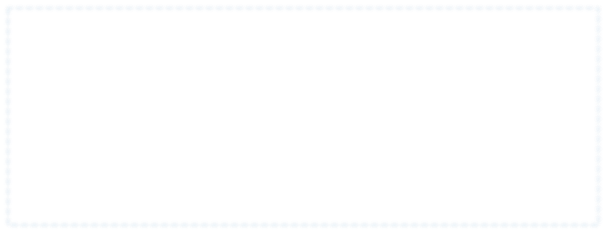
工作节点注册和任务发布，任务分配，奖励**/**惩罚。

#### 自适应卸载策略

本节介绍设计的自适应卸策略，包括问题建模和算法设计两部分内容

* + 1. 问题建模

当工作节点 *i* 加入时，它首先发送包含可用资源、成本函数、当前位置、移动速度的 *registration request* 到根链。将为 *i* 分配一个唯一的 ID，并调用 RSC 创建 5 元组(*vi, li, bi, oti, ci*) 来记录提交的信息。对于新加入的工作节点，其 *bi* 和 *oti*



**根链**

**子链**

**1: 工作节点注册和任务发布**

**任务发布**

触发 TPSC

**押金设置**

**3: 奖励/惩罚**

**奖励/惩罚计算**

触发 R/PSC

**押金设置**

**账户金额更新**

**任务验证**

**账户金额更新**

**任务分配**触发 TASC

**2: 任务分配**

触发 RSC

**注册**

**移动设备**

**客户端**

图 4-2 基于智能合约的计算卸载流程

将被设置为 0。在每个周期开始时，*i* 通过提交 *vi, li, ci* 到根链来更新可用资源、当前位置和成本函数。当客户端发布任务时，他们将需求资源和处理持续时间的请求发送到相应的子链。矿工调用 TPSC 来创建 (*Qj, Dj, Rj, S j*)。

在每个处理周期开始时，子链的矿工调用 TASC 并调用 *TaAlloc*(*.*) 来选择工

作节点并确定它们提供的资源量。为了使任务分配更清晰，首先将相关问题表述为一个随机优化问题，然后描述 TASC 中的函数 *TaAlloc*(*.*) 如何通过解决这个问题来得出长期最优分配。

问题表述：考虑一个由一组与基站连接并作为工作节点的移动设备组成的网络。设 *N* 为工作节点集合。系统被划分为小的处理周期，每个周期由*t* ∈ {1*,* 2*,..., T* } 索引。为了灵活的资源提供，假设已经应用了虚拟化技术，如文献 [[68](#_bookmark215)] 中所述，将工作节点的资源划分为资源块。设 *Hi*(*t*) 表示工作节点 *i* 的资源容量，*xi*(*t*) 表示 *i* 应提供的资源量。为了简化表述，本章只考虑在系统运行期间提供的计算资源，即 CPU。通过将标量 *Hi*(*t*) 和 *xi*(*t*) 转换为向量***H****i*(*t*) := {*Hik*(*t*)|*k* ∈ *K*} 和 ***x****i*(*t*) := {*xik*(*t*)|*k* ∈ *K*}，表述的问题可以轻松扩展到多种类型资源的情况，其中 *K* 是资源类型集合。设 *u*(*t*) 表示在 *t* 内到达的请求集

合。对于每个任务 *j*，其资源需求由 *a j*(*t*) 表示。在 *t* 内所需的资源总量可以由

*R*(*t*) := ∑ *j*∈*u*(*t*) *a j*(*t*) 给出。

首先为任务分配方案的设计提出三个指导原则：1）在任务到达的随机条件下，执行任务所投入的资源应平均满足到达任务的需求；2）移动设备为执行任务投入资源时，应最小化这些开销；3）在处理任务时，应优先考虑可靠的移动设备。对于第一个指导原则，定义资源需求积压量 *Q*(*t*)，其动态变化为：

*Q* (*t* + 1) = ⌈*Q* (*t*) −∑*i xi* (*t*) + *R* (*t*)⌉+*.* (4-1)

∈*N*

其中 ⌈*x*⌉+ := max{*x,* 0}。注意对于 *Q*(*t*)，∑*i*∈*N xi*(*t*) 和 *R*(*t*) 都是时变的，根据指导原则 (1)，需要 *Q*(*t*) 是“稳定”的，即根据文献 [[55](#_bookmark203)] 中定义 2.3(p.17)， lim*T* →∞ 1*/T* ∑*T* 1 *Q*(*t*) *<* ∞。

*t*=

针对其余指导原则，本节定义 *reliable* − *cost* 表达式作为任务分配问题的目标：

*C* (*x* (*t*)) = *F* (*x* (*t*)) + 1 *x* (*t*) (4-2)

*i i i i Hi* (*t*) *i*

其中第一项 *Fi* (*xi* (*t*)) 是衡量资源提供不满意度的成本函数。假设 *Fi* (*.*) 是两次可微的、非递减的，并且当 *xi* (*t*) = 0 时等于 0。此外，考虑到设备的能源限制，电池寿命较低的 *i* 应更不愿意贡献资源。因此，采用以下形式的二次函数：

*Fi* (*x* (*t*)) = *c* （*x*2 (*t*) + *xi* (*t*)(1 − *Ei* (*t*))） (4-3)

1

*i*

其中 *Ei* (*t*) 表示 *t* 时刻的电池寿命。定义可靠因子（Reliable Factor, RF）*Hi* (*t*) 来衡量 *i* 在处理任务时的可靠性。一般来说，一个人的 RF 依赖于其历史任务完成率 *bi* (*t*) 和在基站覆盖范围内的异常逗留时间。利用 *Mi* (*t*) */vi* (*t*) 来估计逗留时间，其中 *Mi* (*t*) 和 *vi* (*t*) 分别表示 *i* 到基站覆盖边界的距离和沿移动方向的移动速度。设 *Hi* (*t*) := φ*bi* (*t*) *Mi* (*t*) */vi* (*t*)，其中φ 是常数权重参数。任务分配问题如下表述：

min lim 1 ∑ ∑ *Ci* (*xi* (*t*)) (4-4)

*T* →∞ *T t*∈*T i*∈*N*

*s.t* **C1** : *Q* (*t*) *is stable*

**C2** : *xi* (*t*) ∈ [0*, Hi* (*t*)] *,* ∀*i* ∈ *N*

其中目标是最小化整体 *reliable* − *cost* 表达式的时间平均值。约束 **C1** 确认了第

一个指导原则，而 **C2** 是资源容量约束。考虑到 *Ci* (*.*) 是凸的，上述问题可以被视为凸随机优化。

* + 1. 算法设计

由于 **R** (*t*)、*vi* (*t*)、*Hi* (*t*) 是随机的，设计一种离线优化方法并通过事后知识得出最优解是困难的。另一方面，仅优化每个时间段的瞬时目标可能会损害长期最优性，因为问题是时间耦合的。考虑到这些情况，本节通过 Lyapunov 优化[[55](#_bookmark203)] 形成原问题的最小漂移加惩罚（Min-Drift-Plus-Penalty，MDPP）。*Q* (*t*) 的Lyapunov 函数可以表示为 *Lq* (*t*) = 1*/*2*Q*2 (*t*)。设 ∆*Lq* (*t*) = *Lq* (*t*) − *Lq* (*t* − 1)。根据文献 [[55](#_bookmark203)]，得到MDPP 的上界，

∆*Lq* (*t*) + *V* E{∑*i*∈*N Ci* (**x***i* (*t*)) |*Lq* (*t*)}

≤*B*′ + *Q* (*t*) E{*d* (*t*) |*Lq* (*t*)}+ *V* E{∑*i*∈*N Ci* (**x***i* (*t*)) |*Lq* (*t*)}

(4-5)

其中 *V* 是一个预先给定的非负常数，称为惩罚。令 *d* (*t*) = *R* (*t*) −∑*i*∈*N xi* (*t*)，选取MDPP 的上界作为优化目标。每个 *t* 上的任务分配问题可以重新划分为:

min *V* ∑*i*∈*N Ci* (*xi* (*t*)) + *Q* (*t*) *d* (*t*) *s.t* **C2**

上述问题在每个时隙都是一个确定性凸优化。具体来说，是它的优化器

*xi*∗ (*t*) := {argmax{*V* ∑ *Ci* (*xi*) + *Q* (*t*) *d* (*t, xi*)}}*Hi*(*t*)

*xi*

(4-6)

0

*i*∈*N*

*Hi*(*t*)

其中 *d* (*t, xi*) := *R* (*t*) −∑*i*∈*N* 话，*xi*∗ (*t*) 可解：

*xi*。令 {*xi*

}0 := min{max{*xi,* 0}*, Hi* (*t*)}。给定凸性的

∇*xiV* ∑*i*∈*N Ci* (*xi*) + *Q* (*t*) ∇*xid* (*t, xi*) = 0 (4-7)

根据上述设计，*TaAlloc(.)* 的执行流程如下：首先估计 *Q* (*t*) 并通过观察任务到达和工作节点注册信息来确定每个工作节点的 *Ci* (*xi* (*t*))。然后为每个工作节点计算 *xi*∗ (*t*) 并写入区块，更新 *Q* (*t* + 1)。*TaAlloc(.)* 的伪代码如算法[4-1](#_bookmark61)所示。

* + 1. 奖励**/**惩罚机制设计

在确定了每个工作节点提供资源的量 *xi*∗ (*t*) 后，矿工调用 R/PSC 并调用

*Re/PeAss(.)* 来确定完成任务的奖励和未完成任务的惩罚。

算法 **4-1:** *TaAlloc(.)* 流程

输入： 网络中的节点 *N* , 当前时间 *t* 的历史信息 *Hi*(*t*), 时间 *t*, 队列状态 *Q*(*t*), 非负惩罚参数 *V* , 当前时间 *t* 的系统状态 *H* (*t*);

输出： 对于每个节点 *i*, 时间 *t* + 1 的最优任务分配 *xi*∗(*t*);

1: **while** *i* ∈ *N* **do**

2: *xi*∗(*t*) ← argmax{*V* ∑*i*∈*N Ci*(*xi*) + *Q*(*t*)*d*(*t, xi*)};

*xi*

3: **if** *xi*∗(*t*) ≥ *Hi*(*t*) **then**

4: *xi*∗(*t*) ← *Hi*(*t*);

5: **else if** *xi*∗(*t*) ≤ 0 **then**

6: *xi*∗(*t*) ← 0;

7: **end if**

8: 将 *xi*∗(*t*) 写入区块链并通知工作节点 *i*;

9: **end while**

10: *Q*(*t* + 1) ← *Q*(*t*) + *R*(*t*);

奖励：对于每个 *t*，被选中的工作节点集合表示为 *W* (*t*) ≜ {*i*|*xi*∗ (*t*) *>* 0}。通过 *Ct* (*x*) = max*i*∈*W* (*t*) *Ci* (*x*) 定义 *t* 的成本算子 *Ct* (*.*)。*R* (*t*) 表示 *t* 期间资源使用的报 酬。为了激励工作节点提供资源，*R* (*t*) 应不小于 *Wi* (*t*) 中工作节点的累积成本， 即 ∑*i*∈*W* (*t*) *Ci* (*xi*∗ (*t*)) *< R* (*t*)。设 *t* 时的报酬为

*R*(*t*) = ∑*i*∈*W* (*t*) *Ct* (*xi*∗ (*t*)) (4-8)

*W* (*t*) 中的工作节点根据其提供资源的量共享报酬 *R* (*t*)。根据以下公式确定*W* (*t*) 中每个工作节点的奖励：

*Ri* (*t*) = *xi*∗ (*t*) *R*(*t*)

∑*i*∈*W* (*t*) *xi*∗ (*t*)

− 1 *x*∗ (*t*) (4-9)

根据方程 [(4-9](#_bookmark62))，*Ri* (*t*) ≥ *Fi* (*xi*∗ (*t*)) 对所有 *i* 成立。这表明为系统提供资源对工作节点是有利的。即，奖励方案是个体理性的。

*hi* (*t*) *i*

押金计算和惩罚：工作节点是人携带设备，在资源提供过程中可能违背承诺，这损害了系统的性能和可持续性。因此，有必要对不诚实行为的工作节点进行惩罚。借助区块链智能合约的自动执行，本节引入了押金的概念。在工作节点 *i* 确认资源提供后，*Re/PeAss(.)* 将根据其提供的资源和 *i* 的可靠性设定押金。

如果工作节点完成任务，押金将被免除并返回到工作节点的钱包。否则，押金将被没收以补偿其他工作节点的损失。借助过 *Di*(*t*) = α*Ci* (*xi*∗ (*t*)) */*(*H* (*i*)) 计算 *i* 的押金，其中 α 是一个常数。

通过上述方法，*R/PSC* 在 *t* 时首先调用 *Re/PeAss(.)* 根据方程[(4-9](#_bookmark62)) 确定 *Ri* (*t*)

和 *Di* (*t*)，并将结果写入区块链。*R/PSC* 验证任务处理并更新 (*Qj, Dj, Rj, S j*) 的

完成状态 *S j*。如果任务完成，*R/PSC* 将价值 ∑*t Ri* (τ) 的货币转移到工作节点

τ=*tb*

的账户，其中 *tb* 和 *t* 分别是资源提供开始和结束的时间。如果未完成，*R/PSC* 将没收未完成任务的工作节点的押金并将其押金分配给其他工作节点作为补偿。算法[4-2](#_bookmark63)展示了 *R/PSC* 的伪代码。

算法 **4-2:** R/PSC 流程

输入： 集合 {*xi*∗ (*t*)}*i*∈*N* , 时间 *t*, 成本函数 *Ct* (*.*);

输出： 奖励集合 {*Ri* (*t*)}*i*∈*N* , 惩罚集合 {*Di* (*t*)}*i*∈*N* ;

1: 计算总奖励 *R*(*t*) ← ∑*i*∈*W* (*t*) *Ct* (*xi*∗ (*t*))。

2: **for** *i* ∈ *W* (*t*) **do**

3: 奖励: *Ri* (*t*) ← *R*(*t*)*xi*∗ (*t*) */* ∑*i*∈*N xi*∗ (*t*) − *xi*∗ (*t*)*/hi* (*t*)。

4: 惩罚: *Di*(*t*) ← α*Ci* (*xi*∗ (*t*)) */*(*Hi*(*t*))。

5: **end for**

6: TasVerify(.) 。

7: **for** *i* ∈ *W* (*t*) **do**

8: **if** *i* 结束任务 **then**

9: 支付 ∑*t Ri* (τ) 数量的货币至 *i* 的账户。

τ=*tb*

10: **else if** *i* 未完成任务 **then**

11: 从 *i* 的账户没收 ∑*t Di* (τ) 数量的货币。

τ=*tb*

12: 将上述 ∑*t Di* (τ) 数量的货币发给其他完成任务的工作节点。

τ=*tb*

13: **end if**

14: **end for**

#### 理论分析

本节分析了系统方案的几个重要特性，包括任务分配最优性、奖励可靠性、防止免费搭车和系统鲁棒性。

任务分配最优性：以下定理限定了 *xi*∗ (*t*) 与离线最优解的差距。

定理 **4.1** 对于每个时隙*t*, 给定∑*i*∈*N Ci* (*xi*∗ (*t*)) 以及后续得出的最优解∑*i*∈*N Ci* （*xopt* ）,

*i*

lim

1 ∑*T* ∑

*Ci* (*xi*∗ (*t*)) −∑

*Ci* （*xopt* ） ≤ *B*′

(4-10)

*T* →∞ *T*

*t*=1

*i*∈*N*

证明： 见附录.[A.1](#_bookmark285)。 □

*i*∈*N*

*i*

*V*

上述定理表明定理 **2** 展示了与 *xi*∗ (*t*) *, i* ∈ *N* 相关的平均队列长度 *Q* (*t*) 的无限性。

定理 **4.2** 假设存在一个正 ε, 给定惩罚为 *V* , 对于通过解决[4.4.2](#_bookmark59)得到的 ***Q*** (*t*), ***Q*** (*t*)

是稳定的, 同时以下不等式成立:

limsup 1 *T*

*Q t B*′ + *V* δ

(4-11)

*T* ∈∞

*T* ∑ ∑

*t*=1 *j*∈*J*

*j* ( ) ≤ ε

其中 δ := ∑

*i*∈*N*

（max*t,xi*∈**C2**{*Ci* (**x***i*)} −min*t,xi*∈**C2**{*Ci* (**x***i*)}）.

证明： 见附录.[A.2](#_bookmark287)。 □

可靠性分析：根据文献 [[46](#_bookmark194)]，定义如果参与工作节点的奖励最大化，则奖励方案是可靠的。对于本节提出的任务分配和奖励算法，这意味着没有工作节点可以通过操纵其成本函数获得更高的奖励。以下定理确保了所提策略的可靠性。

定理 **4.3** 在每个 *t*，奖励方案是可靠的，即对于任何工作节点 *i*，有 *R*′(*x*′ (*t*)) ≤

*i*

*i*

*Ri*(*xi*∗ (*t*))，其中 *R*′*i* 是通过将 *Fi*(*.*) 更改为 *Fi*′(*.*) 而得出的奖励，且 *Fi*′(*x*) ≠ *Fi* (*x*),

*x*′ (*t*) 是通过使用 *F*′(*.*) 计算的贡献资源量。

*i i*

证明： 见附录. [A.3](#_bookmark289)。 □

防止免费搭车：客户端和工作节点都可能成为免费搭车者。如果工作节点在分配任务时就收到奖励，它可能会在不提供足够资源的情况下接受奖励。如果工作节点在分配任务后才收到奖励，客户端可能会通过虚假报告完成状态来拒绝支付。系统通过部署智能合约引入了安全押金和自动转账机制。每个工作节点和客户端在执行和发布任务之前都必须向区块链设置安全押金。区块链矿工根据预定义的验证函数验证任务完成状态。如果矿工批准完成，R/PSC 将完成任务的工作节点的安全押金退回账户，并将客户端的奖励也转给他们。否则， R/PSC 将没收未完成任务的工作节点的安全押金，并退还客户端的押金。上述过程由区块链自动运行，无需第三方干预，从而防止了免费搭车行为。

系统鲁棒性：与现有依赖中心节点解决方案不同，本方案使用基于 Plasma 的区块链进行分布式管理。这是因为处理信息（例如任务分配结果、注册信息等）的区块由区块链中的多个矿工存储。一个或几个矿工的失败不会影响区块链的可访问性。此外，智能合约也由不同的矿工分布式部署和执行，这确保了控制逻辑的去中心化。因此，系统也避免了单点故障，因为工作节点和客户端可以由区块链中的任何矿工服务，而不仅仅依赖于经纪人。

#### 实验与性能分析

实验考虑了一个 2000 × 2000 *m*2 的区域，其中有 750 个移动设备作为工作节点。一个基站位于该区域的中心，通信范围为 1000*m*。总仿真时间为 500*s*，每个单独的时间槽长度为 10*s*，以适应频繁的系统动态。MDPP 参数 *V* 设定为 500， 除非另有说明。假设每个工作节点的计算资源被虚拟化为具有相同容量的多个虚拟CPU。可用虚拟 CPU 的数量是一个随机变量，其中 *Ci* (*t*) ∼ *U* [0*,* 10]，*U* [*a, b*] 表示从 *a* 到 *b* 的均匀分布。10 个客户端作为资源购买者并生成请求来转租资源。请求到达模式遵循参数λ = 5 的泊松分布，除非另有说明。每个请求所需的虚拟CPU 数量从 [1*,* 20] 中随机选择。移动设备变得不诚实的概率设定为 0.1，除非另有说明。采用随机路点[[69](#_bookmark216)] 移动模型，速度变化 *V* (*t*) ∼ *U* [0*,* 20] *m/s*。

实验使用开源项目 Plasma-MVP[[70](#_bookmark217)] 构建基于 Plasma 的区块链。以太坊区块链[[71](#_bookmark218)] 作为根链，实验创建了七个 Plasma 子链并将它们链接到根链。每个 Plasma 链由 20 个矿工管理。本节在配备 Intel i7-9700k /@3.6GhZ CPU 和 Ubuntu 16.04.6 LTS 的台式计算机系统上实现了矿工角色。每个区块包含 1 × 106 gas，执行智能合约时将消耗 21000-50000 gas。即，一个区块至少可以包含由智能合约生成的20 笔交易。对于DPoS，将选择 5 个见证人并每 5 秒生成一个区块。

算法性能：本节研究了仿真过程中的总成本和总队列长度 *Q* (*t*)。图[4-3](#_bookmark69)显示

了不同 *V* （从 500 到 4000）的总成本。可以观察到，不同惩罚参数 *V* 对应的成本都在仿真过程中快速收敛。在仿真开始时，*V* = 4000 的结果具有最低成本，这证实了定理 **2**，较大的 *V* 产生较小的最优差距。图[4-4](#_bookmark70)显示了不同 *V* 条件下的队

列长度。随着 *V* 值的增加，也观察到稳定阶段队列长度的增长趋势。这种观察与定理 **1** 一致，该定理指出队列长度上界与 *V* 值之间的正相关关系。

本节还研究了所提策略在可靠因子变化下的性能，包括不诚实 MDs 的比例和移动速度。图[4-5](#_bookmark71)显示了平均奖励和押金如何随着不诚实工作节点的比例变化而变化。如图所示，随着不诚实工作节点的增加，奖励/押金的值急剧下降/上升。

4.0

V=4000

V=500 V=1000 V=2000

\*104

3.2

2.4

成本

1.6

0.8

0

0 50 100 150 200 250

##### 时间（s）

1.1

\*104

图 4-3 总成本随仿真时间变化

0.9

0.4

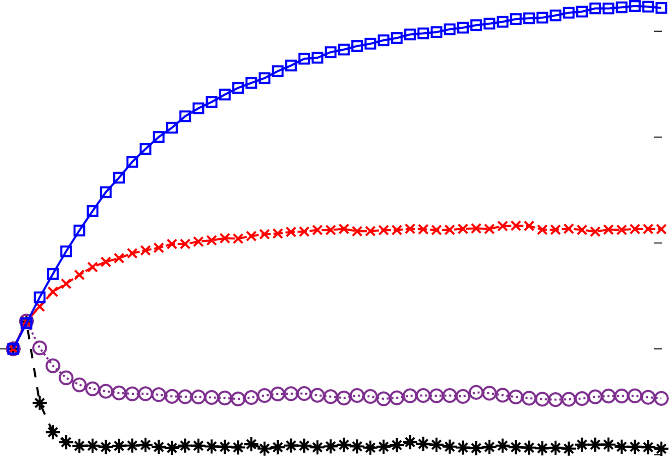
队列长度

0.5

0.3

0.1

0 100 200 300 400 500



V=4000

V=500 V=1000 V=2000

##### 时间（s）

图 4-4 总队列长度随仿真时间变化

这是因为系统试图增加押金以防止更多移动设备成为免费搭车者，这符合本章所提策略的设计目的。图[4-6](#_bookmark72)显示了在不同移动速度范围下的奖励/押金。如图所示，速度范围越高，奖励越小。这主要是因为较高的速度可能会增加移动设备移出基站范围的概率和任务未完成的风险。因此，押金也相应增加。

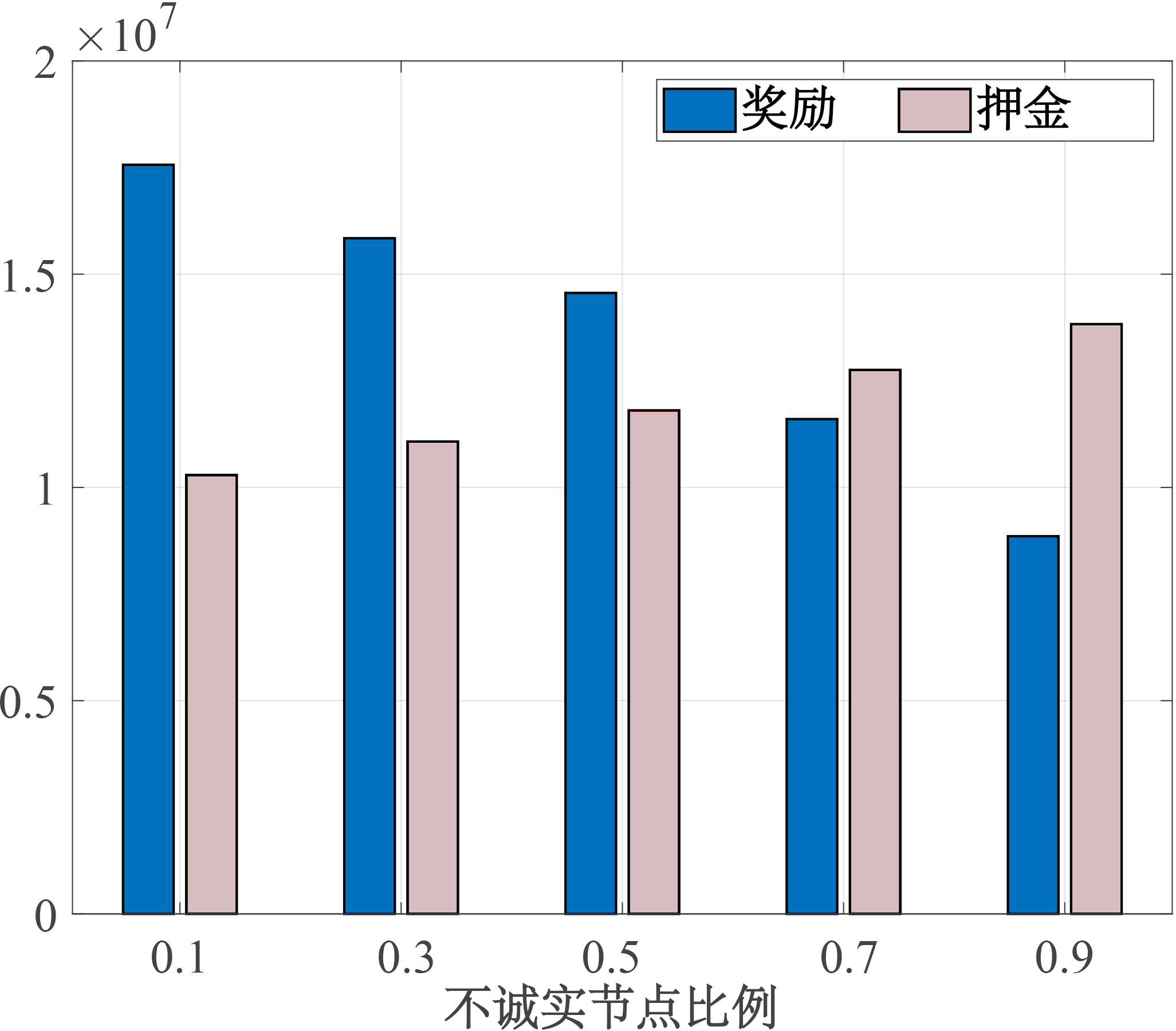


图 4-5 不同比例的不成实节点下奖励与押金变化情况

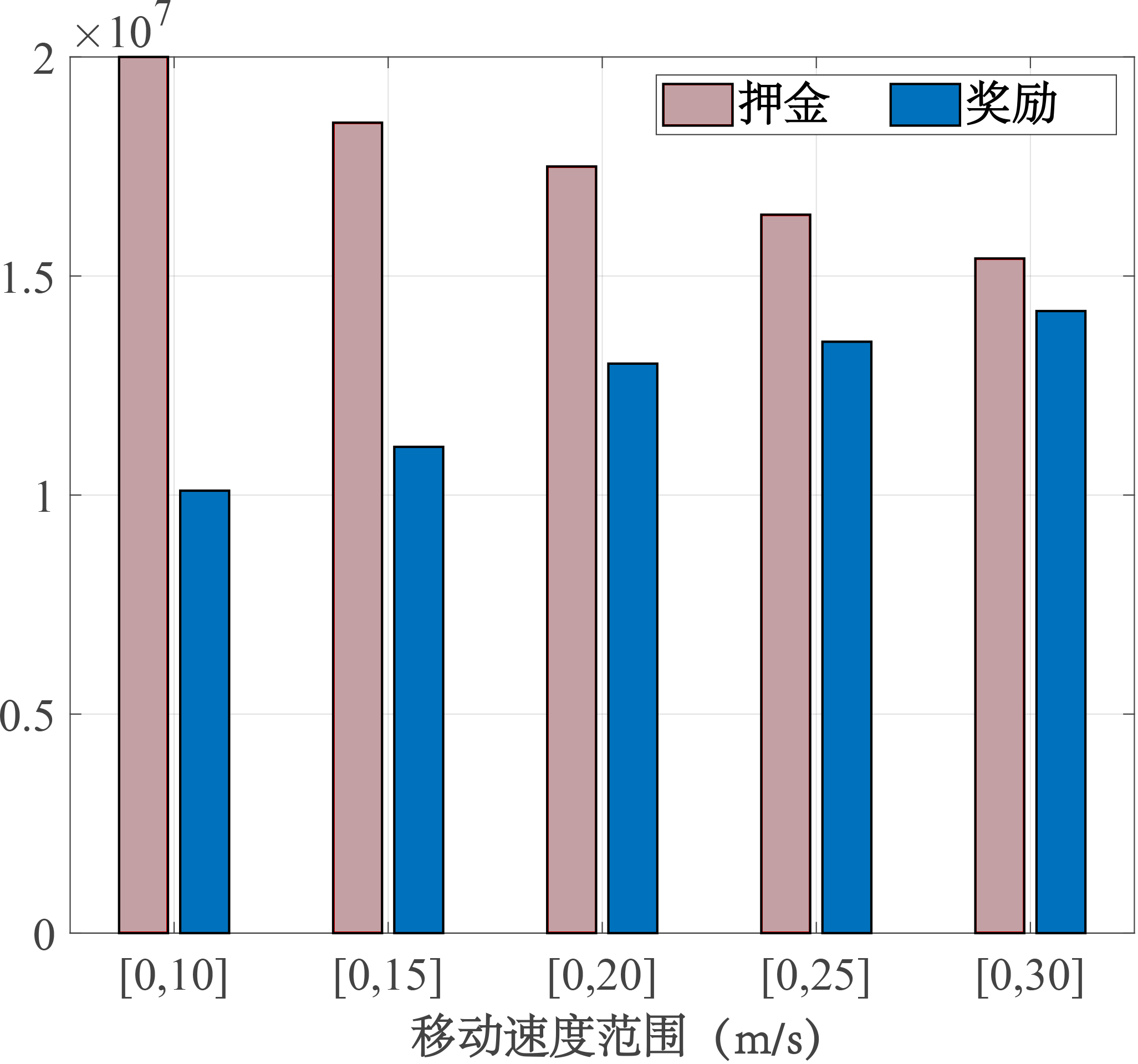
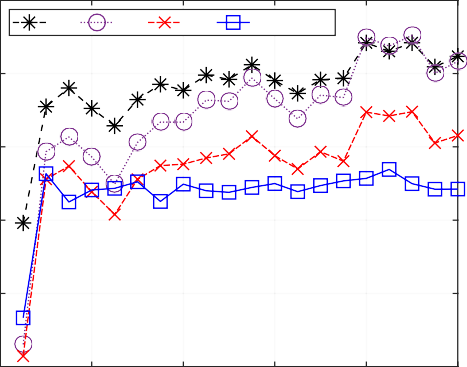


图 4-6 不同移动速度范围下奖励与押金变化情况

与不同解决方案的性能比较：本节将提出的方案与随机选择和可靠因子优先选择，以及一种最新的基于确定性的解决方案[[61](#_bookmark208)] 进行比较。随机选择随机选

5.0 \*104



RS RF DP 所提方案

6.5

\*104

4.0 5.2

3.0 3.9

成本

成本

2.0 2.6

1.0 1.3

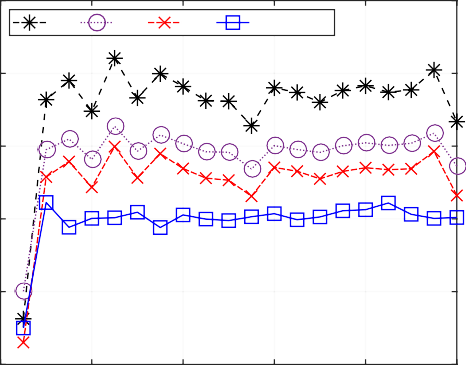
0

0 100 200 300 400 500

时间（s）

(a)

7.0 \*104



RS RF DP 所提方案

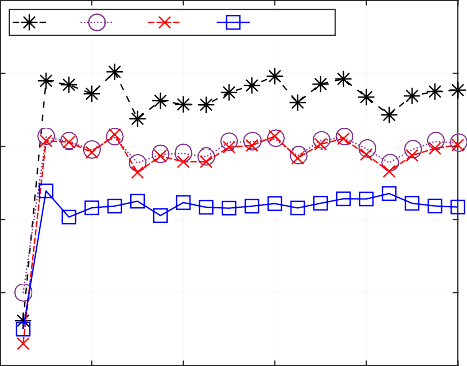
0

1 100 200 300 400 500

时间（s）

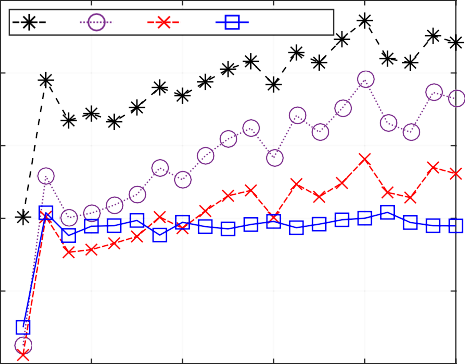
(b)

7.5 \*104



RS RF DP 所提方案

5.6 6.0



RS RF DP 所提方案

4.2 4.5

成本

成本

2.8 3.0

1.4 1.5

0

0 100 200 300 400 500

时间（s）

(c)

0

0 100 200 300 400 500

时间（s）

(d)

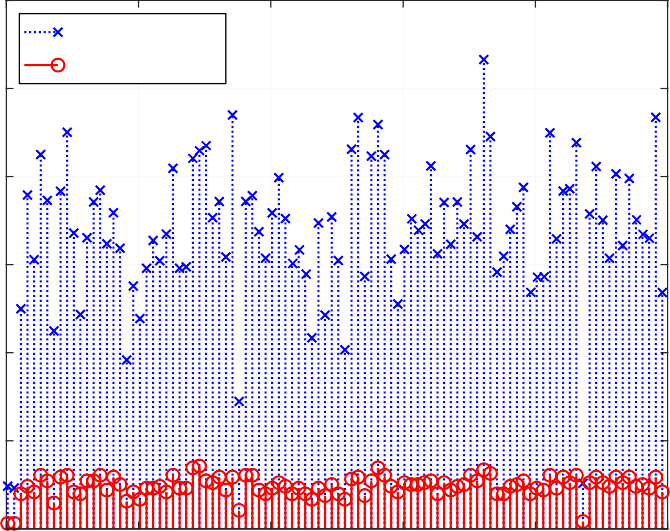
图 4-7 不同请求到达率下（λ = 50*,* 100*,* 150*,* 200），四种解决方案的总成本

择计算节点以确保 *Q* (*t*) 的稳定。可靠因子优先选择方案按 TF *Hi* (*t*) 的降序排列，然后选择排名最高的设备，再通过共同考虑因子和虚拟队列长度的稳定性来确定支付。基于确定性的解决方案将任务分配问题表述为每个时间槽的确定性优化，并通过 Lagrangian 分解求解。本节将随机选择（Random Selection, RS） 简称为 RS，可靠因子优先选择（Reliability Factor preferred, RF）简称为 RF，基于确定性的解决方案（Deterministic-based Poliocy, DP）简称为 DP。

图[4-7](#_bookmark73)（a）-（d）显示了在不同请求到达率下四种解决方案的总成本。根据图所示，所有曲线在仿真过程中都有波动，主要是由于网络环境状态的随机性。此外还观察到，所提策略在四种解决方案中实现了最低成本。DP 在大多数情况下优于其他两个基准。这是因为本章提出的方案考虑了随机性并被证明收敛到长期最优。DP 将相关问题视为忽略计算节点状态变化的确定性优化，因此产生的次优解决方案高于本章所提策略。当任务到达率增加时，所提策略和 DP 之间

的差异变得明显。在 RF 中，选定的具有高 RF 因子的工作节点总是有较低的成本。然而，没有考虑决策的最优性，RF 不能在选择工作节点时确保低执行成本。RS 表现最差，因为奖励是随机确定的。

280



Plasma

Single Chain

200

160

120

交易数

80

40

0

0 100

200

300

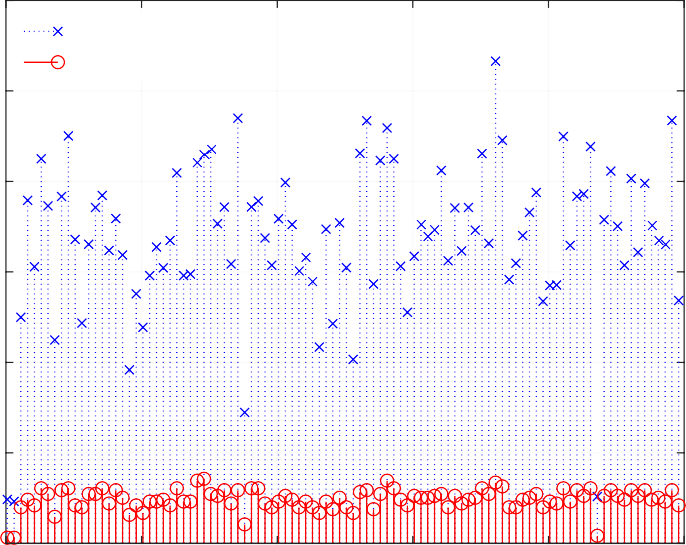
400

500

时间（s）

图 4-8 PoW 共识下即时吞吐量与时间的关系

280



Plasma

Single Chain

200

160

120

交易数

80

40

0

0 100 200

300

400 500

时间（s）

图 4-9 DPoS 共识下即时吞吐量 TPS 与时间的关系

区块链网络性能：本节将基于 Plasma 的区块链性能（简称为 Plasma）与当前广泛使用的基于单链的区块链性能（简称为 Single Chain）进行比较。图[4-](#_bookmark74)

[200](#_bookmark74)

[PoW](#_bookmark74)

[DPoS](#_bookmark74)

[160](#_bookmark74)

[120](#_bookmark74)

时延（s）

[80](#_bookmark74)

[40](#_bookmark74)

[0](#_bookmark74)

[0 32](#_bookmark74)

[64](#_bookmark74)

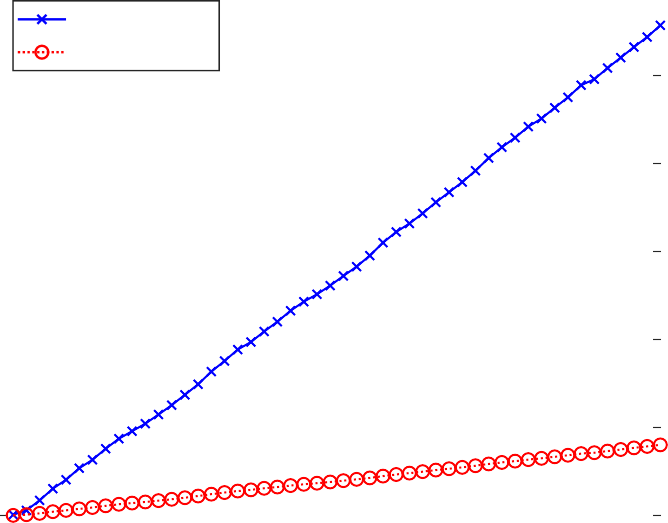
[区块数](#_bookmark74)

[96 128](#_bookmark74)

[1.8](#_bookmark74)

[图 4-10 区块批准时延与区块数的关系](#_bookmark74)

[\*104](#_bookmark74)



[Plasma](#_bookmark74)

[Single Chain](#_bookmark74)

[1.5](#_bookmark74)

[1.2](#_bookmark74)

[累计交易数](#_bookmark74)

[0.9](#_bookmark74)

[0.6](#_bookmark74)

[0.3](#_bookmark74)

[0](#_bookmark74)

[0 100 200 300 400 500](#_bookmark74)

[时间（s）](#_bookmark74)

[图 4-11 累计吞吐量与时间的关系](#_bookmark74)

[8](#_bookmark74)和[4-9](#_bookmark75)分别显[示了使用PoW 和DPoS 共识机制时的每个时间点的吞](#_bookmark74)吐量每秒交易数（Transaction [Per Second，TPS）。图4-8中，与 Single Chain](#_bookmark74) 相对应的数据点要比 Plasma 的[密集得多，即 Plasma 的吞吐量高于 Single Chain。这是](#_bookmark74)因为多个子链同时执行[智能合约，从而具有更高的吞吐量。当使用DPoS 时，也](#_bookmark74)观察到了类似的结果，这证实了 Plasma 在吞吐量方面的优越性。特别是，图[4-9](#_bookmark75)还显示了两

种解决方案的数据点均匀分布。这主要是因为生成块的时间间隔等于 5 秒。本节将批准延迟定义为生成块和批准之间的时间间隔。根据共识设置，一个区块在其第 7 个后继区块生成时将被批准。因为DPoS 的生成间隔为 5 秒，所以它的批准延迟稳定在 35 秒。然而，PoW 通过解决哈希谜题来生成区块，其生成频率由于高复杂度而低且不稳定。因此，PoW 的批准延迟高且不稳定，如图[4-10](#_bookmark76)所示。本节还比较了当使用DPoS 时基于 Plasma 的解决方案与基于单链的解决方案的累积吞吐量，并在图[4-11](#_bookmark77)中展示了结果。代表 Plasma 的蓝色曲线远高于基于单链的解决方案。结果表明，Plasma 中的并行处理可以显著提高吞吐量。根据上述结果，可以得出结论，基于 Plasma 的区块链与DPoS 共识机制更适合本章所提策略。

#### 本章小结

通过对边缘分布式计算的动态性特征及其所面临挑战的深入探讨，本章提出了一种基于智能合约的自适应计算卸载策略。此策略设计并部署了四个智能合约，分别用于工作节点注册、任务发布、任务分配以及奖励与惩罚的去中心化管理，实现了系统内部流程的自动化与去中心化控制。为了适应网络变化和动态性，本策略将任务分配过程视为一个随机优化问题，并据此设计了一套在线任务分配方案，能够及时且有效地将任务分配至适宜的工作节点。此外，策略中还包含了创新的奖励与惩罚机制，旨在确保奖励的个体理性及系统的可靠性，激励参与者的积极贡献并防止不诚实行为。本章还进行了理论分析，全面探讨了其队列稳定性、可靠性、防止免费搭车行为及系统的整体鲁棒性等关键性能指标。通过一系列仿真测试，本研究证实了所提出策略在最小化执行成本方面相较于现有最新解决方案具有明显优势。综上所述，本章的研究工作不仅为边缘计算领域贡献了新的研究视角，还展示了智能合约技术在提高计算效率与增强系统稳定性方面的巨大潜力，为未来的研究和应用提供了重要的理论基础和实践指导。

## 第五章 基于工作量证明的公平化卸载方法

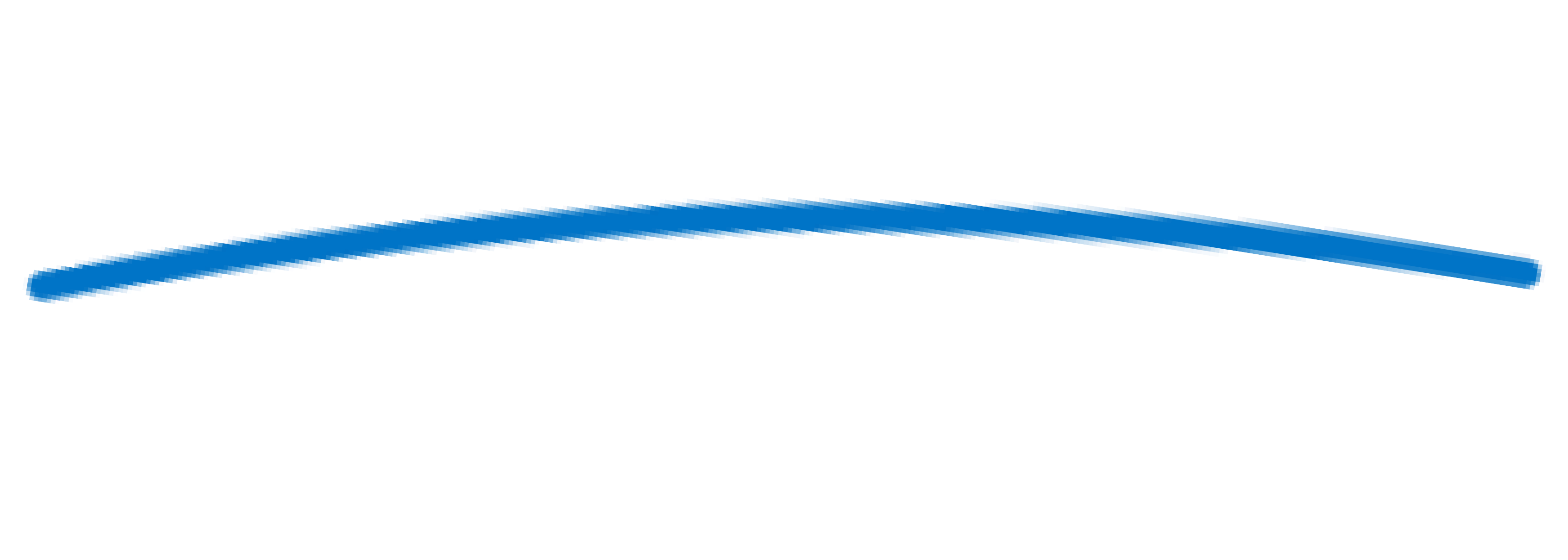
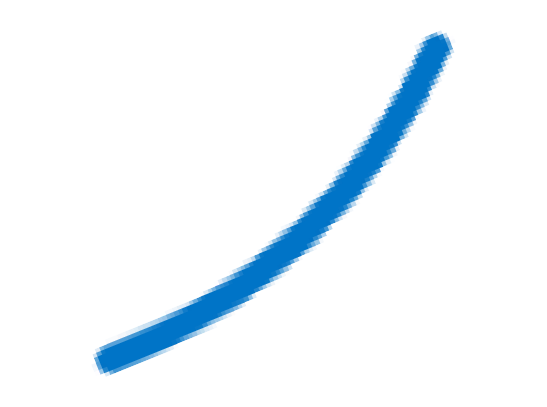
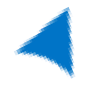
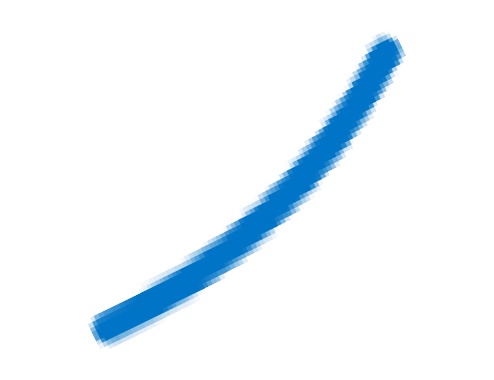
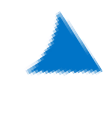
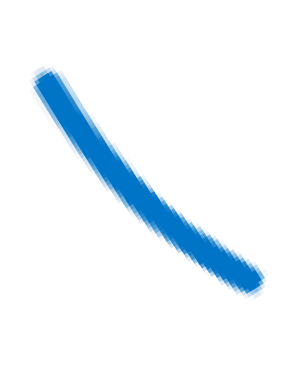
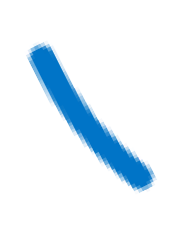
本章专注于边缘分布式计算在5G 技术推动下的发展及其面临的新挑战，尤其强调计算任务的分散化和复杂化对计算卸载公平性带来的影响。随着计算卸载日益依赖于网络终端设备间的协作，确保每个参与节点根据其贡献获得相应回报成为核心议题。公平性的确保不仅关系到计算任务的高效分配，也直接影响到各节点的参与积极性及整个系统的运行效率。针对这一挑战，本章提出了一种创新的基于工作量证明的公平化卸载方法。通过建立DAG 模型详细描述任务间的依赖性，进而采用基于DAG 的适应性区块链架构和多智能体强化学习策略，以优化计算卸载决策。该方法不仅增强了任务分配的透明度和公平性，促进了系统效率的提升和各参与节点的公平受益，还贯彻了Web 3.0 的理念，为构建一个更开放、更公平的计算生态系统打下了坚实的基础。

#### 引言

随着5G 技术的广泛应用以及智能设备处理能力的显著提升，网络终端设备如智能手机、汽车等已经具备了强大的通信、计算和存储能力[[72](#_bookmark219)]。这推动了 边缘计算朝着更加分散化和分布式的形态发展。计算任务现在可以分散到地理位置不同的网络端设备上执行，统称为网络计算节点（Network Computing Point， NCP）[[9](#_bookmark157)]。利用这些NCPs 协作完成计算任务，能够最大化网络的计算潜力，特别 是在通信、计算基础设施匮乏的偏远地区[[9](#_bookmark157)]、救灾现场[[73](#_bookmark220)] 以及战区[[74](#_bookmark221)] 等场景。

与此同时，随着计算业务的复杂化，传统的单一、原子化计算卸载策略已不再适用。任务间的耦合依赖性成为常见现象。这一特性对提升系统效率和性能也起到了关键性作用。例如，在智能交通系统中[[75](#_bookmark222)]，先处理车辆行驶状态信 息再生成交通流量控制策略，可以更精确地调整交通信号，优化交通流量，减少拥堵。此外，考虑任务间耦合依赖性的计算卸载策略，能有效减少冗余计算， 例如，图像的预处理（如降噪、缩放）和高级处理（如特征提取）步骤通常是紧密相关的[[76](#_bookmark223)]。通过在相同或临近的计算节点上调度这些具有依赖关系的任务， 可以共享中间结果，避免重复数据传输和处理，显著减少了总体的延迟和能耗。基于耦合的计算卸载还可增强任务并行性和系统扩展性。在智能制造领域[[77](#_bookmark224)]， 自动化生产线的任务通常包括多个相互依赖的子任务（Sub-Task, ST），例如实时监控生产流程、预测设备故障以及执行质量控制。将这自动化生产线的任务

**依赖性任务**



任务

发起

NCP5

NCP6

NCP2

NCP4

NCP1

NCP3

NCPn

**输出**

ST4

ST3

ST2

ST1

**输入**

**边缘网络**

图 5-1 具有依赖性任务的边缘分布式计算卸载示意图。上半部分展示了具有依赖性特征的任务，它可以分为多个子任务。下半部分表示边缘网络。其中，蓝色线表示数据流，红色线表示计算任务卸载过程。

映射到分布式的边缘计算网络中，可以灵活地调度各个子任务在不同的节点并行执行。不仅减少了数据在网络中的传输延迟，确保实时监控和响应的高效性， 而且通过并行处理，大幅提升了整个智能制造系统的处理速度和生产效率。

随着计算任务的复杂性、耦合性提高，单一节点难以独立完成所有任务，这要求不同的计算节点必须协同工作。然而，实现节点间高效协作的前提是确保计算卸载过程的公平性，即系统中的每个节点都应根据其能力获得相应的回报， 避免出现“强者通吃”的局面，即一个或少数节点占据大部分计算任务和资源， 而其他节点则处于边缘地位。这种公平性的追求与第三代互联网 Web 3.0[[78](#_bookmark225)] 的理念相符，后者强调构建一个开放、透明和去中心化的网络环境，确保每个参与者能公平贡献和受益。Web 3.0 的概念源于对隐私和所有权[[79](#_bookmark226),[80](#_bookmark227)] 日益增长的需求。与 Web 2.0 时代的中心化控制不同，Web 3.0 通过去中心化的方式，将网络的控制权分散到用户手中。这种去中心化不仅增加了网络的安全性和透明度， 还赋予了用户更多的自主权和控制权[[81](#_bookmark228)]。在这样的背景下，计算卸载的公平性 不仅是技术效率的问题，也是实现Web 3.0 愿景的关键一环。只有确保每个参与的节点能够根据自己的能力和贡献获得公平的回报，才能真正实现去中心化和用户赋权的目标。

挑战总结如下：

* + 1. 目前研究主要聚焦于对整个应用程序的计算任务进行卸载[[27](#_bookmark175),[82](#_bookmark229)]。然而，

将整体任务拆解为子任务进行串行、并行处理，能有效减轻单个设备的计算负担，如图[.5-1](#_bookmark81)所示。尽管已有研究针对特定场景提出了解决方案[[31](#_bookmark179),[83](#_bookmark230)]，但这些策略往往难以满足应用场景对子任务结构多样性日益增长的需求。

* + 1. 大多数现有的计算卸载策略采取了集中式设计思路[[84](#_bookmark231),[85](#_bookmark232)]，这不仅增加了系统遭受单点故障的风险，也可能降低了任务处理效率。因此，迫切需要开发出可靠的去中心化任务调度方法，同时确保任务以最优方式被分配和执行。
    2. 当将个人终端设备纳入计算卸载方案时，设计适当的激励机制变得尤为重要。以往研究常忽视为参与计算卸载的节点提供激励的必要性。虽然某些研究[[41](#_bookmark189)] 涉及到奖惩机制，但对于性能较低的设备却往往缺乏足够的激励。因此， 建立一个既公平又合理的激励机制，鼓励所有设备参与计算卸载，对确保边缘分布式网络中计算效率和公平性至关重要。

针对上述挑战，人工智能和区块链技术的发展提供了新的解决方案。深度强化学习已经证明在处理复杂任务结构和动态网络状态的学习中具有显著效果[[86](#_bookmark233),[87](#_bookmark234)]。同时，区块链技术的去中心化、透明数据结构为数据的可追溯性、不变性和不可逆性提供了保障。此外，Web 3.0 的理念促进了去中心化金融

（Decentralized Finance, DeFi）[[88](#_bookmark235)] 和去中心化自治组织（Decentralized Autonomous OrganizatioN, DAO）[[89](#_bookmark236)] 的概念的生长，给边缘分布式计算卸载融合 Web 3.0 提供了灵感。

基于以上考虑，本章提出了一种基于工作量证明的卸载方法，以实现计算任务分配的公平化。这一方法本质上与Web 3.0 的核心理念相契合，通过将任务分配过程融入到区块挖掘流程中，重新定义了计算卸载机制。首先针对边缘分布式计算应用的特性，构建了一个通用的DAG 模型，以刻画各计算任务之间的依赖关系和执行顺序。接着，设计了一个基于DAG 的适应性区块链结构，旨在通过工作量证明技术鼓励节点的广泛参与，也保证任务分配过程的透明性和公平性。此外，算法方面采用了演员-评论家（Actor-Critic）框架下的多智能体强化学习方法，优化去中心化的任务卸载决策，以提高整个网络的计算效率和响应速度。

主要贡献总结如下：

1. 本章针对存在依赖性关系的任务进行了研究，通过将子任务间的耦合关系建模为DAG，定义了一个旨在最小化任务完成时间的问题。
2. 为解决上述问题，本章开发了一个去中心化的多智能体环境，允许NCPs 自主决定是否承担特定子任务。随后，设计了一个基于演员-评论家框架的

分散式多智能体强化学习（Dispersed Multi-Agent Reinforcement Learning, DMA）算法，该算法适用于复杂且动态的环境，为解决上述问题提供了有效的方案。

1. 为激励更多的NCPs 参与计算任务并确保计算能力与奖励之间的公平分配， 本章将任务调度决策过程集成进区块链框架，并将计算卸载转化为一种区块挖掘活动。设计了一个包括一条中心链和多条小链的混合区块链结构， 其中，小链根据任务DAG 的拓扑特性而设计。为防止如自私挖掘[[90](#_bookmark237)] 等恶 意行为，还提出了特定的挖掘机制和主链确认机制。
2. 本章在实验部分将所提方法与四种其他方案进行比较。在任务卸载效率和激励机制的公平性等方面，所提方法展示了显著的优势。

本章的其余部分安排如下：第 5.2 节中描述了系统架构和网络模型。第 5.3

节展示了区块链系统架构。第 5.4 节对任务完成时间最小化问题建模。第 5.5 节中详细介绍了基于多智能体强化学习的任务调度算法。第 5.6 节分析、评估了实验的结果。最后，在第 5.7 节对本章进行总结。

#### 系统架构和网络模型

本节首先介绍研究的系统架构，然后描述了设计方案的网络模型。

请注意，除非另有说明，书法符号（例如 *N* ）表示集合，而黑板粗体（例如 A）用于表示集合的集合。本章使用的符号含义在表[5-1](#_bookmark84)中给出。

* + 1. 系统概述

对于大规模系统，由于通信和信息同步的成本高，针对整个网络中进行任务卸载是不可行的。因此，研究的系统架构中，所有网络节点根据它们的地理位置被分组为几个区域。NCPs 与区域成员共享计算资源。如图 [5-2](#_bookmark85) 所示，设计了一个双层架构，包括计算层和桥接层。

计算层：这一层包含了大量NCPs，配备了可编程执行环境和无线通信接口。这一层的节点执行三个功能：任务启动、任务执行和小链维护，这使得它们成为整个系统的核心。

桥接层：这一层的节点通常在地理上是固定的长期在线的，包括边缘服务器、基站等。这些桥接节点负责与其他区域通信。此外，它们与同一区域内的

表 5-1 符号

|  |  |
| --- | --- |
| **Notation** | 定义 |
| *N*, *N* | 一个 MNU 中 NCP 的个数和 NCP 的集合 |
| *K*, *K* | 子任务的数量和子任务的集合 |
| G | 任务链的 DAG 表示 |
| *Tk*, *Lk* | 子任务 *Tk* 及其数据传输量 |
| *Dk* | 子任务 *Tk* 的输出数据量 |
| *Ck* | 完成子任务 *Tk* 所需的 CPU 周期数 |
| *timemax*  *k* | 子任务 *Tk* 可接受的最大延迟 |
| *di f fk* | 执行子任务 *Tk* 的难度值 |
| *Pu* | NCP *u* 节点任务失败或状态不可用的可能性 |
| *bu*  *k* | 由 NCP *u* 所创建的、对应第子任务 *Tk* 的区块 |
| *C u*  *k* | 从创世区块到 *bu* 的链  *k* |
| C*u* | NCP *u* 接收的链的集合 |
| *Jk* | 申请执行子任务 *Tk* 的 NCP 的集合 |
| *pu* | 发送方 NCP *u* 的传输功率 |
| *hu,v* | 从 NCP *u* 到 *v* 的链路信道增益 |
| *lu,v* | NCP *u* 和 *v* 之前的距离 |
| α | 路径损耗指数 |
| *N*0 | AWGN 的噪声功率 |
| *Bu* | NCP *u* 的信道带宽 |
| *fu,k* | NCP *u* 分配给子任务 *Tk* 的每秒 CPU 周期数 |
| φ | 叔区块的数量上限 |
| S*,* A*,* R | 状态空间、动作空间和奖励 |
| *ru*(*k*) | NCP *u* 执行子任务 *Tk* 的奖励 |

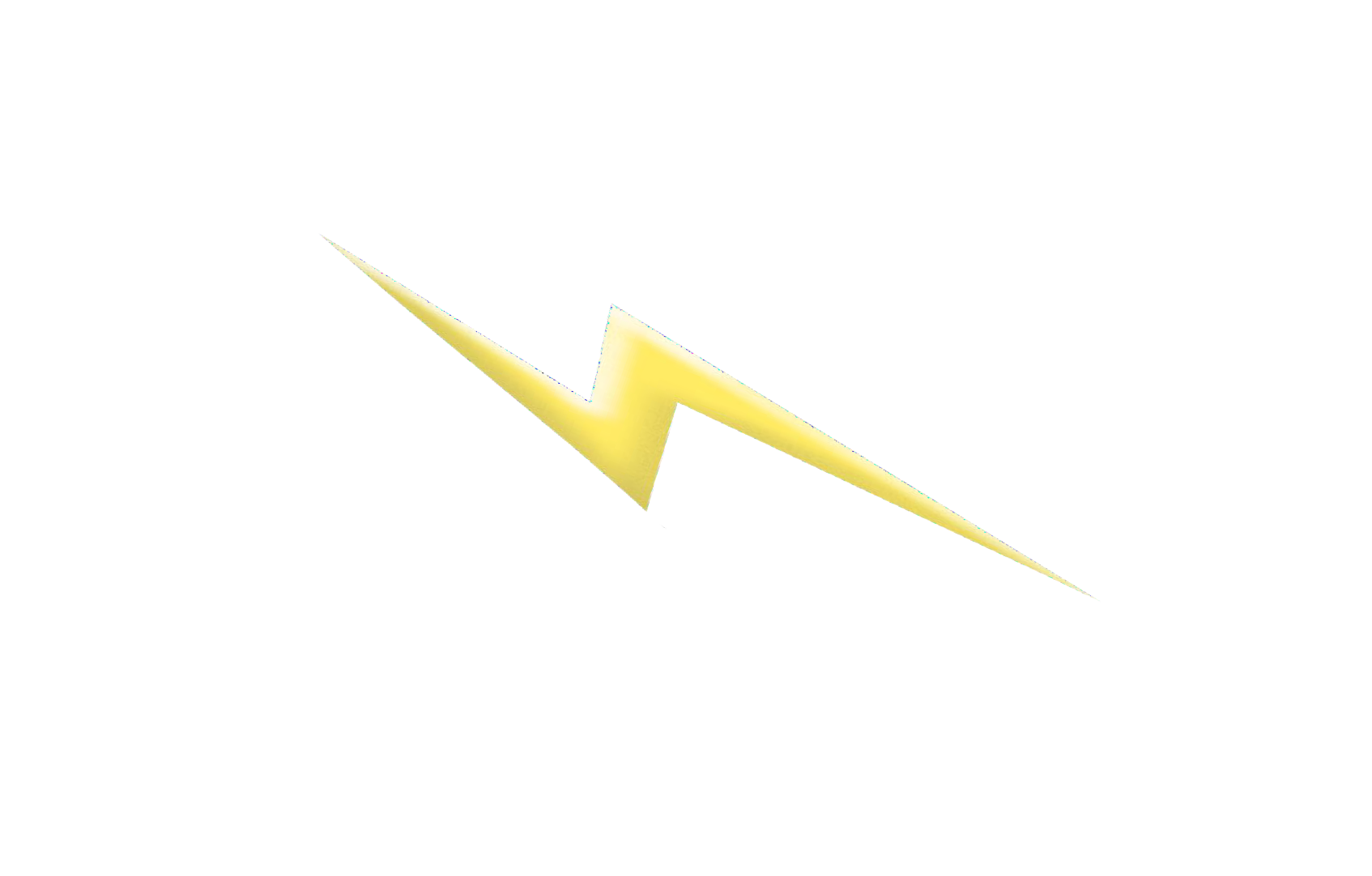
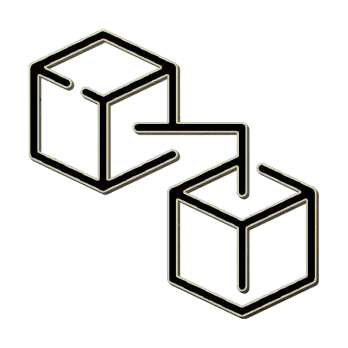
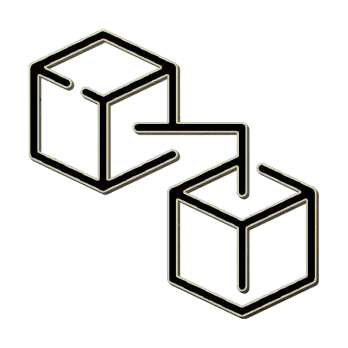
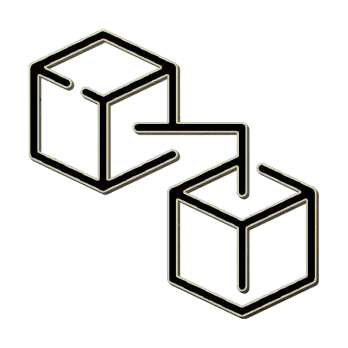
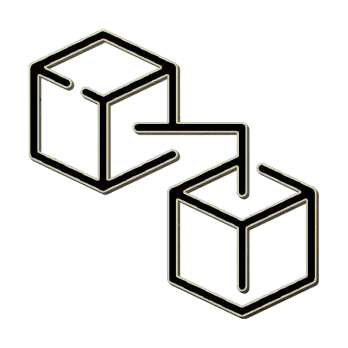
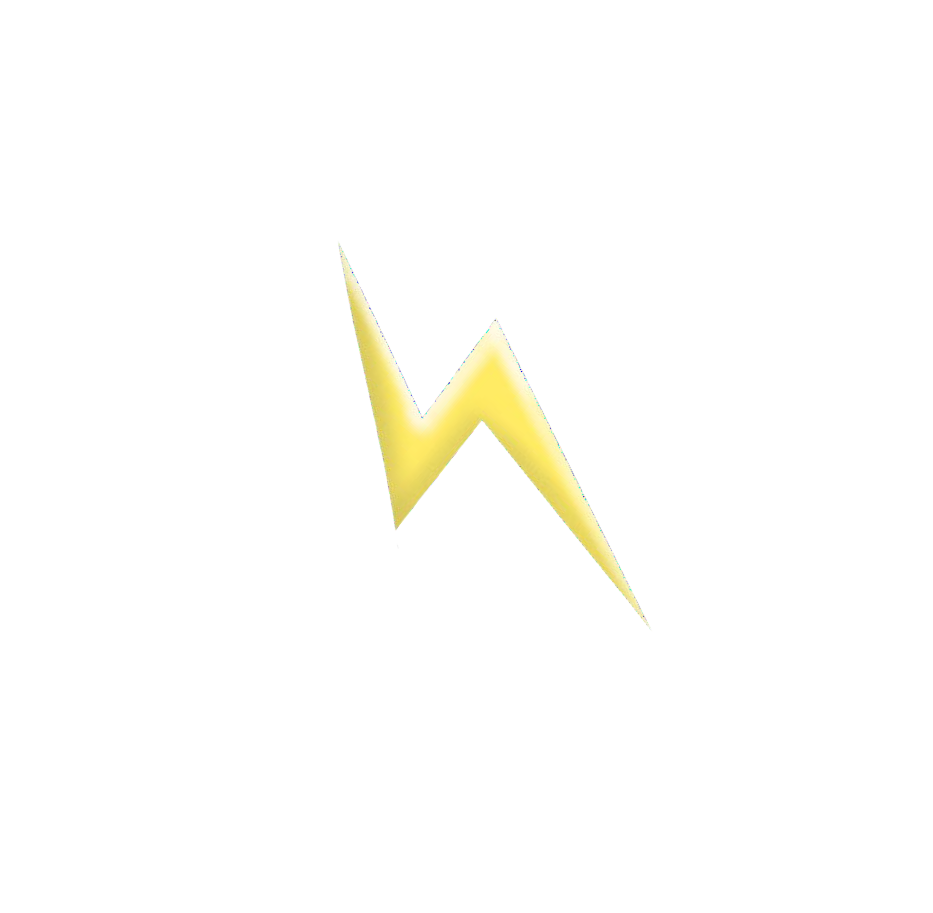
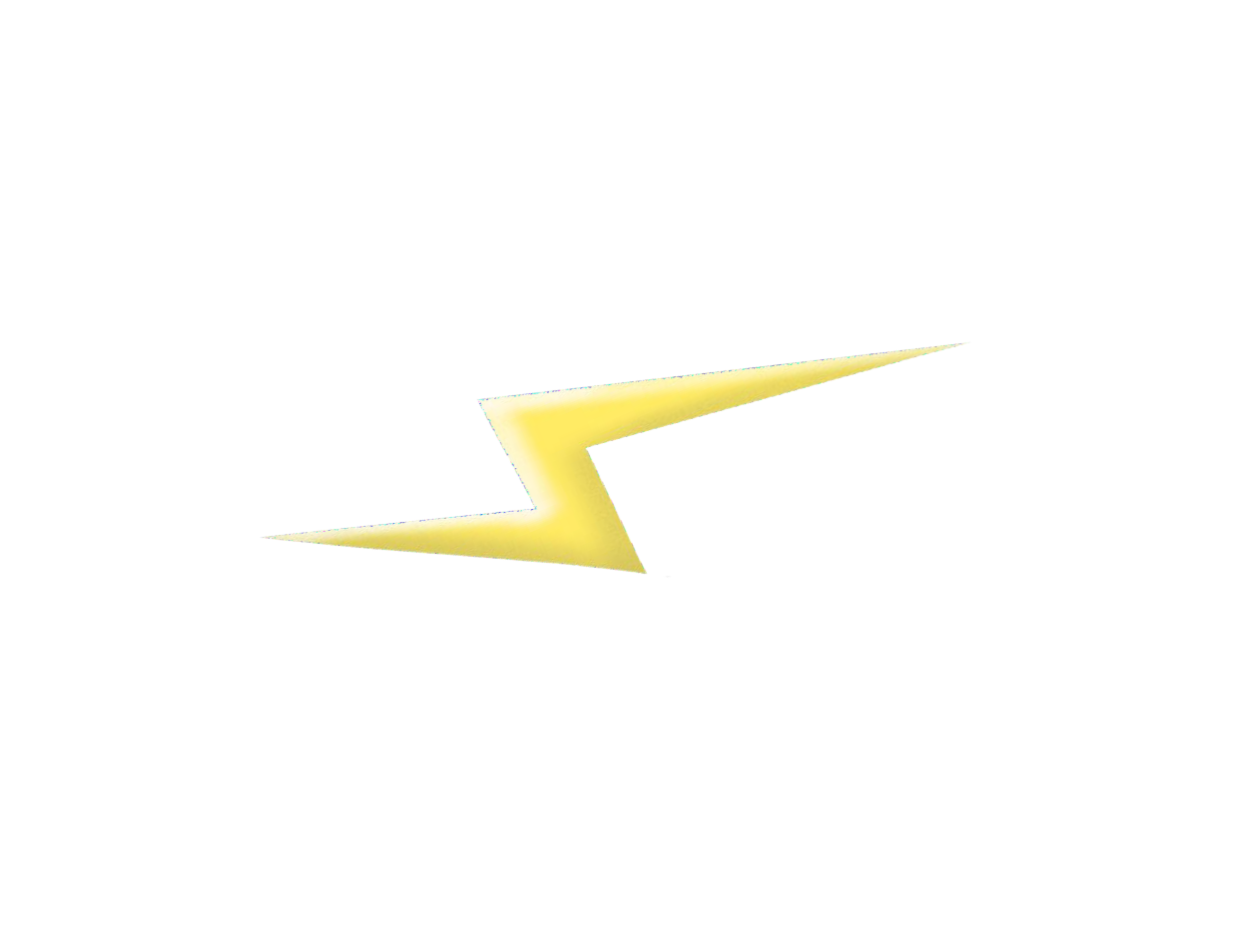
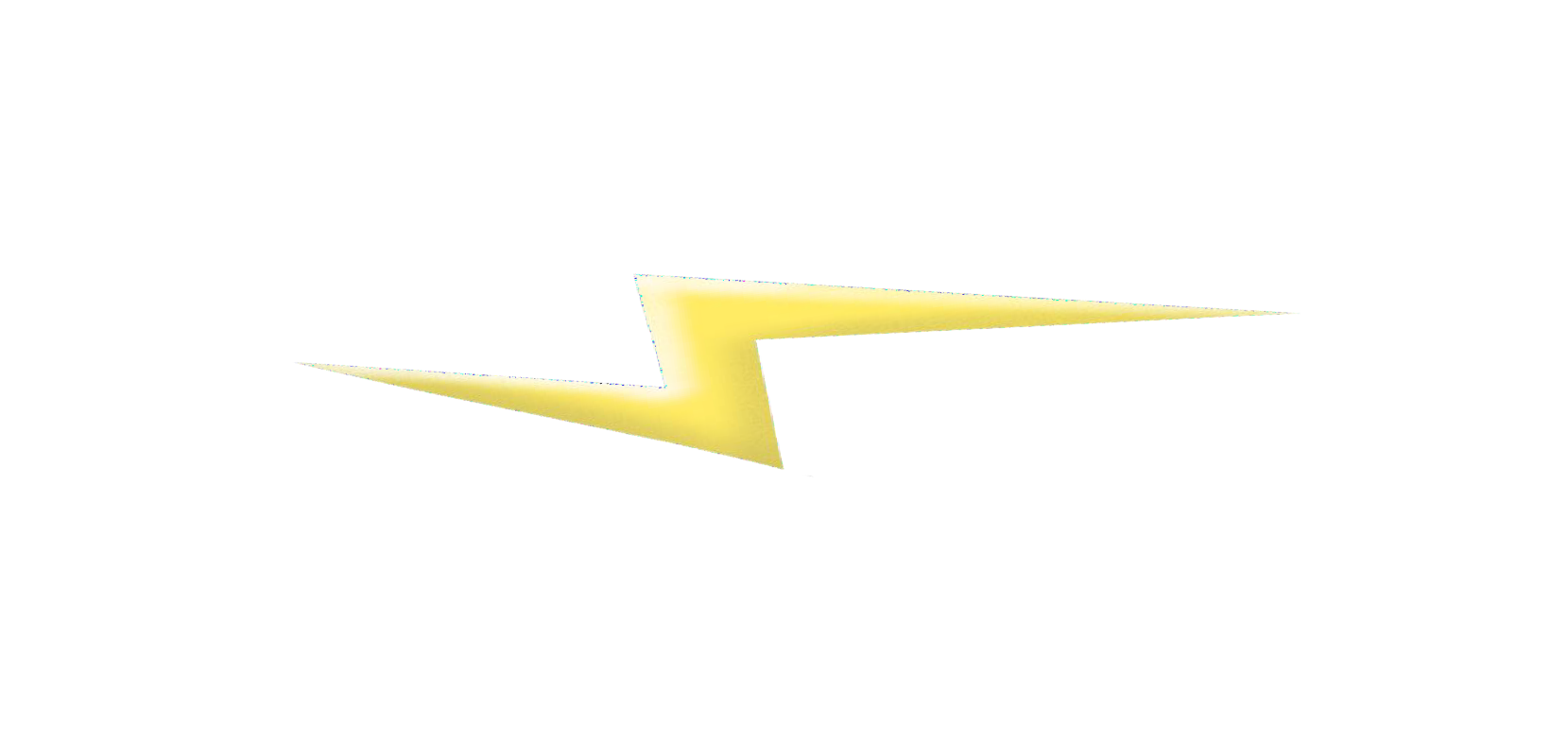
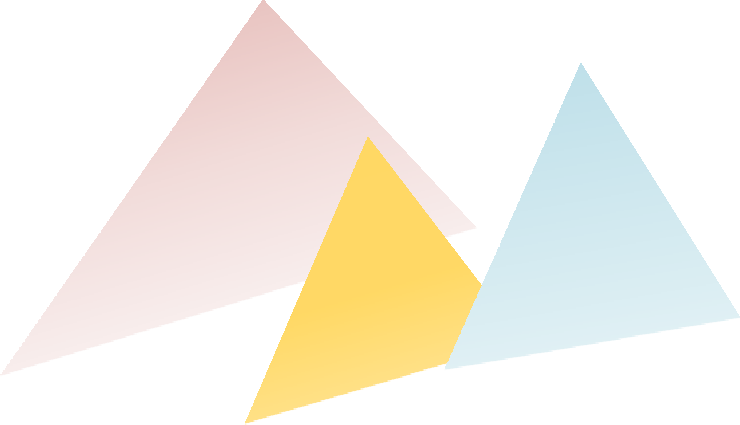
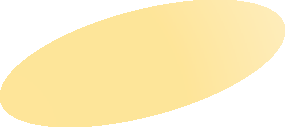
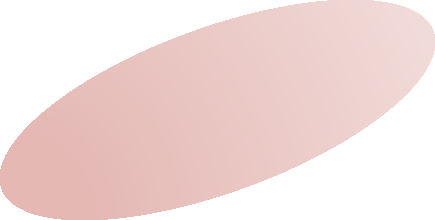
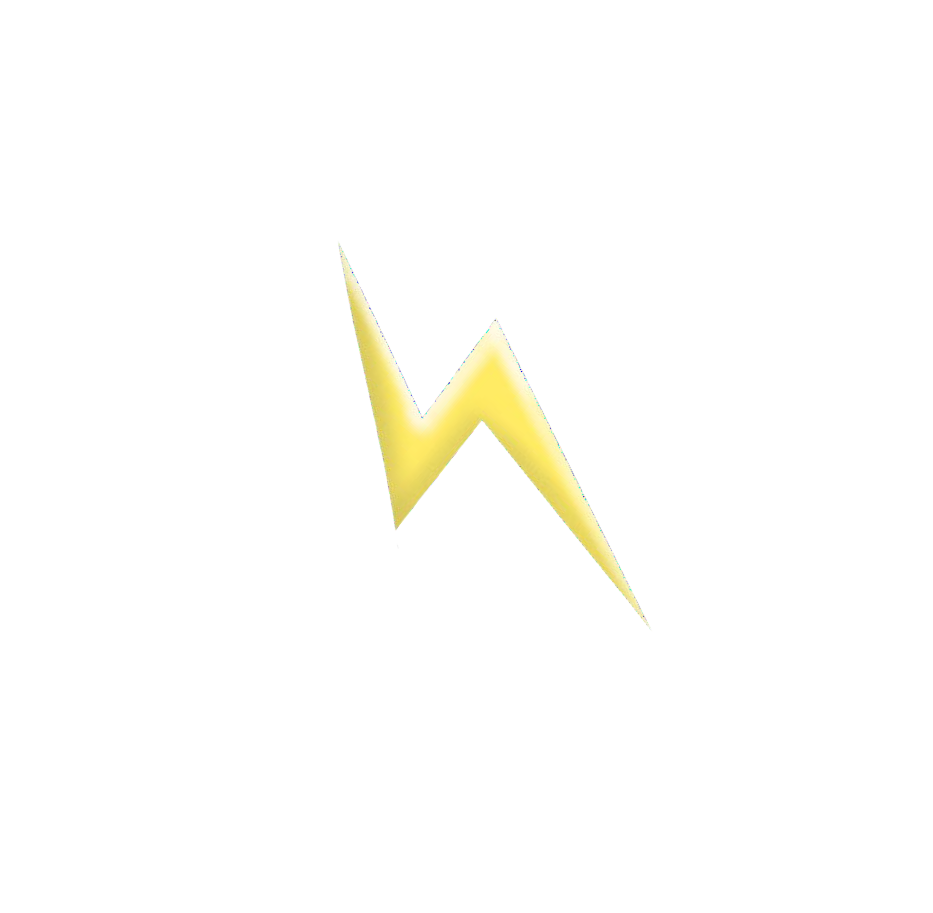
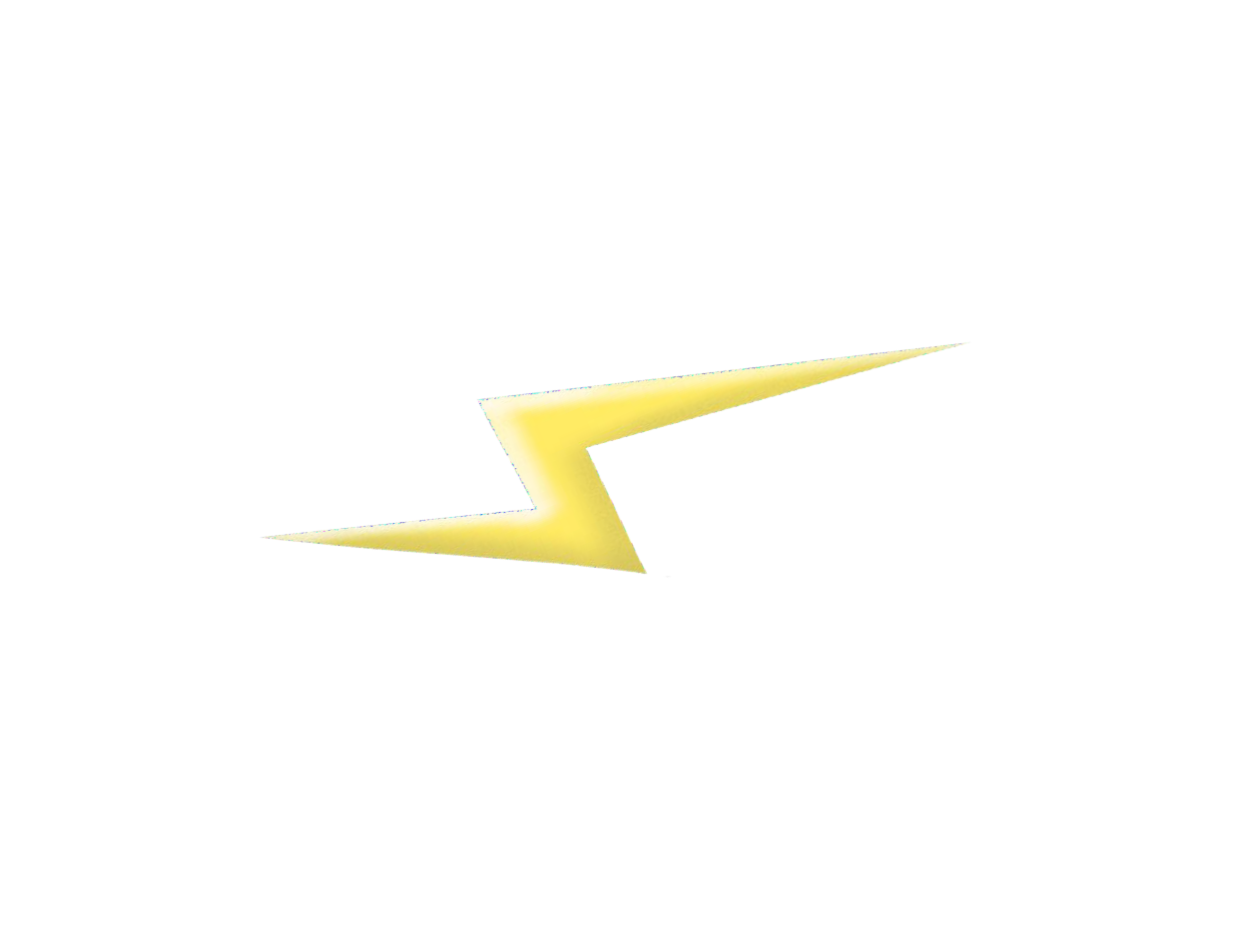
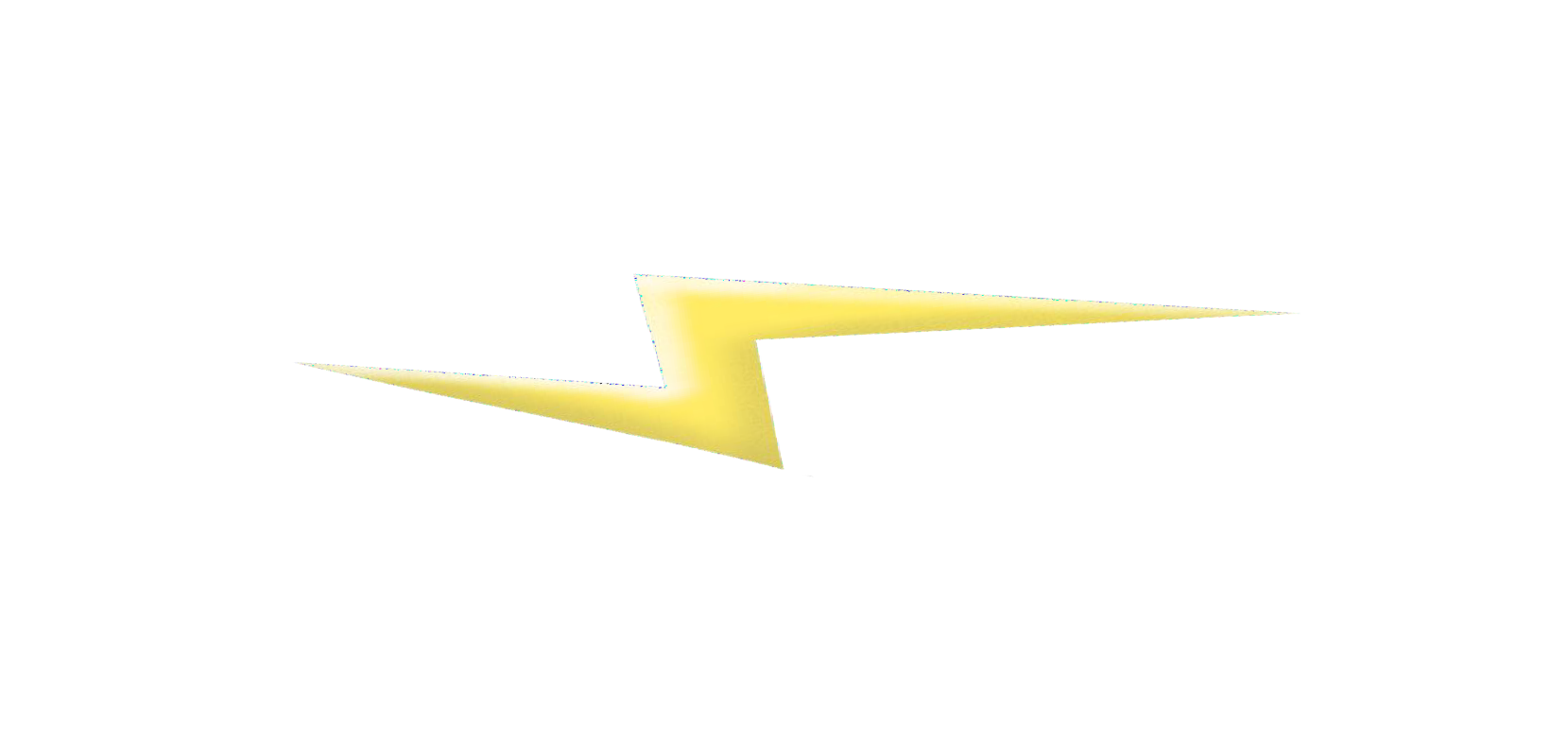
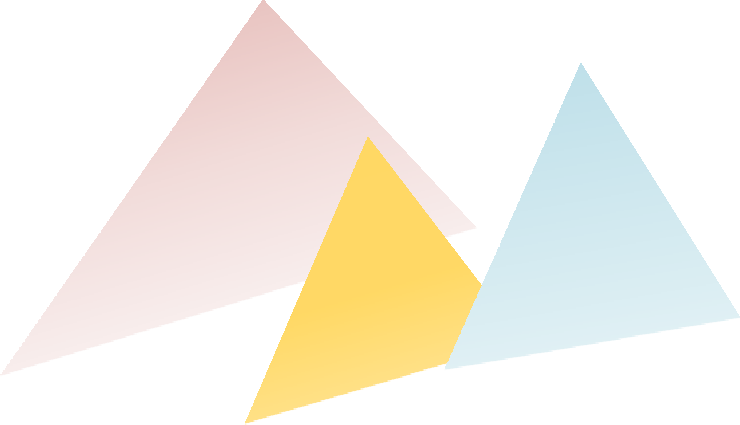
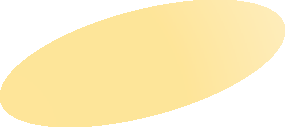
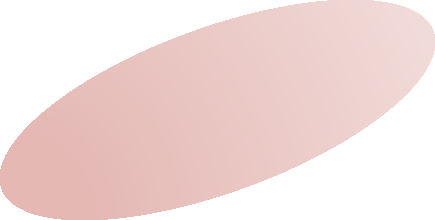
NCPs 直接连接。它们接收来自 NCPs 的请求，发布相应的子任务，并维护中心链。

如上所述，架构中有两种区块链：中心链和小链。中心链运行在每个桥接节点的上，负责记录所有小链的最终状态并为相应的 NCPs 分配奖励。每个小链专门为每个计算任务建立，由参与的 NCPs 维护。小链旨在任务分配和公平激励机制，记录一个任务的整个生命周期。此外，与 Plasma[[63](#_bookmark210)] 和区块链中其他第二层扩容方案[[91](#_bookmark238)] 中的侧链与根链的关系不同，小链不直接连接到中心链。每个任务完成后，小链会将整个链上每个区块的摘要信息上传到中心链，作为历史记录。

在系统中，假设当数据交换的双方在地理上足够接近时，它们将采用设备到设备（Device to Device，D2D）的无线通信技术来提高效率。

工作流程如下：

***桥接层* …**



***计算层***

***中心链小链***

### 区块链

 ***桥接节点***

 ***网络计算节点（NCPs） 区块链***

图 5-2 系统网络架构

* + - 1. 一个NCP 启动一个计算任务，并将请求发送到同一区域内的桥接节点。
      2. 桥接节点对任务进行预处理，并形成包含多个子任务的任务链。然后它将创建小链的第一个区块（即创世区块），包含任务信息和要求等。
      3. 收到创世区块后，同一区域的NCPs 决定是否根据 DMA 算法确定是否执行任务链中的下一个子任务。
      4. NCP 完成子任务，并作为矿工，负责根据挖矿机制生成新的区块，并广播它。一个新的区块表明相应的子任务已由该区块的矿工完成。
      5. 接收到新区块后，NCP 执行主链确认机制以选择要跟随的主链。然后， NCP 再次执行步骤 3 中，做出决定。最后，重复步骤 4 和 5，直到整个任务完成。
      6. 桥接节点根据小链的主链中的区块进行奖励分配，并通过发起新的交易将它们上传到中心链，这是网络内各方之间数字化数据的记录存证。
    1. 网络模型

每次的任务卸载都在一个区域内进行。为了简化，本章将一个最小网络单元（Minimal Network Unit，MNU）视为一个区域，正如图[5-2](#_bookmark85)中不同颜色的分组所示。

在 MNU 中，定义了由 *N* 个NCPs 组成的网络。NCPs 的集合用 *N* = {1*,* 2*,* · ·

* *, N*} 表示。不失一般性，假设每个任务有 *K* 个独立的计算密集型子任务，由*K* = {1*,* 2*,* · · ·*, K*} 表示。为了描述子任务之间的内在关系，本章考虑一个菱形DAG 作为依赖任务的通用模型，该模型在工业界和学术界广泛使用[[92](#_bookmark239)]。

DAG（由 G 表示）显示了 STs 的处理步骤。令 *Tk* = {*Dk,Ck,timemax, di f fk*}

*k*

描述第 *k* 个子任务。包含的元素如下：

* + *Dk* 表示输出数据的量，包括下一个子任务的系统设置和输入参数。
  + *Ck* 是执行第 *k* 个子任务所需的 CPU 时钟速率（每秒 CPU 周期数）[[93](#_bookmark240)]。
  + *timemax* 表示第 *k* 个子任务的可接受最大延迟。用于衡量子任务是否按时完成。

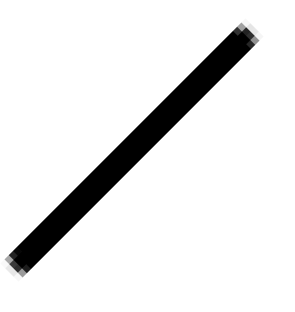
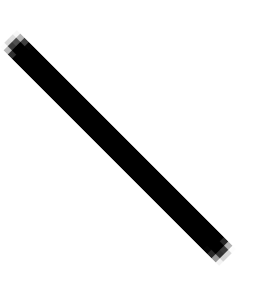
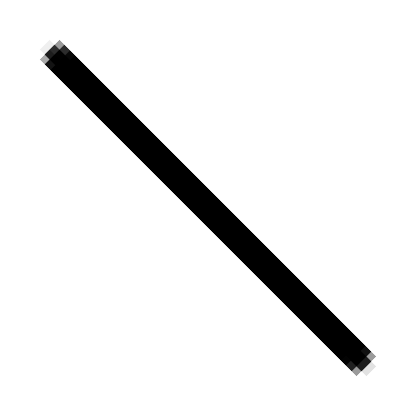
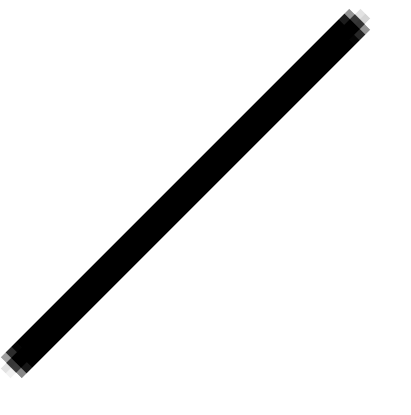
*k*

* + *di f fk* 显示执行第 *k* 个子任务的难度。它与 *Ck* 相关，即 *Ck* 越高，*di f fk* 越高。

*Tk* 到邻居 STs 的数据传输链路集合由 *Lk* 表示，其方向表示数据流动以及邻居 STs 的依赖关系。*Tk* 和 *Lk* 的集合分别由T 和L 表示。因此，有 G = {T*,* L}。此外，为了提供更通用的模型，向G 最后添加了一个虚拟任务，即 *TK*+1，因为计算任务的结果需要发送回请求设备。

为了清晰起见，本节以文献 [[94](#_bookmark241)] 中描述的VR 的计算任务为例介绍，如图[5-3](#_bookmark87)所示。整个任务被分割成几个子任务：预处理、工作负载分割、前景渲染、背景渲染和帧整合，其中前景、背景两个渲染部分可以同时进行。图[5-3](#_bookmark87)的上部， *ST*4 依赖于 *ST*3 的输出，并且也是 *ST*6 的依赖之一。此外，同一阶段的 STs（*ST*4 和 *ST*5）彼此独立，这意味着它们可以同时执行。

在分散网络中，每个任务可以在本地处理或卸载到一个或多个具有空闲资源的合适的NCPs 上。同时，考虑到计算节点的动态性，NCP *u* 可能会处于不可用状态或者任务失败，可能性用 *Pu* 表示。



***相同结构***

**阶段 5**

**阶段 4**

**阶段 3**

**阶段 2**

**阶段 1**

**E2**

***链接方向***

**F2**

**C2**

**B2**

***主链区块***

***叔区块***

***废弃区块***

**E1**

**F1**

**C1**

**B1**

**A1**

**D1**

**D3**

**D4**

**C3**

**B3**

***DAG结构***

***区块链***

**D2**

***工作流***

***背景渲染***

**5**

**ST :**

**ST2 :**

***预处理***

**ST1 :**

***源数据***

**ST6 :**

***帧整合***

**ST4 :**

***前景渲染***

**ST3 :**

***负载分割***

***任务链***

图 5-3 任务链以及 DAG 区块链设计

#### 区块链架构设计

* + 1. 基于工作量证明的区块链设计

为了实现系统可扩展性、公平的激励机制和可靠的分布式决策，本节提出了一个由中心链和小链组成的双层区块链结构。

中心链建立在所有区域的小链之上，支持系统的可扩展性，并为所有计算任务提供可追溯性能力。可以用具有认证功能的联盟链 Fabric[[95](#_bookmark242)] 框架来搭建， 并由桥接节点维护。现有联盟链的主要功能[[96](#_bookmark243),[97](#_bookmark244)]，如交易记录、实用拜占庭容 错共识和访问控制机制等，足以满足系统设计需求。并且随着场景的扩大，参与者的增多，可以轻松扩展到数百个节点。

小链为任务卸载提供支持。每个小链对应一个单一的计算任务，并由一个桥接节点和参与该任务的计算节点维护。小链与计算任务共享相同的DAG 结构。

实际上，图 [5-3](#_bookmark87) 的下部是一个小链的示意图。实线的区块结构与 DAG 任务相匹配，即 {*ST*1*, ST*2*, ST*3*, ST*4*, ST*5*, ST*6} → {A1*,* B1*,* C1*,* D1*,* E1*,* F1}。每个区块代表一个

子任务完成的声明，区块之间附着的方向对应于子任务之间的依赖性。

具体来说，如果区块 B 附着于区块 A 后，B 将 A 视为其父区块。反过来说， B 是 A 的子区块。值得注意的是，单个子任务可能有多个前驱/后继任务处于同一阶段，导致产生拥有多个父区块或子区块的区块。例如，F1 有 D1 和 E1 作为父区块，C1 是 D1 和 E1 的父区块。

此外，本章采用改进后的PoW 作为小链中的共识机制。在传统的PoW 中，

矿工需要解决复杂的哈希难题以获得挖矿权，这将带来巨大的计算资源消耗。在所提方法中，创新性地将浪费能源的挖矿过程转变为子任务计算过程，只有完成计算任务的 NCPs 应该有权生成区块并广播它。

由于PoW 共识的公平竞争特性，单个子任务可以同时由多个 NCPs 竞争承担，导致一个子任务会有多个候选区块，例如 B1、B2 和 B3 都代表 *ST*2。然而， 相同的子任务最终只能对应一个唯一的区块。因此，那些没有被随后的区块直接附着的区块被定义为叔区块（Uncle Block）。此外，为了节约计算资源，必须对叔区块的生成设定限制。定义 φ 为与给定子任务相关的叔区块的最大数量。不属于主链且不是叔区块的区块被定义为废弃区块。例如，如果 φ = 1，B2 是叔

区块。B3*,* C2*,* D3 是废弃区块。注意 B2 是叔区块而非 B3。这个决定由下一个区

块的矿工做出，他通常考虑除了父区块外，首先接收到的 φ 个区块作为叔区块。由于叔区块的存在，DAG 区块链将如图[5-3](#_bookmark87)中虚线部分所示分叉。因此，选

择主链以进行区块链扩展至关重要，即主链确认机制。

在传统的区块链中，例如比特币[[52](#_bookmark200)]，采用最长链规则，主链是最长的链。然而，在实际应用中应该考虑两种情况：

情况 **1**：最长链规则意味着只有最快生成的区块会被采纳进主链，其他区块将被丢弃。通常，具有最高计算能力的NCP 可以创建这些区块。这将使其他NCP 的努力无用，可能导致它们长期失去参与的动力。在这种情况下，一旦强大的NCP 离线或遇到网络波动，即将到来的任务将无法及时处理，从而损害整个系统的鲁棒性。

情况 **2**：如果NCP 实施自私挖矿，即矿工不会广播生成的区块，直到自己创建了足够的区块链。在这种情况下，秘密链有很高的可能成为主链。因为秘密链的生成过程中没有传播延迟，这使得这种方法对那些分享计算结果的诚实矿工不公平。

综上所述，仅采用最长链规则将不会激励设备参与并破坏系统公平性。为

开始

**否**

**是**

执行该任务？

通过挖矿机制产生新区块

通过P2P网络传播区块

等待其他节点的新区块

完成计算任务

**DMA** 算法决策

NCP 收到子任务

整个任务是否

完成？

**否**

**是**

结束

接收到响应该子任务的多个区块

等待从其他节点接收到的新区块

继续进行下一个子任务

**通过主链确认机制确认主链**

*（其他区块成为叔区块或者被抛弃)*

图 5-4 区块链扩展流程图

了清晰地描述区块生成过程并进一步确保系统的公平性，本节提出了挖矿机制和主链确认机制。

为了促进更直观的理解，在深入解释这两种机制之前，创建了一个流程图来说明小链扩展的过程，如图 [5-4](#_bookmark90) 所示。

挖矿机制

* + - * 完成子任务的矿工生成相应的区块，并通过P2P 网络广播。其他计算节点验证收到的区块，并决定是否在该区块之后继续挖矿，前提是该区块不处

于最后阶段。

* + - * 每个区块包括两个组成部分：区块头和区块账本。区块头存储父区块的哈希，以及最多 φ 个叔区块的哈希。此外，矿工信息、相应子任务 ID 和时间戳也记录在区块头中。区块账本中存储的任务完成验证因子将用于验证此区块的可靠性。

主链确认机制

* + - * 受到 Ghost 协议[[98](#_bookmark245)] 的启发，本节设计了一种主链确认机制，以保障这些矿工的利益，并进一步激励边缘设备参与。与 *Tk* 对应并由 NCP *u* 挖掘的区块定义为 *bu*。值得注意的是，第一个区块和最后一个区块由桥接节点 *u*0 创建

*k*

（即 *bu*0 和 *bu*0 ）。从创世区块到 *bu* 的链表示为 *C u* = ｛*bu*0 *,* · · ·*, b*ν *,* · · ·*, bu*u，

*k*

*k*

1

*K*+1

*k*

*k*

1

*k*−1

*k*

其中 ν 表示某个 NCP ∈ *N* 。特别地，把 *di f fk* 作为 *b*ν 的权重。随后，*C u*

的权重（即 *W eight* （*C u*））可以通过包含区块及其叔区块的总权重来计算，

如方程[5-1](#_bookmark91)所示：

*k*

*W eight*(*C u*)=

*k*

∑*k*

2(*di f fi*∗1 + *di f fi*−1∗*U ncle* (*b*·*i*))*, k>* 1

(5-1)

0*, k* = 1

*i*=

其中 *U ncle* （*b*ν ） 是 *b*ν 中叔区块的数量，它可以从 *b*ν 的区块头中获取。值

*k*

*k*

*k*

得注意的是，区块链中叔区块的数量的大小与链的权重增加相关联。而权重的增多则表明已经有大量的算力被投入到这条链中，因此该链具有更高的安全性。因此，权重最大的链将被节点认为是主链。

* + - * 设计的主链确认机制如算法[5-1](#_bookmark93)所述。接下来。本节从NCP *u* ∈ *N* 的角度描述这一过程。当NCP *u* 需要确认主链时，它首先将接收到的链集合打包为C*u*。接着它为每条链计算 *W eight* (·)。为方便起见，*Ck*· 的权重在伪代码中写作 *WCk*· 。然后，*u* 根据 *WCk*· 的降序排序C*u* 中的链。如果几条链有相同的权重，先到达的区块有优先级。最后，C*u* 中的第一条链被视为主链。
    1. 安全性分析

BIDC 框架由一个中心链和若干小链组成。中心链采用传统的联盟区块链结构，以 Fabric 等平台为例。鉴于这种传统框架，中心链的安全性已经很成熟，不需要进行深入的分析。相比之下，安全性的重点转移到了小链上。

算法 **5-1:** 主链确认机制

输入： 收到的链集合 C*u*;

输出： 主链;

1: **for** 每个候选链 *Ck*· ∈ C*u* **do**

2: 初始化 *Ck*· 的权重 *WCk*· ← 0

3: **while** *k >* 1 **do**

4: 获取 *b*·*k* 的叔区块数 *U ncle*(*b*·*k*)*, b*·*k* ∈ *Ck*·

5: 计算 *b*·*k* 及其叔区块的总权重

（ ）

*WCk*· ← *WCk*· + *di f fk* ∗ 1 + *di f fk*−1 ∗ *U ncle*(*bk*· )

6: 更新 *k* ← *k* − 1

7: **end while**

8: 记录 *WCk*·

9: **end for**

10: 按 *WCk*· 降序排列 C*u* 及相应区块到达时间的顺序 *S ort*(C*u*)

11: **return** C*u*[0]

小链采用了修改后的PoW 机制，虽然保留了区块链技术固有的核心安全属性，但将计算工作量重新分配到具有实际效用的任务上。而这一改动实际上并不削弱传统区块链系统的基础安全原则。下面从三个角度详细对该点进行详细阐述：

抵抗常见攻击：小链能够抵御常见的区块链安全威胁，包括双花攻击、大多数（51%）攻击和女巫攻击。通过要求 NCPs 执行有用的计算任务以获得区块添加的特权，小链进一步提高了执行恶意行为的成本和复杂度。这一策略对潜在的恶意节点起到了威慑作用。

数据完整性：通过使用密码学的哈希函数以及共识机制，子链上数据的完整性得以保证。每个区块通过其自身数据内容和前一个区块的哈希生成的独特哈希，不可变地连接到其前导区块上。这种结构使得未经授权的数据修改在计算上变得不可行。此外，虽然设计的PoW 变体将获得挖矿权从解决抽象数学难题转变为实际计算任务的过程，但保留了关键的共识理念：要求大多数 NCPs 的认可才能将一个区块添加到区块链上，并成为主链的延申。

去中心化：小链依赖于基于PoW 的共识，促进了所有 NCPs 在执行计算任务中的公平参与框架。这种平等主义方法进一步促进了去中心化，从而增强了网络的安全性。

#### 优化问题建模

在分散网络中，保证任务的及时性是首要任务。从宏观上看，计算任务的整体完成时间主要由传输延迟（NCPs 之间的传输延迟）和计算延迟（每个 NCP 执行子任务的时间）组成。

本节首先分析传输延迟和计算延迟，然后对组合优化问题进行公式化表述。请注意：根据实验结果，BIDC 中的区块 *b*·*k* 大小仅为 1-3KB，与 *Dk* 的大小

相比太小，因此可以忽略区块传播时间；*Tk*+1 是一个虚拟任务，因此传输和计

算成本均被视为零。

传输延迟**:**

传输延迟取决于数据大小和传输速率。为了充分利用计算资源，完成 *Tk*

（*k* ≤ *K*）的NCP *u* 将输出数据 *Dk* 传输给附近申请下一个子任务的NCPs（记为*Jk*）。假设每个设备都有一个单天线用于上行通信，并且 *Dk* 通过无线组播通信在这些设备之间传输。当采用无线模式并且双方使用独占频段（也可以被视为采用多路复用的无线模式的上行频段）时，带路径损耗和瑞利衰落的信道模型的信噪比（Signal-to-Noise Ratio, SNR）为：

*u,*

*SNR*

*u,v* =

*pu* ∗ *hu,v* ∗ *l*α*v N*0

(5-2)

其中 *pu* 表示发送者 *u* 的传输功率，*hu,v* 表示从 *u* 到 *v* 的信道增益，*lu,v* =∥ (*xu, yu*) *,* (*xv, yv*) ∥ 是NCP *u* 和 *v* 之间的距离，α 是路径损耗指数。背景噪声被认为是具有噪声功率 *N*0 的加性白高斯噪声（Additive White Gaussian Noise，AWGN）。

根据香农公式，从NCP *u* 到 *v* 的可用传输速率表达为：

*rateu,v* = *Bu* log2 （1 + *SNRu,v*） (5-3)

其中 *Bu* 表示NCP *u* 的信道带宽。

因此，给定 *Dk* 的数据大小 *S ize*(*Dk*)，传输延迟可以通过以下公式得到：

*tk,u,v* = *S ize*(*Dk*)

(5-4)

*trans*

*rateu,v*

计算时延**:** 计算延迟主要取决于计算要求和NCP 内部处理器的能力。当 *Tk* 在NCP *u* 中执行时，*u* 的全部计算资源将被用于计算任务[[99](#_bookmark246)]。假设 *fu,k* 是 *u* 分配给 *Tk* 的每秒 CPU 周期数。那么，计算引起的延迟可以表示为：

*k,u comp*

*t*

= *Ck fu,k*

(5-5)

联合优化问题：总体时间消耗是所有子任务的传输和计算时间之和。文献

[[100](#_bookmark247)] 中的作者采用所有子任务的线性求和作为 *ttotal*。但他们忽略了DAG 任务中的并行计算特性，即几个子任务可以并行执行，例如图[5-3](#_bookmark87)中的 *ST*4 和 *ST*5。

为了准确捕捉DAG 的特性，采用递归方法来描述 *ttotal*。集合 *Prek* 被用来描述 *Tk* 的前驱（例如 *ST*4 和 *ST*5 是 *ST*6 的前驱）。假设 *Tk* 的开始时间为 *Start* (*k*)， 且 *Start* (1) = 0。那么，整体完成时间可以表示为：

*ttotal* = *Start*(*K* + 1)

｛ ′ *k*′*,u*

= max

Start(*k* ) + *tcomp* + *ttrans* ; *k*

∈ *PreK*+1

*k*′*,u,v* ′

� (5-6)

综上所述，组合优化问题可以表述为：

min *ttotal* (5-7a)

**s.t.** *U ncle* (*b*·*k*) ≤ φ*,* ∀*k* ∈ *K fu,k* 0*,* ∀*u, k* ∈ *N , K* (5-7b)

#### 任务调度算法设计

在优化问题 eq.[5-6](#_bookmark95)中，每个NCP 独立地为每个子任务做出计算决策，以最小化计算任务的总完成时间。然而，不同NCP 的决策相互依赖，随着NCP 数量的增加，问题的计算复杂性也在上升。为了解决这个复杂问题，本节采用了深度强化学习的手段。将优化问题转化为MDP，以表示每个NCP 的决策过程。

* + 1. 学习问题的表述

在多智能体强化学习方法中，每个NCP 被视为一个智能体。为了获得最佳奖励，每个NCP 通过与环境和其他智能体的交互来学习自己的决策策略。元组

⟨R*,* S*,* A⟩ 用于表示MDP，其中 R 是奖励，S 是状态空间，A 是动作空间，通常R

由 S 和 A 获得，即 R ← S × A。下述内容将详细介绍算法中MDP。

状态空间：假设在主链中生成第 *k* − 1 个区块的NCP 是 *u*∗。第 *k* 个子任务的状态空间包括两个主要部分：

* + - * *u*∗ 和 *u* 之间的地理距离。*u*∗ 的地理位置 (*xu*∗ *, yu*∗ ) 可以在 *bu*∗

*k*−1

的区块头中

找到，因此NCP *u* 可以计算距离 *lu,u*∗ 。

* + - * 第 *k* 个子任务的属性。在每个计算任务开始时，桥接节点会将所有子任务的信息通过创世区块广播给NCPs， 如： 每个子任务的四个属性： *Dk,Ck,timemax, di f fk*。

*k*

NCP *u* 的状态空间可以描述为 *Su*(*k*) = ｛*lu,u*∗ *, Dk,Ck,timemax, di f fk*u。因此，

*k*

系统状态空间为

.

*S* (*k*) = {*S*1(*k*)*, S*2(*k*)*,..., SN* (*k*)}

动作空间：每个NCP 自行决定是否执行第 *k* 个子任务。*au*(*k*) ∈ {0*,* 1} 表示计算决策。具体来说，如果NCP *u* 决定执行第 *k* 个子任务，则 *au*(*k*) = 1，反之*au*(*k*) = 0。关于第 *k* 个子任务的NCP *u* 的动作空间是 *Au*(*k*)，且 *au*(*k*) ∈ *Au*(*k*)。因此，系统动作空间可以定义为

*A* (*k*) = {*Au*(*k*)*, A*2(*k*)*,..., AN* (*k*)}

奖励函数：系统的目标是最小化总完成时间，如eq.[5-6](#_bookmark95)所述。然而，当决策时，每个NCP 并不知道整个计算任务的总完成时间。此外，优化目标中的整体完成时间并非所有子任务完成时间的简单总和。因此，使用子任务的时间消耗作为即时奖励并不理想。为了应对这种情况，本节为不同类型的NCP 设计奖励函数，以指导它们对每个子任务的决策方向。

* + - * 对于生成主区块或叔区块的NCP，提供积极奖励以激励它们保持一致的决策方向。奖励表示如下：

*r* (*k*) = β*u,k* ∗ *di f f*

(5-8)

*u tk,u,v* + *tk,u t*

*trans comp*

其中 β ∈ {0*.*5*,* 1} 取决于NCP *u* 是在主链上还是非主链上。如果 *bu* 是一个主区块，β*u,k* = 1。如果 *bu* 是一个叔区块，β*u,k* = 0*.*5。根据 eq.([5-8](#_bookmark98))，执行困难任务并以低延迟生成主区块的NCP 将获得更多奖励。

*k*

*k*

* + - * 对于生成废弃区块的NCP，给予消极奖励以防止它们生成废弃区块。因为这种不受鼓励的行为会耗费它们的计算资源，而不会对计算任务有所贡献。设计的奖励函数如下：

*r* (*k*) = − β*u,k* ∗ *di f f*

(5-9)

*u* 1*/Tmain* − 1*/Tsel f t*

其中 *Tmain* 是相应主区块的时间消耗，*Tsel f* 是这个废弃区块的延迟。与叔区块一样，β*u,k* = 0*.*5。这个函数表明，当它的延迟接近主区块的延迟时， 惩罚相对较少。总的来说，对于废弃区块，奖励始终是负的。

* + 1. 分散式多智能体强化学习算法

使用 π = {π1*,* π2*, ...,* π*N* } 来表示智能体策略的集合， 相应的参数集合是θ = {θ1*,* θ2*, ...,* θ*N* }。每个智能体 *u* 旨在通过更新参数θ*u* 来获得最优策略πθ∗*u* 。最优策略定义为

πθ∗*u* = argmax *J*(θ*u*) (5-10)

θ

*u*

其中 *J*(θ*u*) 是目标函数，如下所示：

*J* θ E

＿ *K* γ*k*−1 *r*

*k* ＿ (5-11)

其中 γ 是折扣因子。

( *u*) =

θ*u* ∑

*k*=1

* *u*( )

环境

*s*1 (*k* )

*a*1 (*k* ) *r*1 (*k* )

演员

目标网络

**'

1

软更新

评估网络

**

1 更新

*a*' (*k* )

1

*a*1 (*k* )

策略

梯度

评论家

……

目标

网络

*Q* '

1

软更新

评估

网络

*Q*

1

*Q*

1

*Q*'

1

*Q*

1

更新

损失函数

智能体**1(NCP)**

智能体 **2**

智能体 **N**

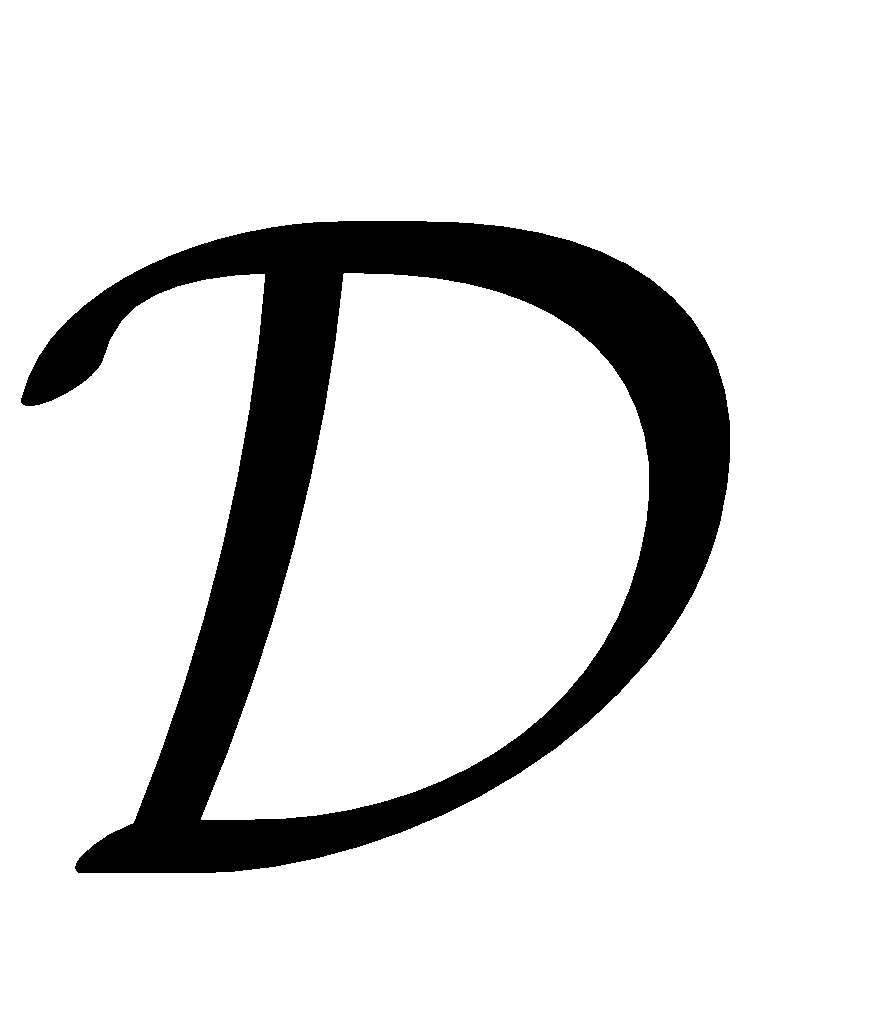
经验回放

评论家

评论家

演员

演员

图 5-5 DMA 算法结构示意图

所提出的 DMA 算法的网络结构如图[5-5](#_bookmark100)所示。DMA 算法基于深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法，结合了策略梯度

和DQN 的优点。对于每个智能体，有两个主要组成部分：演员和评论家。它采用集中训练和分散执行的方法。换句话说，在训练阶段，每个智能体的评论家网络收集所有智能体的状态和行动信息，但在执行阶段，每个智能体的演员网络仅根据局部信息（即智能体自己的行动和状态）做出决策。在每个决策持续期间， 演员生成行动值 *au*(*k*)，然后评论家通过状态-行动函数 *Qu*(·) 评估演员的行动。

在第 *k* 个子任务的决策持续期的开始，NCP 将接收其自身状态 *su* ∈ *Su* 并

将其输入到演员网络中，以基于当前策略获得最优行动 *au*。在执行行动 *au* 后， NCP *u* 接收其奖励 *ru*，状态 *su* 转移到 *s*′*u*。(*su, au, ru, s*′*u*) 作为一个转换集合存储在 本地经验回放缓冲区 *Du* 中。假设 *Du* 的容量为 κ*r*。当转换集合数量超过 *Du* 时， 使用最新的来替换最早的一个。维护经验回放缓冲区是参数更新过程的重要步

骤。

DDPG 算法的目标是通过调整演员和评论家的评估和目标网络中的参数直至收敛，以找到最优策略π∗。在训练阶段，每个回合（Episode）中的参数θ*u* 和θ *Q* 在评估网络中进行更新。接下来介绍参数更新方法。

*u*

演员网络：在训练过程中，每个智能体首先从 *Du* 中抽取 κ*m* 个转换集合， 形成一个小的批量。演员网络的目标是通过与环境和其他智能体的交互，获得最优策略 π∗ 来最大化目标函数 *J*(θ*u*)。因此，演员网络通过计算目标函数的梯度来更新参数，如下所示：

∇θ*u J*(θ*u*)

= E*s,a*∼*D*

*u*

∇θ π*u*(*s*)∇*a Q*π (**s***, a*1*, ...aN* )|*au* = π*u*(*su*)l

(5-12)

≈ 1 ∑∇θ π (*s*) · ∇ *Q*π (**s***i, ai , ...ai* )|*ai*

*u*

*u*

*u*

*u a*

= π (*si* )

κ*m i u*

*u u* 1 *N u u u*

其中 *i* 是小批量中样本的索引，**s** = {*s*1*, . . . sN* } 是状态集。因此，参数 θ*u* 可以朝

∇θ*u J*(θ*u*) 的方向调整。

评论家网络：评论家网络旨在获得最优的状态-行动函数。它通过最小化目标状态-行动函数的损失函数来调整评估网络中的参数，该损失函数写为：

*L* (θ*u*) = E[*yu* − *Q*π (**s***, a*1*, a*2*, ..., aN* )]2 (5-13)

*u*

*yu* 是目标状态-行动函数：

*yu* = *ru* + γ*Q*π′ (**s**′*, a*′1*, a*2′ *, ..., a*′*N* |*a*′ =π′ (*s* )) (5-14)

*u*

*u*

*u*

*u*

其中 π′ 是目标策略。然后，参数 θ *Q* 可以使用损失函数 *L* (θ*u*) 的梯度来更新。

*u*

与评估网络不同，目标网络采用软更新，表示为：

θ*u*′ = τθ*u* + (1 −τ)θ*u*′ *,* (5-15)

其中 τ ∈ (0*,* 1) 是更新步长。

为方便理解，该算法的伪代码的形式如算法[5-2](#_bookmark101)所述。

算法 **5-2:** 分散式多智能体强化学习算法

输入： 子任务集合 *K* , 探索概率 ε, 折扣因子 γ;

输出： 最优策略 πθ∗*u* ;

1: **for** 每个 episode **do**

2: 接收初始状态 *S* (0); 3: **for** 每个子任务 *T k* **do** 4: **for** 每个代理 *u* **do**

5: 执行 ε-贪婪策略：从均匀分布 *U* (0*,* 1) 生成一个随机数 *p*;

6: **if** *p <* ε **then**

7: 随机选择一个动作 *au*(*k*);

8: **else**

9: 选择动作 *au*(*k*) ← πθ*u* (*su*(*k*));

10: **end if**

11: 获得奖励 *ru* 和新状态 *s*′*u*;

12: 在回放缓冲区中存储 (*su, au, ru, s*′*u*);

13: 经验回放：从 *Du* 中抽取一个包含 *M* 样本的小批量 (*su, au, ru, s*′*u*);

14: 设置 *yu* ← *ru* + γ*Q*π′ (*s*′*, a*′ *, . . . , a*′ )|*a*′ =π′ (*s* );

*u*

1

*N*

*u*

*u*

*u*

15: 通过最小化损失函数 *L* (θ*u*) ← E[(*yu* − *Q*π (**s***, a*1*, . . . , aN* ))2] 更新评估网

*u*

络中的批判者;

16: 使用采样的策略梯度

∇θ *J*(π*u*) ← 1 [∇θ *Q*π (**s**′*, a*1*, . . . , aN* ) · ∇θ π*u*(*au*|*su*)] 更新评估网络中的

*u*

演员;

17: **end for**

*M u u u*

18: 更新演员和批判者目标网络中的参数 θ*u* ← τθ*u* + (1 −τ)θ*u*′ ;

19: **end for**

20: **end for**

#### 实验与性能分析

本节对所提方法的性能进行评估。首先，介绍实验设置。然后，说明对比方案。最后，展示并分析实验结果，以验证所提方法的可行性和优越性。

**5.6.1** 实验设置

本章在一台 DELL PowerEdge R730（3.2GHz, 64GB 内存，GeForceGTX 3090） 上使用 OMNeT++ 仿真软件和 INET 框架[[101](#_bookmark248)]，创建了一个覆盖 100\*100 *m*2 区域的仿真环境。该环境包括一个作为桥接节点的边缘基站和九个具有计算能力的边缘设备，作为NCPs，如图 [5-6](#_bookmark104)所示。红色星星代表桥接节点（即边缘服务器或基站），位于中心，而圆圈代表NCPs。为了直观地传达不同NCPs 的计算能力差异，圆圈的大小和颜色进行了区分，较大且颜色较浅的圆圈表示每秒 CPU 周期数更高，算力更强。

###### 100



桥接节点

80



60

40

20



0 0 20 40 60 80 100

图 5-6 实验环境

3*.*5

3*.*0



2*.*5

GHz

2*.*0

1*.*5

下面将详细介绍在实验中使用的参数信息。计算任务被构建为具有依赖性的可分割任务，这些任务需要各种计算资源，包括处理单元（CPU）、传输能力

（网络带宽）等。任务链结构由 2 个分支组成，类似于图 [5-3](#_bookmark87)。每个NCPs 的计算资源 *fu*（即可用 CPU 周期）服从正态分布，µ = 2*.*5 和 σ = 0*.*5 GHz，范围从 1 GHz 到 4 GHz。传输功率 *pu* 从 [150*,* 250] mW 中随机选择，信道带宽设置为 20

MHz。此外，输出数据大小 *Dk* 和计算资源需求 *Ck* 分别服从 [500*,* 1000] KB 和

表 5-2 章节五实验参数

|  |  |
| --- | --- |
| 符号 | 数值 |
| *fu* | 正态分布，µ = 2*.*5，σ = 0*.*5，范围 [1,4] GHz |
| *pu* | 在 [150,250] mW 中随机选取 |
| *Bu* | 20 MHz |
| *Dk* | 在 [500,1000] KB 中均匀分布 |
| *Ck* | 在 [100,300] MHz 中均匀分布 |
| *Pu* | 50% |
| α | -6 |
| *N*0 | 2 × 10−13 W |

[100*,* 300] MHz 的均匀分布。具有任务失败可能性的NCPs 被定义为不可靠节点， 可能性 *Pu* 设置为 50%。上述及其他详细的参数信息在表 [5-2](#_bookmark105) 中呈现。

* + 1. 对比方案

为了进行比较，考虑了以下四个基准方案：

* + - * 本地：所有子任务都由发起请求的节点处理。
      * 随机：对于每个子任务，桥接节点在该区域随机选择一个NCP 执行。
      * 贪婪：每个子任务都会卸载到计算和通信资源最大的NCP。
      * **CODIA** 文章 [[102](#_bookmark249)]：在这篇文献中，作者提出了一种智能计算卸载方案， 适用于依赖型物联网应用。它首先基于多队列优先级调度算法分析了子任务的优先级，然后采用单个智能体的深度强化学习方法卸载子任务。
    1. 实验结果

为了评估BIDC 的性能，本节进行了一系列数值实验，重点关注收敛性、时间成本和公平性。需要注意的是，除非另有说明，子任务数量为 9，NCP 数量为9，不可靠NCP 数量设置为 2，叔区块的最大数量φ 为 3（从用户总数的 30% 上取整）。

首先验证 DMA 算法的收敛性。计算一个回合中所有NCP 的总奖励作为垂直坐标，如图[5-7](#_bookmark108)所示。在不同NCP 数量（6，9，12，15）下，训练过程均在 80-150 回合内收敛。这证明了设计的算法使每个NCP 都能形成成熟的策略来承担计算任务。此外，参与NCP 越少，DMA 收敛越快。6 个NCP 的模拟大约需要 80 个回合才能稳定，而 15 个NCP 则需要 140 个回合。尽管更多的参与者增加了系统

400

6 NCPs

9 NCPs

12 NCPs

15 NCPs

300

200

100

奖励

0

−100

−200

0 100 200 300 400 500 600

回合（Episode）

图 5-7 不同 NCP 数（6, 9, 12, 15）下的奖励值收敛性

400

6 STs

9 STs

12 STs

300

200

100

奖励

0

−100

−200

0 100 200 300 400 500 600

回合（Episode）

图 5-8 不同子任务数（6, 9, 12）下的奖励值收敛性

的复杂性，DMA 算法在相对较少的回合内仍然保持收敛，凸显了所提方法的可扩展性。

然后，调整子任务数量（6，9，12），实验结果显示在图[5-8](#_bookmark109)中。可以观察到子任务数量对收敛速度影响不大。所有三条曲线在 100 个回合内收敛。

此外，本节还测试了两种不同的任务链结构：单链和随机DAG。在单链结构中，没有分支，子任务顺次连接。在随机DAG 中，子任务之间的连接通过邻接矩阵随机构建。如图[5-9](#_bookmark110)所示，单链需要最少的回合才能收敛，随机DAG 需要最多。然而，三种结构之间的差异相对较小。根据图[5-8](#_bookmark109)和图[5-9](#_bookmark110)中的结果，所提方法有效地适应了子任务数量的变化和各种计算需求。

400

随机 DAG

单链

两个分支

300

200

100

奖励

0

−100

−200

−300 0 100 200 300 400 500 600

回合（Episode）

图 5-9 不同结构的任务链奖励值的收敛情况

接下来，在图[5-10](#_bookmark111)中研究了不同NCP 数量下每个任务的时间消耗性能。可以看出，随机耗时最长。本地排名第二，因为它仅依赖自身的计算资源，并且尽管没有传输成本，也无法并行处理子任务。其他三种方案随着NCP 数量的增加呈现出下降趋势，因为它们都利用了全局计算资源。贪婪和 **CODIA** 的时间消耗相似，因为它们都尝试为每个子任务选择最佳NCP，但忽略了设备节点的不可靠性和故障概率。这样，本章所提方法提供了最低的成本，几乎是随机的一半时间。

此外，本节还评估了在不同不可靠NCP 比例下的性能。图[5-11](#_bookmark112)显示所提方法始终表现出最少的时间消耗，增长率低于其他方案。这可以归因于DAG 区块链设计，其中每个NCP 都被激励贡献其计算资源，竞争性的挖矿机制保护系统免受节点故障的影响。这些实验结果表明，本章所提方法可以有效应对现实环境中设备故障和掉线情况。

在实验的最后部分，使用 Jain 公平性指数[[103](#_bookmark250)] 评估了四种方案的公平性。公

2*.*0



随机贪婪本地

CODIA

所提方案

1*.*6

平均每个任务时间开销 (s)

1*.*2

0*.*8

0*.*4

0*.*0

6 9 12 15

NCP 数量

图 5-10 不同 NCP 数量下的时间消耗对比

2*.*0



随机贪婪

CODIA

所提方案

1*.*6

平均每个任务时间开销 (s)

1*.*2

0*.*8

0*.*4

0*.*0

0 1/9 2/9 3/9 4/9 5/9

不可靠 NCP 比例

图 5-11 不同不可靠节点比例下的时间消耗对比

平性由以下公式给出：

（∑*N ru* ）2

（

*JI* =

*u*=1 *fu*(1−*Pu*)

）

(5-16)

*N*

*N* ∑

*u*=1

*ru* 2

*fu*(1−*Pu*)

1*.*0



随机贪婪本地

CODIA

所提方案

0*.*9

0*.*8

0*.*7

Jain 公平性指数

0*.*6

0*.*5

0*.*4

0*.*3

0*.*2

0*.*1

0*.*0 6

9 12 15

NCP 数量

图 5-12 不同 NCP 数量下的 Jain 公平性评估

其中 *ru* 是一个回合中NCP *u* 的总奖励， *fu* 表示其计算资源，*Pu* 是状态不可用的可能性。可以从图[5-12](#_bookmark113)中观察到贪婪和本地的公平性相对较低。这是因为本地依赖于一个设备，而贪婪利用几个具有高计算能力的固定设备来执行整个任务。随机和本章所提方法是两个公平性指数最高的方案。这可以归因于随机为每个NCP 提供平等机会，而所提方法不仅为计算提供了公平竞争的环境，还奖励了挖掘叔区块的NCP。

#### 本章小结

本章深入探讨了边缘分布式计算中的公平性问题，特别是在复杂和耦合的计算任务中实现公平的计算任务分配。通过提出基于工作量证明的公平化卸载方法，确保了系统中每个节点都能根据其贡献获得相应的回报。该方法通过建立DAG 模型来详细描述任务之间的依赖关系，并采用基于DAG 的适应性区块链结构以及多智能体强化学习算法，有效提升了任务分配的透明性和公平性。实验结果证实了所提出方法的公平性和可靠性。本章的工作展示了人工智能和区块链技术在促进边缘分布式计算发展中的巨大潜力，为未来研究提供了新的视角和工具。

## 第六章 基于链上信息的安全卸载方案

上述三章分别围绕可信性、弹性和公平性问题，提出了相应的解决策略。在本章中，以边缘分布式DNN 训练为例，专注于探讨边缘分布式计算领域中的一项重要但尚未充分解决的问题——卸载安全性问题。随着深度学习技术的迅猛发展，边缘分布式模型训练逐渐成为研究热点。在此过程中，卸载安全性问题显得尤为关键。具体来说，在边缘分布式DNN 训练场景中，计算任务分配至地理位置分散的边缘设备上执行。其中潜在的恶意客户端可能采取模型中毒攻击等手段篡改的模型更新，破坏全局模型的准确性，对整个系统的性能和可靠性构成严重威胁。针对此问题，本章提出了一种基于链上信息的安全计算卸载方案，旨在有效识别并隔离恶意客户端，以确保边缘分布式DNN 计算卸载的安全性。该方案巧妙结合了数据并行与模型并行策略，优化了训练数据的分布和神经网络模型的层级划分，从而提高了训练效率。此外，通过引入基于损失值的信用评估机制和基于洗牌的隔离机制，两种机制协同作用，有效筛选和隔离潜在的恶意客户端。更进一步，利用区块链技术和智能合约，实现了计算任务的透明、公平卸载和激励机制的公正执行，增强了系统的攻击防御能力，确保了模型聚合的准确性及整个训练过程的安全性。本章所提方案不仅有效减少了恶意客户端对模型训练过程的负面影响，也为边缘分布式计算提供了有力支持，为机器学习技术在边缘计算环境中安全应用提供了新的思路。

#### 引言

在信息化和数字化时代的推进下，基于神经网络的机器学习技术已成为支撑智慧城市建设、自动驾驶及医疗健康等关键领域发展的重要基石。特别是DNN 模型，例如 GPT-3[[104](#_bookmark251)] 和 PALM[[105](#_bookmark252)]，为复杂的序列预测、高维数据的特征学习提供了前所未有的精度和效率。然而，随着模型规模和复杂性的增加，对计算资源的需求也相应提升。传统的云计算模式需要将大量数据传输至云端处理，这不仅增加了传输链的负担，还可能引入不可接受的延迟，进一步影响机器学习模型的即时更新与迭代[[106](#_bookmark253),[107](#_bookmark254)]。此外，个人用户往往缺乏足够的计算资源来支持这些大模型的训练，这从一定程度上限制了他们享受科技进步所带来的便利。

近年来，随着边缘设备处理能力的显著提升，边缘智能作为一种新兴领域应运而生，它通过将边缘计算与人工智能技术相结合，不仅优化了数据处理流

程，降低了传输延迟，还减少了对云计算中心的依赖，提高了整个系统的效率和响应速度[[108](#_bookmark255),[109](#_bookmark256)]。这种技术进步为解决计算能力不足的问题提供了新的思路。边缘分布式DNN 计算由此诞生[[110](#_bookmark257)–[113](#_bookmark258)]，通过将训练任务分布至多个边缘设备上，既减轻了单个设备的计算负担，也优化了整个计算处理流程。这一领域主要包括两种方法：数据并行[[114](#_bookmark259)–[116](#_bookmark260)] 和模型并行[[117](#_bookmark261),[118](#_bookmark262)]。数据并行方法涉及将数据集分布到不同的客户端进行并行处理，而模型并行方法则将DNN 模型分割，各客户端负责模型的不同部分，以实现顺序和接力式的处理。尽管边缘分布式计算在处理能力和效率上取得了显著进步，但随着边缘数据量的激增citeAAAI-2021-1,tii-2022-1 和互联设备的多样化，新的安全挑战也随之出现。特别是在计算卸载过程中，如果任务被卸载到恶意客户端，可能会遭受模型中毒攻击，恶意客户端通过提交篡改的模型更新来破坏全局模型的准确性[[36](#_bookmark184),[119](#_bookmark263)]。为了应对这些挑战，虽然提出了如拜占庭鲁棒性等防御机制[[120](#_bookmark264)–[123](#_bookmark266)]，但这些方法主要通过最小化恶意客户端的影响，而不是直接检测并排除恶意行为。其他防御策略如文献 [[38](#_bookmark186)] 提出的解决方案专门针对特定的攻击模式，缺乏缺乏针对模型并行场景的通用性和有效性。在模型并行的环境中，这些防御机制的不足之处变得更加明显，因为一个恶意客户端可能对整组的计算结构产生影响，从而危害整个模型的性能。

在此背景下，开发有效的机制以识别恶意行为，并确保计算模型及其结果的准确无误，对于边缘分布式计算卸载而言，是一个亟需解决的关键挑战。同时，鉴于边缘计算设备的多样性及不同边缘基础设施提供商的服务，实施适当的激励机制变得至关重要。缺乏有效的激励措施，边缘计算节点（工作节点）可能不会积极参与到协作计算任务中来[[41](#_bookmark189)]。此外，建立边缘计算节点间的信任， 确保计算任务能够安全、透明、可验证地卸载到可信赖的设备上，对于提升边缘分布式计算的性能和安全性，保障系统的可靠性和用户的最终体验至关重要。

因此，本章提出了基于链上信息的分布式计算卸载方案，旨在对边缘分布式DNN 计算中的恶意客户端进行检测识别，从而实现安全的计算卸载。为了提升训练效率，该方案融合了数据并行与模型并行策略，对训练数据进行分割，同时将神经网络模型进行层级划分识别恶意客户端的关键在于采用基于损失值的信誉评估机制，通过分析客户端在各自小组中的表现，于每个训练周期（Epoch） 后更新其信誉评分。这一评估标准源自于预先实验的观察，即完全由诚实客户端构成的小组与包含至少一个恶意客户端的小组，在损失值方面表现出显著差异。基于此，进一步设计了一种基于洗牌的隔离机制重新分组客户端。这种策略根据客户端的更新后信誉评分，调整其在后续训练周期中的分组概率，使得

信誉相近的客户端更有可能被分配至同一组。为了应对激励和信任方面的挑战， 本方案整合了区块链技术。通过智能合约，实现了透明且公正的性能评估与任务分配，确保了根据客户端的贡献公平地分配奖励并更新其声誉。区块链的不可篡改特性为整个过程的完整性和透明性提供了保障，进而促进了参与者之间的信任和积极参与。此外，加入了拜占庭容错机制以增强系统抵御攻击的能力， 确保了模型聚合的准确性。

综上所述，本章所提方案在确保训练效率的同时，有效最小化了恶意客户端对模型训练过程的影响。本章的主要贡献可总结如下：

* + - 设计了一个新颖的基于链上信息的分布式计算卸载方案，通过结合数据并行与模型并行的方法，以及有效识别并隔离恶意客户端的机制，保障了分布式DNN 计算卸载的安全性。
    - 提出基于损失的信用评估和基于洗牌的隔离机制，有效地筛选和隔离潜在的恶意客户端。
    - 引入了区块链框架以解决计算客户端间的激励与信任问题，利用智能合约实现了一个透明且公平的性能评估和任务卸载，为分布式DNN 训练提供了一个稳健的支撑平台。

本章将按以下顺序进行：第 6.2 节阐述系统架构和对抗模型。第 6.3 节描述恶意节点识策略。第 6.4 节分析了实验结果。最后，在 6.5 节对本章进行总结。

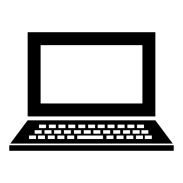
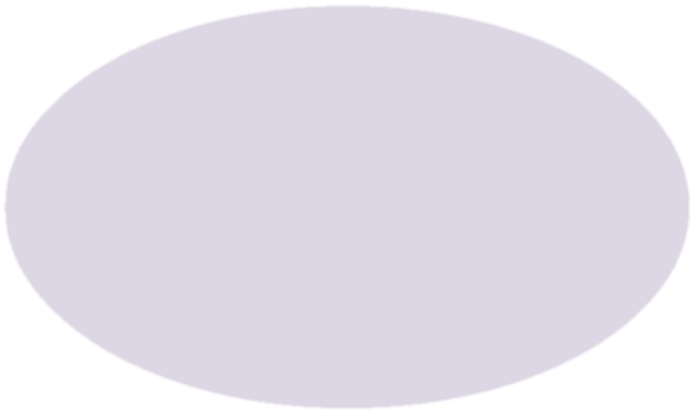
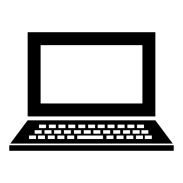
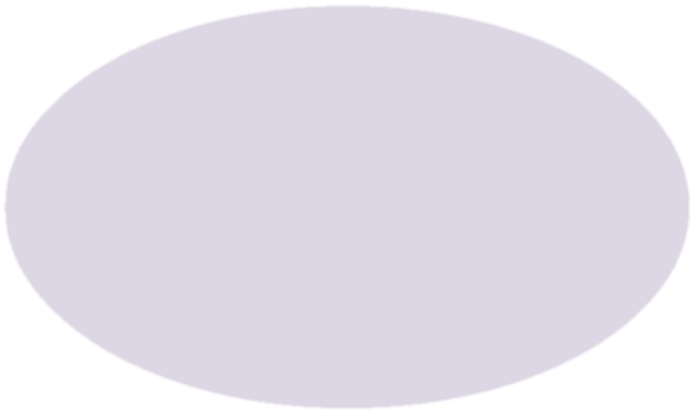
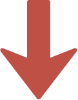
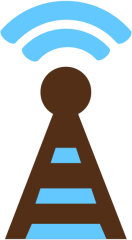
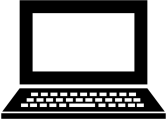
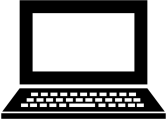
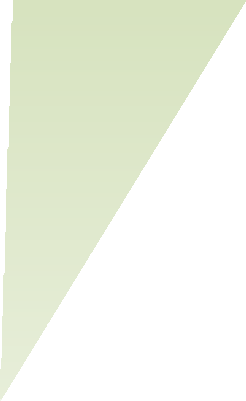
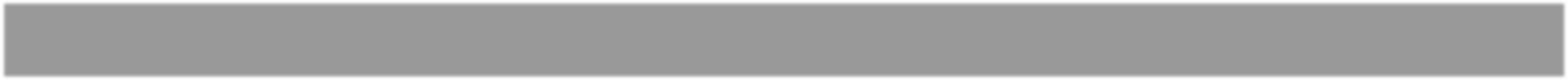
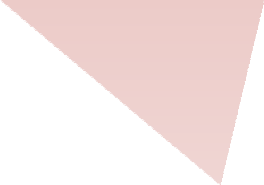
#### 系统模型和攻击模型

* + 1. 系统模型

图 [6-1](#_bookmark119) 描述了在分布式学习场景中所提方案的系统模型。当出现计算任务时，边缘服务器或区域基站（也称为主节点）在区块链上发布其任务内容和奖励信息，并根据已在区块链上注册的边缘设备的计算能力和声誉选择 *N* 个工作节点。节点注册、任务发布和其他相关过程的交互可以参考作者之前发表的文章[[124](#_bookmark267)]。然后，主节点将 *N* 个边缘设备分成几个小组，共同训练全局模型 *M* 。*M* 由 *L* 个顺序层组成，每个层 *Li* 包含一组参数 *Wi* 和一个前向传播函数 *Fi*。

为了加速训练，在组之间采用数据并行，而在每个组内采用模型并行。主节点将整个数据集 *D* 划分为 *G* × *Q* 部分，以支持 *G* 个小组的数据并行训练。在每个小组内，进行 *Q* 次迭代训练，作为一个Epoch。同时，主节点将 *M* 划分为 *K* 个

**与区块链交互工作流程**



**与智能合约交互**

*单元 1 单元 2 单元 K-1 单元 K*

***...***

*组 1*

*模型 1*

*损失 1*

*Q次迭代之后*

*/*

LACE 智能合约

**②**

*组 G*

*模型 G*

*/ 损失 G*

SBIM 智能合约

**①**

**⑥**

**设备池（客户端）**

*...*

***...***

**③**

BTA 智能合约

**④**

**⑤**

**主节点**

**区块链**

上传参数

拉取参数

**拜占庭容错聚合**

*相关性*

*可验证性*

*随机性*

***攻击者聚类算法***

**基于洗牌的隔离机制**

*优先级排序*

*优先级计算*

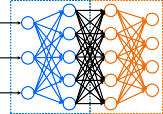
*加权模型聚合*

*模型选择*

***...***

***...***

***...***

***...***



|  |  |
| --- | --- |
| **基于损失的信用评估** | |
| *基于损* | *小组评分* |
| *失的信* |  |
| *用评估* |  |
| *算法* | *信用更新* |

**③**



图 6-1 系统模型

上传

参数

单元，每个单元 *k* 包括从 *i* 到 *j* 的连续神经网络层，表示为 L*k* = {*Li, Li*+1*, ..., Lj*}。然后，每个单元将被分配给一个客户端进行训练。模型划分方法可以参考文献[[125](#_bookmark268)]，并满足 *N* = *G* × *K*，这是确保了参与的客户端数量等于总数。在单元 *k* 中， 参数 W*k* = {*Wi,Wi*+1*, ...,Wj*} 和前向传播函数 F*k* 通过 *Fi* ◦ *Fi*+1 ◦ *. . .* ◦ *Fj* 计算得出， 输出值表示为 *ak* = F*k*(*ak*−1)。梯度为 *gradk* = ∇*loss*(*D,* W*k*)。此外，令 *G t* 表示时隙 *t* 中的所有组的集合，其中每个时隙为一个迭代回合（Episode）。工作流程如下：

* + - * 主节点将初始模型参数上传到区块链，然后将所有可用客户端分组，为每个客户端分配一个单元。客户端根据分配的单元从区块链拉取相应的模型参数。
      * 开始训练。每 *K* 个客户端形成一个小组 *g*，协作训练模型 *M* ，表示为 *Mg*。小组 *g* 的损失为 *Lg* = *l*1*, l*2*, ..., lQg*，其中 *lq* 表示第 *q* 次迭代中的损失值。
      * 完成一个 Epoch 后，*Lg* 和 *Mg* 被打包成一个交易，并上传到区块链。
      * 主节点从区块链检索损失记录 *L*1*, L*2*,..., LG* 并调用基于损失的信用评估

（Loss-aware Credit Evaluation，LACE）智能合约来评估每个小组的表现， 并更新客户端的信用。

* + - * 随后，通过拜占庭容错聚合（Byzantine-tolerance Aggregation，BTA）智能合约将一组 *Mg* 聚合，以产生新的全局模型。
      * 同时，主节点调用基于洗牌的隔离机制（Shuffling-based Isolation Mechanism， SBIM）智能合约，为下一个 Epoch 的训练重新分组客户端。

这六个步骤持续迭代，直到训练结束。

* + 1. 攻击模型

恶意客户端可以通过参与训练过程对 *M* 发起模型中毒攻击。考虑到每个客户端负责整个模型中的固定单元，因此，恶意客户端将掌握以下知识：学习率、所负责单元的初始参数设置、前一个单元的神经网络输出值 *ak*−1 以及反向传播过程中后续单元的梯度 *gradk*+1。然而，与传统设置不同的是恶意客户端只能控 制特定神经网络层的输出。这限制了攻击只能在 *ak* 和 *gradk* 两个地方发生。

接下来，本节基于引言中提到的两种类型的攻击进行模型中毒攻击的分析和建模。对于非针对性中毒攻击，恶意客户端的目标是减缓模型的收敛速度并降低其准确性，这可以通过篡改输出值或梯度来实现。介绍三种典型的攻击手段：1）引入扰动[[126](#_bookmark269)]，在输出的参数中添加噪声信息，降低数据精准度；2）反转梯度更新[[36](#_bookmark184)]，反向传播过程中进行梯度反向更新；3）随机化输出值。

关于针对性中毒攻击，如前所述，绝大多数此类攻击涉及数据操纵，而更新操纵方法需要对整个DNN 模型有全面的了解。由于模型并行中的恶意客户端只能访问特定单元的输入数据和反向传播梯度信息，执行针对性中毒攻击几乎变得不可行。因此，本章中涉及的攻击模型仅专注于非针对性中毒攻击。

* + 1. 基于区块链的声誉机制

在边缘分布式计算环境中，实施有效的激励和信任机制对于维护协作计算的完整性和效率至关重要。考虑到此类网络中节点的可靠性，开发一个能够可靠评估、激励并在参与者之间建立信任的系统至关重要。

为解决此问题，在各节点上建立了区块链系统，其在本架构中扮演着核心角色。通过将所有关键数据，包括计算记录及信用历史等，记录于一个不可篡改的账本之上，该系统在确保整个流程的透明度与可验证性方面发挥关键作用。

此外，一些关键的流程如信用更新、奖励分配及工作节点选择等，与智能合约集成。这确保了所有重要操作均依据既定规则，以一致且准确的方式执行，从而提升了系统的效率与可靠性。接下来，将详细介绍具体实施步骤。

* + - 1. 声誉构建和更新

客户端的声誉基于他们在每个计算过程中获得的信用评分。这个分数使用指数移动平均（Exponential Moving Average, EMA）来确定：设 *Rt* 为时间 *t* 时候的声誉分数，*ct* 为时间 *t* 获得的信用，α（其中 0 *<* α *<* 1）为平滑因子。基于EMA 的声誉分数计算如下：

*Rt* = α ∗ *ct* + (1 −α) ∗ *Rt*−1 (6-1)

这种方法逐渐减少了旧信用的重要性，允许对客户端的可靠性进行动态评估，并更强调近期行为的影响。

该步骤在系统中的实施依赖于 LACE 智能合约。

* + - 1. 奖励分配

在每项计算任务完成之后，奖励将依据客户端的声誉得分，按照一定比例分配至各客户端。此种分配方式旨在保证基于持续表现及可靠性的公正奖励机制，从而激励节点进行高质量的计算。

奖励分配也是通过设计的智能合约实现的，可以参考第三章内容。

* + - 1. 工作节点选择

基于客户端声誉的加权概率分布指导工作节点的选择。首先使用声誉分数上的 *softmax* 函数来计算客户端 *n* 被选中的概率：

*Prn* =

*ct*−1

*t*

*n*

(6-2)

∑ *j*∈*G t c j*−1

*p*

接下来，采用轮盘赌选择算法[[127](#_bookmark270)]，将客户端分配至 [0,1] 区间内，其段落

大小与各自的概率重要性成正比。通过生成一个位于 [0,1] 区间内的均匀分布随机数来决定此轮所选客户端，具体依据该随机数所落之段决定。随后，依据公式（[6-2](#_bookmark122)）对尚未被分配的客户端的概率进行重新评估。被选中的客户端将依序

算法 **6-1:** 工作节点选择算法

输入： 客户端集合 C，工作节点容量 *N*，客户端在 C 中的声誉;

输出： 选中的工作节点 U;

1: 初始化工作节点集合 U = [].

2: **while** |U| *< N* 且 C 非空 **do**

3: 使用公式（[6-2](#_bookmark122)）计算 C 中每个客户端的概率；

4: 生成 [0,1] 区间内的均匀随机数 *x*；

5: 根据轮盘赌选择使用 *x* 选择客户端 *i*；

6: 将客户端 *i* 添加到 U；

7: 从 C 中移除客户端 *i*。

8: **end while**

9: **return** U 。

被编入小组，直至小组容量 *K* 达到预设值，随即开始向下一小组分配后续客户端。该流程详细在算法 [6-1](#_bookmark123) 中概述。

客户端的选择受到其声誉分数的影响，从而提高了分布式计算环境中的公平性和信任度。

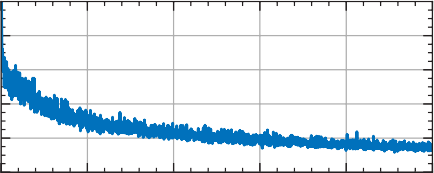
工作节点选择的计算也是由主节点调用 SBIM 智能合约来实现的。

#### 恶意客户端检测策略

* + 1. 基于损失的信用评估
       1. 预实验

在模型并行训练领域，关于恶意客户端使用的攻击策略及其对模型损失的后续影响的研究很少。因此，作者事先进行了一系列预实验，专注于前一节描述的三种典型攻击策略。实验结果如图 [6-2](#_bookmark126) 所示。

测试结果表明，即便面临中毒攻击，模型在训练过程中的损失值均方误差损 失（Mean Square Error，MSE）依然能够实现收敛。推测这一现象的原因在于DNN 的强大解释性。此外，两个重要现象被观察到：1. 相较于未受污染的模型，受污染模型在收敛之后展现出更高的平均损失值。2. 受污染模型的损失方差更大， 这暗示了学习过程中存在更大的不稳定性。这些现象不仅阐明了在恶意干扰下损失值收敛行为的特点，也为针对此类攻击开发更为强大的防御机制提供了指导。基于上述观察，本章进一步提出了一种基于损失的信用评估机制，该机制

0*.*8

0*.*6

MSE

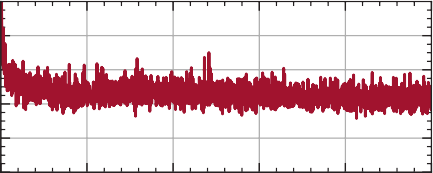
0*.*4

0*.*2

0*.*0 0 500 1000 1500 2000 2500

迭代

0*.*8

0*.*6

MSE

0*.*4

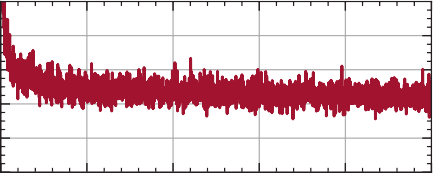
0*.*2

0*.*0 0 500 1000

迭代

1500 2000 2500

(a) 无攻击 (b) 高斯扰动

0*.*8

0*.*6

MSE

0*.*4

0*.*2

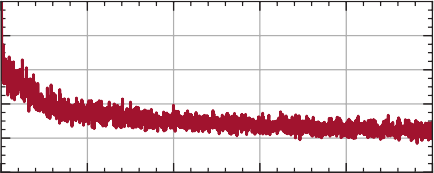
0*.*0 0

500 1000 1500 2000 2500

迭代

1. 梯度反转

0*.*8

0*.*6

MSE

0*.*4

0*.*2

0*.*0 0 500 1000 1500 2000 2500

迭代

1. 输出随机化

图 6-2 不同攻击下分布式 DNN 学习的损失表现（MSE）

包括两个阶段：小组评分和信用更新。

* + - 1. 小组评分

经过一个 Epoch 的训练后，小组 *g* 生成模型 *Mg* 的损失数据 *Lg*。然后基于不同小组的损失值对每个小组进行评分。

本节提出了一种基于核密度估计（Kernel Density Estimation, KDE）[[128](#_bookmark271)] 的的小组评分机制。KDE 方法通过使用特定的核函数及带宽参数，借助选定的样本集合来近似变量的概率分布。对于小组 *g*，损失值 *l* ∈ *Lg* 的核密度估计可以表示为

1 *Q*

 *fg* (*l*) = ∑ *Kh* (*l li*) *,* (6-3)

� −*Q*

*i*=1

其中 *Kh*(*l*) = 1 *K*(*l, h*)。*K*(·) 是核函数。*h >* 0 是带宽。在本章中，使用的核函数是高斯核，即 *K*(*l, h*) = exp(− *l*2 )。

*h*



2 2*h*

然后得到 *f*�*g* (*l*) *,* ∀*l* ∈ *Lg*。直观上，模型 *Mg* 的损失收敛到值 *lg*∗，这对应于

*fg* (*l*) 的峰值；曲线顶部的宽度表示收敛损失的波动幅度。基于半高全宽机制

�

� � （ ）

（Full Width at Half Maximum）[[129](#_bookmark272)]，用 *wg* 表示顶部宽度，即满足 *fg* (*li*) *>* 1 *fg lg*∗ 的损失值 *li* ∈ *Lg* 的数量。结合预实验中的观察，可以得出结论：拥有较大 *lg*∗ 和较大 *wg* 的小组有较高可能性包含恶意客户端。因此依据下述小组评分机制，对

2

小组 *g* 进行评分。

*Sg* = 1 − *N*(*lg*∗ · *wg*)*,* (6-4)

其中 *N*(·) 是归一化函数。

在小组评分后，使用诸如 K 均值（K-means）[[130](#_bookmark273)] 等聚类方法将所有小组划

分为两类，表现良好的小组（Well Groups，WGs）*G t*

*w*

和表现较差的小组（Poor

Groups，PGs）*G t* 。有 *G t* ∪ *G t* = *G t* ，且 *G t* ∩ *G t* = ∅。

*p*

*w*

*p*

*w*

*p*

* + - 1. 信用更新

令 *cn* 表示客户端 *n* 的信用，*gn* 表示包含客户端 *n* 的小组。然后，对于每个客户端 *n*，按以下方式更新其信用：

*ct* = *ct*−1 + *It* · *S* · Φ*,* (6-5)

*n*

*n*

*gn*

其中 Φ 是一个固定参数，用于衡量更新的幅度。*It* 是一个衡量奖惩的参数，设计如下：

*p*

*p*

*t* −0*.*5*,* 如果 *n* ∈ *G t* 且 *n* ∈ *G t*−1

*p*

*I* = *p*

*t w*

0*.*5*,* 如果 *n* ∈ *G*

−1*,* 如果 *n* ∈ *G t* 且 *n* ∈ *G t*−1

1*,* 如果 *n* ∈ *G t*

*w*

*w*

且 *n* ∈ *G t*−1

且 *n* ∈ *G t*−1 *.*

*w*

(6-6)

采取此种机制，始终处于 WGs 的客户端信用将迅速提升。相反，始终处于 PGs

的客户端信用通常会更快降低。此设计有效提升了识别潜在恶意客户端的效率。整个基于损失的信用评估流程详见算法 [6-2](#_bookmark131) 所示。

* + 1. 基于洗牌的隔离机制

基于洗牌的隔离机制的目标是在最少的 Epoch 内隔离恶意客户端。直观上， PGs 的数量，即 | *G t* |，可以用来衡量恶意客户端的聚集程度。因为将所有恶意客户端限制在最少数量的小组中会导致 | *G t* | 的最低值。因此，主要目标是最小

*p*

*p*

化 E 1 ∑*T* | *G t* |－，其中 *T* 表示训练 Epoch 的总数。

*T*

*t*=0

*p*

为了最小化小组间恶意客户端的分布，维持WGs 不变，仅对处于 PGs 中的客户端进行重新分组。在此基础上，设计恶意客户端聚类算法。该算法基于与各自的信用分数 *ci* 成正比的概率随机选择客户端分组。具体而言，以优先级值

算法 **6-2:** 基于损失的信用评估算法

输入： 小组集合 *G t* ；所有小组的损失向量 *Lg,* ∀*g* ∈ *G t* ；上一时隙的信用：

*ct*−1*,* ∀*n* ∈ *N* ;

*n*

输出： 信用 *ct* ，PGs 集合 *G t* 和 WGs 集合 *G t* ;

*n p w*

1: # 小组评分 #

2: **for** 所有 *g* 在 *G t* 中 **do**

3: 使用公式（[6-3](#_bookmark127)）计算核密度估计；

4: 使用公式（[6-4](#_bookmark128)）的方式给小组 *g* 打分；

5: **end for**

6: 将所有小组分类为 PGs 和 WGs。

7: # 信用更新 #

8: **for** 所有客户端 *n* **do**

9: 使用公式（[6-5](#_bookmark129)）更新信用 *ct* 。

*n*

10: **end for**

为基准进行分组，将信用分数相近的客户端将更有可能归入同一组。该值的计算与客户端的信用密切相关。此外，为了符合区块链的分布式计算特性并优化算法效率，每个客户端独立计算其优先级值。主节点仅负责收集和排序这些值以进行分组。因此，整个过程必须满足三个关键要求：1. 随机性：算法必须包含高度的随机性，以确保不可预测性。2. 可验证性：客户端计算过程必须是可验证的，以排除任何对优先级值的伪造。3. 相关性：优先级值应与客户端的信用正相关。

为了实现这些要求，本研究使用基于可验证随机函数（Verifiable Random Function，VRF）[[131](#_bookmark274)] 的加密抽签算法。在 VRF 的背景下，每个客户端 *i* 必须拥有一个公/私钥对，表示为 (*pki, ski*)。对于给定任何输入字符串 *seed*，函数

*VRFsk*(*seed*) 生成两个输出：哈希值 *hash* 和证明值 π。

* 对于 随机性，使用公开披露的随机 *seed*。这个 *seed* 是由最新区块的哈希和当前系统时间戳派生的，由主节点广播出去。来自 VRF 的输出 *hash* 是一个伪随机数，均匀分布在 0 和 2*hashlen* − 1 之间，其中 *hashlen* 表示哈希的位长度。
* 对于 可验证性，*hash* 由 *sk* 和 *seed* 唯一确定。证明 π 使得任何掌握 *pk* 的节点都能验证与 *seed* 相关的哈希的合法性，而无需访问 *sk* [[132](#_bookmark275)]。
* 对于 相关性，通过将 *hash* 除以 2*hashlen* 来归一化，得到区间 [0*,* 1) 内的值， 表示为 *hash/*2*hashlen*。随后使用此值来获取客户端 *i* 的优先级值，遵循二项分布 *B*(*k*; *ci, p*)。这里的概率 *p* = τ*/W* ，其中 τ 代表参与洗牌过程的客户端

总数，*W* = ∑ *ci*。然后定义选择区间为 *I j* = ∑ *j B*(*k*; *c, p*)*,* ∑ *j*+1 *B*(*k*; *c, p*)），

*k*=0

*k*=0

其中 *j* ∈ {0*,* 1*,..., ci*}。客户端的哈希值如果落入区间 *I j*，那么 *j* 则为其优先

级值。直观上，较高的 *ci* 会导致区间 0 和 1 之间的划分更密集，从而哈希值落入更高区间编号的可能性更高，即优先级值更大。

在客户端完成这些计算后，主节点接收所有 *ji* 值，并按降序排序，依次选择前 *K* 个值形成一个小组，最终完成重新分组过程。

基于加密抽签的洗牌机制，不仅在重新分组客户端时促进公平和随机性，而且有效地将具有相似信用的客户端聚类在一起。这些在算法 [6-3](#_bookmark133) 中进行了总结。

* + 1. 理论分析

接下来对本节所提算法的计算复杂性和效率及进行理论分析。

* + - 1. 计算复杂性

在算法 [6-2](#_bookmark131) 中，第 3 行执行的 KDE 过程的计算复杂性为 *O*(*Q*2)。此外，为每个小组评分的任务时间复杂性恒定，表示为 *O*(1)。考虑到该算法遍历 *K* 个小组， 且给定 *N* = *K* × *Q*。那么小组评分阶段的累积时间复杂性等同于 *O*(*K* ×(*Q*2 + 1))。这个表达式可以进一步简化为 *O*(*K* × *Q*2)，即 *O*(*N* × *Q*)。此外，算法第 6 行描述的聚类方法，产生的时间复杂性为 *O*(*K*)。随后的阶段，即信用更新，需要遍历 *N* 个元素。因此，算法 [6-2](#_bookmark131) 的总体时间复杂性被确定为 *O*(*N*2)。

对于算法 [6-3](#_bookmark133)，需要分成两个阶段展开讨论。初始阶段涉及参与洗牌过程的各个客户端的并行执行。在此阶段，每个客户端的计算复杂性为 *O*(*c*)，其中 *c* 表示客户端的信用评分。随后的阶段，即主节点执行阶段，使用冒泡排序算法。计算复杂性为 *O*(*N*2)。完成排序任务后，算法继续对排序的元素进行分组，该操

作产生的复杂性为 *O*(*N/K*)。因此，算法 [6-3](#_bookmark133) 的总体计算复杂性被确定为 *O*(*N*2)。

综上，整体算法的时间复杂性为 *O*(*N*2)。

* + - 1. 收敛性分析

在设计的模型中，客户端分布 *G t* 作为状态，洗牌过程作为状态之间的转换。标在于实现系统向一个稳定状态的收敛，在该状态下，尽可能地将所有恶意客

算法 **6-3:** 恶意客户端聚类算法

输入： *G t* ，*G t* ，小组容量 *K*，最新的区块链哈希 *hash*

，*timestamp*;

*p w block*

输出： 客户端的新小组分布 *G t*+1;

1: 初始化

*seed* ← *HASH*(*hashblock,timestamp*) 和 *p* ← τ*/W* 。

2: # 优先级计算（客户端独立执行）#

3: *< hash,* π *>*← *VRFsk*(*seed*)

4: *j* ← 0

5: **while**  *hash* ∈*/*  ∑ *j B*(*k*; *c, p*)*,* ∑ *j*+1 *B*(*k*; *c, p*)） **do**

6: *j*++

2*hashlen*

*k*=0

*k*=0

7: **end while**

8: **return** *< hash,* π*, j >*

9: 通过 SBIM 智能合约将其上传到区块链

10: # 优先级排序（主节点执行）#.

11: 通过 SBIM 智能合约从所有客户端收集 J ← { *j*1*, ..., ji, ...*}

12: 将集合 J 降序排序

13: 初始化一个空列表 *groups*

14: **for** 在 J 中前 *K* 个连续的元素 **do**

15: 用这 *K* 个元素形成一个小组，并将此小组添加到 *groups*

16: **end for**

17: **return** *G t*+1 ← *groups* + *G t*

*w*

户端集中于一处。因此，提出的洗牌机制可以被描述为一种具有特定属性的马尔可夫过程：有限状态空间：客户端分布的状态是有限的，符合有限状态空间的标准。不可约性：尽管有些情况概率较小，但所有状态可以过渡到任何其他状态。这使得马尔可夫链不可约。非周期性：没有一个固定的周期 *k* 来决定状态的转变，这使得本节的马尔可夫链具有非周期性特征。根据强韧性理论（Strong Ergodic Theorem）[[133](#_bookmark276)] 中的定理 16*.*0*.*2，一个有限状态空间、不可约且非周期性的马尔可夫链，在经过足够多的转换后，无论其初始状态如何，都会稳定到一个稳定状态。对于给定的 *r >* 1 和 *R <* ∞，收敛率遵循：

∥*Pn* (*x,* ·) −π (·)∥ ≤ *Rr*−*n,* (6-7)

其中 π 表示极限概率测度，表示以一致的几何速率收敛。此外，将洗牌工作集中在 WGs 上可进一步加速这一收敛过程。因此，整体算法可收敛。

* + 1. 拜占庭容错聚合

为了在恶意客户端的检测阶段仍保持全局模型的高准确度，设计了拜占庭容错聚合机制。这个过程可以与洗牌过程并行执行。在收到所有小组更新后的参数后，主节点丢弃 PGs 输出的模型，并聚合WGs 的模型。聚合通过基于它们相应评分的加权平均来执行，从而产生一个新的的全局模型。加权聚合的数学公式如下：

*G*

（ ）*M* = *wt S* ∗ *M ,* (6-8)∑ *g*

*t*+1 *t*

*g*

*g*=1

其中 *wt*(*Sg*) 表示小组 *g* 的权重，即 *wt* （*Sg*） = *Sg* 。

*w*

∑ *j*∈ *t S jG*

#### 实验与性能分析

为方便后续描述， 本节将提出的基于链上信息的安全卸载方案命名为Patronus[①](#_bookmark137)。利用 Python 3.10 在 PYTORCH DISTRIBUTED[[134](#_bookmark277)] 上进行实现。在本节中，首先介绍了实验设置，然后评估收敛性和可靠性。

* + 1. 实验设置
       1. 数据集

本节在 ETTh1、ETTm1 [[135](#_bookmark278)] 和 ECL[[136](#_bookmark279)] 数据集上训练 Informer[[137](#_bookmark280)] 和双LSTM[[138](#_bookmark281)]。Informer 是一种采用编码器-解码器架构的预测模型，拥有 11,330,055 个参数；而双层长短期记忆网络（Long Short-Term Memory，LSTM）包含 4,233,735个参数。ETTh1 和 ETTm1 代表来自中国两个城市的两年电力数据，其中 ETTh1 适用于每小时读数，ETTm1 用于每 15 分钟间隔的读数。ECL 则展示了来自美国 321 个客户在两年内每小时的电力消耗数据。

① 灵感来自《哈利 波特》系列中的“守护神咒”。与“守护神咒”对抗摄魂怪的保护功能类似，Patronus 可以先发制人地识别和反击攻击者，在分布式 DNN 训练领域起到了安全保障的作用

* + - 1. 测试环境

表 6-1 章节六实验参数

|  |  |
| --- | --- |
| 符号 | 数值 |
| 总客户端数量 *N* | 30 |
| 恶意客户端数量 | 6 |
| 分组数量 *G* | 6 |
| 初始信用 *R* | 50 |
| 平滑因子 α | 0.5 |
| 默认训练模型 | *In f ormer* |
| 默认训练数据 | *ETTm*1 |

实验部署在局域网（Local Area Network，LAN）环境中。客户端由 15 个树莓 派（4v Cortex-A72 和 4G DDR4 RAM）和 15 个虚拟机（4v CPU 和 8 GB RAM）组成，这些虚拟机托管在不同的计算机系统上。DELL PowerEdge R730（3.2GHzx8， 64G RAM）作为主节点。客户端和主节点之间的通信通过有线光纤连接和无线WiFi 实现。

* + - 1. 参数设置

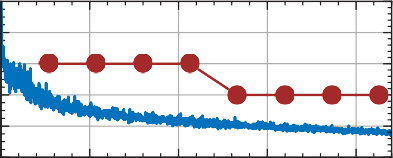
除非另有说明，客户端数量设置为 30，其中 6 个恶意客户端（四舍五入至20%）。小组数量为 6（*G* = 6），每个小组包含 5 个客户端（*K* = 5）。这些恶意客户端通过引入遵循高斯分布的随机扰动来执行攻击。DNN 模型是 Informer，训练数据是 ETTm1（为了节省训练时间，减少能耗，我们只用了最新一年的数据， 即整个数据集的 1/2）。每个客户端的初始信用设置为 50。上述以及其他具体参数设置如表[6-1](#_bookmark138)所示。

* + - 1. 基准
         * **V-DT**（普通分布式 DNN 训练）：一种基本的分布式DNN 训练架构，包括数据并行和模型并行。具体描述可以在文献[[139](#_bookmark282)] 中找到。V-DT 随机选择客户端进行计算任务，缺乏针对潜在攻击防御的机制。
         * 多 **Krum**[[122](#_bookmark265)]：经典拜占庭鲁棒算法 Krum 的增强版本。其核心原理是良性客户端之间的梯度相对相似，与恶意客户端的梯度形成鲜明对比。通过迭代计算，过滤出潜在的恶意梯度，增强了分布式学习的鲁棒性。
         * **FLDetector**[[140](#_bookmark283)]：通过检查模型更新的一致性来检测攻击。具体来说，主

节点根据历史更新来预测后续模型更新。如果更新与预测偏差较大，则相应的客户端被视为恶意。

值得注意的是，鉴于模型并行背景下对中毒攻击的研究有限，上述基准最初是为数据并行场景设计的。因此，在进行比较时，本节采用了它们的概念并应用于设计的实验场景中。

* + 1. 结果和分析

5

0*.*8 4

0*.*6 3

MSE

0*.*4 2

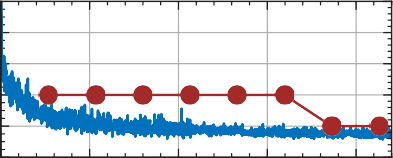
0*.*2 1

0*.*0 0 500 1000 1500 2000 0

迭代

1. 在 Informer 模型上训练 ETTh1

5

0*.*8 4

中毒组数

MSE

中毒组数

0*.*6 3

0*.*4 2

0*.*2 1

0*.*0 0 500 1000 1500 2000 0

迭代

1. 在 Informer 模型上训练 ETTm1

0*.*8

0*.*6

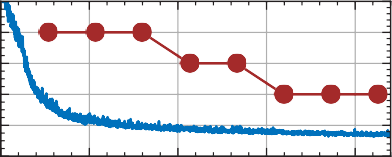
MSE

0*.*4

0*.*2

0*.*0 0

5

4

3

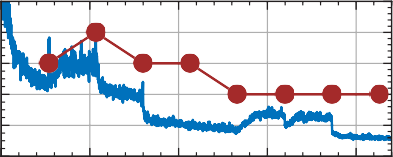
2

1

500 1000 1500 2000 0

迭代

5

0*.*8 4

中毒组数

MSE

中毒组数

0*.*6 3

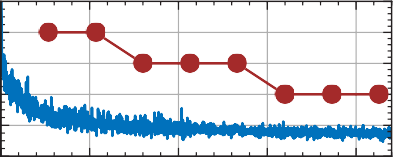
0*.*4 2

0*.*2 1

0*.*0 0 500 1000 1500 2000 0

迭代

1. 在 Informer 模型上训练 ECL
2. 在 LSTM 模型上训练 ETTh1

5

0*.*8 4

0*.*6 3

MSE

0*.*4 2

0*.*2 1

0*.*0 0 500 1000 1500 2000 0

迭代

1. 在 LSTM 模型上训练 ETTm1

0*.*8

0*.*6

中毒组数

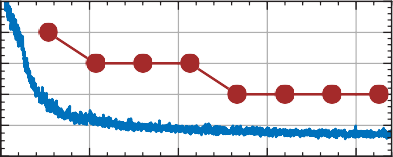
MSE

0*.*4

0*.*2

0*.*0 0

5

4

中毒组数

3

2

1

500 1000 1500 2000 0

迭代

(f) 在 LSTM 模型上训练 ECL

图 6-3 在不同模型和数据集下的损失性能

首先，通过在三个不同的数据集上训练 Informer 和双 LSTM 模型来验证恶意客户端检测策略的性能。所得到的损失值如图[6-3](#_bookmark140)所示。对于 Informer 模型， **Patronus** 在所有数据集上在 1000 次迭代内收敛。在双 LSTM 中，它在图[6-3](#_bookmark140)（e）

和（f）中经过 500 次迭代后达到收敛。而在图[6-3](#_bookmark140)（d）中，MSE 表现出显著波动。结合“中毒组数”指标，推测是由于k-means 聚类错误地将一个 PG 识别为WG。因此，当这个组参与模型聚合时，它向全局模型引入了显著偏差。此外， 通过标记恶意客户端并观察其分布（即每个 Epoch 后的具有恶意客户端的组数， 即中毒组数）来追踪恶意客户端的聚合表现。数据表明，无论初始分布如何，它们通常在多个 Epoch 后最终聚集在 1-2 个组中。

为了验证区块链加入后对系统性能的影响，实验中测量了有链情况和无链情况（此时需要一个集中式节点来执行模型整合、恶意客户端筛选等操作）情况下完成模型训练无轮所需的时间。经过多次测试，平均时间分别为 4483.742s 和4499.264s，两者之间的差距并不显著。此外，观测到涉及区块链的操作，如数据上链和下载，所需时间仅为 0.03 至 0.06 秒，而区块确认时间大约为 1 秒。因此，相对于耗时的模型推理训练过程及每轮训练自身可能出现的时间波动，区块链操作所消耗的时间几乎可以忽略不计。

接下来，为了评估本章所提方案在不同情景下的性能，于三个方面进行比较实验：网络规模、恶意客户端比例和攻击类型。这些都与三个基准进行了对比。

1.4



Patronus

V**-DT**

多Krum

FLDetector

1.2

1.0

0.8

MSE

0.6

0.4

0.2

0.0

10% 15% 20% 25% 30%

恶意客户端比例

图 6-4 不同恶意客户端比例下的 MSE

在图[6-4](#_bookmark141)中，恶意客户端的比例从 10% 变化到 30%，并记录了四种方法在验证集上的表现。观察的信息如下：

* 缺乏防御机制的 **V-DT** 策略难以收敛到合理值。

1.0



Patronus

V**-DT**

多Krum

FLDetector

0.8

0.6

MSE

0.4

0.2

0.0

20 30 40 50

客户端数量

图 6-5 不同客户端数量下的 MSE

* **Patronus** 在MSE 性能上始终优于其他方法。当恶意客户端数量增加时，其优越性尤为明显。
* **FLDetector** 和多 **Krum** 显示出次优结果。**FLDetector** 的表现不佳可以追溯到其依赖于一致的模型梯度进行预测。鉴于实际的模型中毒攻击可能并不总是导致显著的梯度变化，这种方法的有效性受到了损害。与此同时，多**krum** 根据损失方差消除潜在有害的模型更新。然而，在缺乏进一步的恶意客户端检测的情况下，它的表现优于 **FLDetector**，但落后于 **Patronus**。

然后调整网络规模为 20、30、40 和 50 个客户端。图[6-5](#_bookmark142)显示，**V-DT** 仍然无法收敛，导致了严重的错误。相比之下，**Patronus** 能够适应不断增加的网络规模，随着规模的扩大，其性能显著提高。这种改进源于聚合方法，随着聚类样本的增加，该方法在区分 PGs 和 WGs 方面更为出色。多 **Krum** 和 **FLDetector** 无论网络规模如何都保持一致的性能。由于缺乏洗牌策略以及依赖于初始恶意客户端分布来实现收敛，尤其是考虑到恶意客户端比例保持不变，这是它们保持一致的原因。

最后，本节评估了在不同攻击策略下各种方法的性能对比，如图[6-6](#_bookmark143)所示。发现 **Patronus** 优于其他方法，在所有类型的攻击中均展示了最佳结果。其性能显著优于 多 **Krum** 和 **FLDetector**，这两者的结果相当。这证明所提方案在面对多种攻击策略时的安全性保障。

1.0



V-DT

多-Krum

FLDetector

Patronus

0.8

0.6

MSE

0.4

0.2

0.0

###### 引入扰动 随机化输出 梯度反转更新

* 1. 本章小结

图 6-6 不同攻击方式下的 MSE 情况

本章通过以边缘分布式DNN 训练为核心案例，深入探讨了边缘分布式计算环境中计算卸载的安全挑战，并提出了基于链上信息的安全计算卸载方案。该方案巧妙地融合了数据并行与模型并行的策略，优化了训练效率。同时，还设计了一种恶意客户端检测策略。通过多轮迭代的方式，该策略能够识别并隔离出恶意客户端，有效地提升了系统的安全性。此外，借助于区块链技术，所提出的方案确保了分布式计算任务的公正性与透明性，为边缘分布式DNN 计算的安全实施提供了强有力的支持。本研究的提出，不仅为应对边缘分布式计算中的安全挑战提供了有效的解决方案，还为未来边缘分布式DNN 计算的发展提供了宝贵的探索经验。

## 第七章 结束语

#### 论文总结

随着信息化和数字化进程的深入，智能设备的广泛应用及互联网技术的迅速发展，共同促进了新兴产业和应用技术的快速增长。在这样的背景之下，边缘分布式计算作为一种创新的计算范式，因其能在数据产生的近端分散地执行计算任务而得到广泛关注。边缘分布式计算展现出多源性、异构性和动态性三大特点，这些特点不仅为计算提供了新的可能性，同时也引入了一系列挑战。首先，多源性导致计算节点的来源极为广泛，这使得确保节点的可信性变得更加困难，加剧了处理流程追踪的难度，从而对计算任务的安全性和高效性构成威胁。其次，异构性意味着计算节点在性能参数上的差异，这对计算卸载策略的制定和任务分配的优化提出了挑战。另外，动态性反映了网络环境及计算需求和资源供给的持续变化，这对维持计算效率和用户体验提出了更高要求。这些特性共同导致了以下四个主要挑战：（1）节点可信度难以保证：由于计算节点来源多样，从个人设备到企业服务器，其安全性和可靠性参差不齐，难以全面保证，影响计算任务的安全与效率。（2）计算卸载策略僵化：现有的计算卸载策略过于僵化，常基于静态假设，缺乏应对网络和计算需求变化的灵活性，导致计算任务分配不够高效，进而影响计算效率和用户体验。（3）任务卸载的不公平性：如何公平地分配计算任务是一大难题，不平衡的任务分配可能降低系统整体效率，破坏节点参与积极性。（4）计算卸载的可靠性：在分布式环境中，确保每个节点的计算结果的准确性和可靠性至关重要，如果任务被卸载到潜在的恶意节点，将危害模型的准确性。

围绕上述难题挑战，本文从信任基础、弹性卸载、公平化调度以及安全性保障四个方面开展了深入研究。针对节点可信度难以保证的挑战，提出了基于多链协同的可信计算卸载模型；面对计算卸载策略僵化的问题，设计了智能合约驱动的自适应卸载策略；针对任务卸载的不公平性，提出了基于工作量证明的公平化卸载方法；最后，为了解决计算模型可靠性问题，设计了基于链上信息的安全卸载方案。通过这四方面的研究，本文形成了一整套基于区块链的边缘分布式计算卸载研究方案，为提高计算效率、确保计算安全、优化计算效率和提升用户体验提供了坚实的理论和技术支撑。本文研究工作主要可分为以下

四个部分：

第一部分（对应第三章）：在边缘分布式计算环境中，由于计算参与者的多样性和计算场景的复杂性，确保节点间的可信度成为了一大挑战。为了解决这一问题，本文设计并实现了一种基于多链协同的可信计算卸载模型。该模型通过在不同的区块链之间建立信息同步与验证机制，不仅增强了数据处理的透明度与可追溯性，提高计算效率，还利用智能合约自动管理计算任务的分配与执行，从而显著提高了系统的信任度。实验结果证明，这种可信计算模型在提高边缘计算环境的安全性和效率方面有明显优势，为可信分布式计算卸载提出了一种创新的解决策略。

第二部分（对应第四章）：针对计算卸载的静态化和僵化问题，特别是在动态变化的网络环境和不断变动的计算需求背景下，提出了一种智能合约驱动的自适应卸载策略。该策略能够实时调整任务分配，以达到资源使用的最优化。核心算法采用了随机优化技术，能够迅速适应环境变化，确保在复杂条件下计算卸载的高效性和用户体验的优越性。实验结果显示，这种自适应卸载策略显著提升了资源利用效率，减少了延迟，增强了系统的整体性能。

第三部分（对应第五章）：针对多节点协作的计算环境中，如何公平分配计算任务以避免资源浪费和提升节点参与积极性的问题，本研究引入了基于工作量证明的公平化卸载方法。该方法不仅量化了任务分配的公平度，还通过策略的动态调整保证了长期的公平性。采用的工作量证明机制确保了根据计算能力进行任务分配，保障了参与者的公平对待。实验验证表明，该方法能够有效提升任务分配的公平性，进而增强了系统的整体计算效率和稳定性。

第四部分（对应第六章）：针对分布式计算环境中的计算模型可靠性问题，提出了基于链上信息的安全卸载方案。该方案通过引入基于损失感知的信用评估机制和基于洗牌的隔离机制，实现了对参与节点的动态信用评分及基于评分的恶意节点快速隔离。同时，这种区块链驱动的方案有效地提高了计算过程的透明度和结果的可靠性，确保了系统对恶意攻击的抵抗力。实验结果证实，这种安全卸载方案在增强计算安全性方面的同时，保持了系统运行的高效性。

综上所述，本文对基于区块链的边缘分布式计算卸载的关键技术难题进行了深入的探索。针对其面临的一系列关键挑战，从信任基础、自适应计算卸载策略、公平性任务分配以及安全性保障四个核心方面进行了全面深入的研究。通过对边缘计算节点的可信度提升、计算任务的弹性卸载、任务调度的公平化处理以及计算安全性的增强，本研究构建了一套完整的基于区块链的边缘分布式计算卸载优化研究方案，有效地提高了系统的整体性能。研究成果不仅包括发

表的高质量学术论文，还涵盖了一系列专利申请，体现了本文的创新性和实际应用价值。在研究方法上，本文采用了问题引入、数学建模、算法设计、性能分析和实验验证等研究步骤，针对性地解决了边缘分布式计算环境下关键的技术难题。通过这一系列科学严谨的研究方法，本文不仅为边缘分布式计算卸载提供了理论和技术支撑，也为未来在类似领域的研究提供了宝贵的经验和参考。

#### 未来研究工作

随着信息技术和硬件设备的飞速发展，边缘分布式计算作为一种新兴的计算模式，正面临着日益复杂的应用场景和持续升级的技术要求。这些趋势不仅带来了新的计算需求和网络环境，也对现有的计算卸载及资源优化机制提出了更高挑战。基于个人能力和研究条件的局限，尽管本研究在一定程度上解决了边缘分布式计算卸载中的部分关键问题，但仍有改进和深入研究的空间。因此， 作者计划从以下几个方面对本文的研究成果进行优化和拓展：

* + 1. 在第三章提出的基于多链协同的可信计算卸载模型中，以人工智能的计算卸载算法为例，探索通过智能合约与神经网络紧密结合的方法，以提升边缘计算环境中节点的可信度和计算效率。然而智能合约和神经网络的工程实现面临许多挑战。例如代码兼容性和区块验证效率问题，特别是如何高效集成深度学习模型至区块链平台。同时，确保智能合约执行的效率同时保障计算结果的准确性和可靠性将是研究的焦点。此外，优化智能合约的编写和执行机制，减少区块链计算和存储开销也将是未来工作的一部分。
    2. 对于第四章提出的自适应计算卸载策略，虽然能够根据网络环境和计算需求变化动态调整任务分配，但进一步降低移动设备能耗仍是重要考量。未来研究将注重如何在保障计算卸载效率的同时，进一步降低移动设备的能耗， 这可能包括开发更高效的算法来预测任务执行能耗，以及设计能自适应调整计算负载和通信策略的智能系统，以最小化能耗而不牺牲任务执行效果。
    3. 第五章提出的基于工作量证明的公平化卸载方法有效地提升了任务分配的公平性，但在实际应用中，鉴于不同任务对执行时间的敏感度各异，对于处理不同优先级任务的灵活性尚显不足。未来的研究努力将旨在开发一种更加智能化的任务分配机制。该机制将能够全面考虑任务的紧急度、计算复杂度以及节点的当前状态，以此为基础动态调整任务优先级和分配策略。这样不仅能确保高优先级任务得到及时处理，同时在保障系统整体效率的基础上，还将进一步提升计算卸载公平性。
    4. 在提升计算模型安全性的同时，保护数据隐私是另一个重要的研究方向。特别是在第六章提出的基于链上信息的安全卸载方案中，如何在不泄露用户敏感信息前提下完成数据验证和计算，是亟需解决问题。未来工作将探索使用先进加密技术，如同态加密和零知识证明等，实现数据安全处理和计算，确保用户数据隐私不被泄露，同时不影响计算准确性和效率。

由于作者的研究水平和知识储备的限制，论文中难免存在不足之处，敬请各位评审专家和读者批评指正，在此表示衷心的感谢！

## 参考文献

1. 超高清视频产业发展白皮书（2022 年）[R].
2. 张立. 虚拟现实产业发展白皮书（2023 年）[R]. 2023. 在线获取: [https://ccidgroup.com/ info/2080/38581.htm](https://ccidgroup.com/info/2080/38581.htm).
3. Patel S Y, Mehrotra A, Huskamp H A, et al. Variation in telemedicine use and outpatient care during the COVID-19 pandemic in the United States: study examines variation in total US outpatient visits and telemedicine use across patient demographics, specialties, and conditions during the COVID-19 pandemic.[J]. Health Affairs, 2021, 40(2): 349–358.
4. 元宇宙白皮书（2023 年）[R]. 2023. 在线获取: [http://www.caict.ac.cn/english/research/ whitepapers/202312/P020231211384567117786.pdf](http://www.caict.ac.cn/english/research/whitepapers/202312/P020231211384567117786.pdf).
5. Voigt P, Von dem Bussche A. The eu general data protection regulation (gdpr)[J]. A Practical Guide, 1st Ed., Cham: Springer International Publishing, 2017, 10(3152676): 10–5555.
6. Zaharia M, Xin R S, Wendell P, et al. Apache spark: a unified engine for big data process- ing[J]. Communications of the ACM, 2016, 59(11): 56–65.
7. Dittrich J, Quiané-Ruiz J A. Efficient big data processing in Hadoop MapReduce[J]. Proceed- ings of the VLDB Endowment, 2012, 5(12): 2014–2015.
8. Li T, Sahu A K, Talwalkar A, et al. Federated learning: Challenges, methods, and future directions[J]. IEEE signal processing magazine, 2020, 37(3): 50–60.
9. Schurgot M R, Wang M, Conway A E, et al. A dispersed computing architecture for resource- centric computation and communication[J]. IEEE Communications Magazine, 2019, 57(7): 13–19.
10. Wu H, Chen J, Nguyen T N, et al. Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems[J]. IEEE Transactions on Industrial Informatics, 2022, 19(2): 2117– 2128.
11. Qi L, Hu C, Zhang X, et al. Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment[J]. IEEE Transactions on Industrial Informatics, 2020, 17(6): 4159–4167.
12. Atta A, Abbas S, Khan M A, et al. An adaptive approach: Smart traffic congestion control system[J]. Journal of King Saud University-Computer and Information Sciences, 2020, 32(9): 1012–1019.
13. Afzali M, Pourmohammadi H, Mohammad Vali Samani A. An efficient framework for trust evaluation of secure service selection in fog computing based on QoS, reputation, and social criteria[J]. Computing, 2022, 104(7): 1643–1675.
14. Adewuyi A A, Cheng H, Shi Q, et al. Sc-trust: a dynamic model for trustworthy service composition in the internet of things[J]. IEEE Internet of Things Journal, 2021, 9(5): 3298– 3312.
15. Al-Khafajiy M, Baker T, Asim M, et al. COMITMENT: A fog computing trust management approach[J]. Journal of Parallel and Distributed Computing, 2020, 137: 1–16.
16. Liang J, Zhang M, Leung V C. A reliable trust computing mechanism based on multisource feedback and fog computing in social sensor cloud[J]. IEEE Internet of Things Journal, 2020, 7(6): 5481–5490.
17. Wang X, Sui Y, Wang J, et al. A distributed truthful auction mechanism for task allocation in mobile cloud computing[J]. IEEE Transactions on Services Computing, 2018, 14(3): 628– 638.
18. Alamer A, Basudan S. An efficient truthfulness privacy-preserving tendering framework for vehicular fog computing[J]. Engineering Applications of Artificial Intelligence, 2020, 91: 103583.
19. Xu X, Liu X, Xu Z, et al. Trust-oriented IoT service placement for smart cities in edge com- puting[J]. IEEE Internet of Things Journal, 2019, 7(5): 4084–4091.
20. Xu X, Shen B, Yin X, et al. Edge server quantification and placement for offloading social media services in industrial cognitive IoV[J]. IEEE Transactions on Industrial Informatics, 2020, 17(4): 2910–2918.
21. Song H, Gu B, Son K, et al. Joint optimization of edge computing server deployment and user offloading associations in wireless edge network via a genetic algorithm[J]. IEEE Transac- tions on Network Science and Engineering, 2022, 9(4): 2535–2548.
22. Mtibaa A, Harras K A, Fahim ATowards computational offloading in mobile device clouds.

// 2013 IEEE 5th international conference on cloud computing technology and science[C]. Bristol, UK: 2013: 331–338.

1. Zhu Y, He Q, Liu J, et al. When crowd meets big video data: Cloud-edge collaborative transcoding for personal livecast[J]. IEEE Transactions on Network Science and Engineering, 2018, 7(1): 42–53.
2. Wang L, Xu Y, Xu H, et al. Accelerating decentralized federated learning in heterogeneous edge computing[J]. IEEE Transactions on Mobile Computing, 2023, 22: 5001–5016.
3. Yang L, Gan Y, Cao J, et al. Optimizing aggregation frequency for hierarchical model training in heterogeneous edge computing[J]. IEEE Transactions on Mobile Computing, 2022, 22(7): 4181–4194.
4. Chen C, Zeng Y, Li H, et al. A multihop task offloading decision model in mec-enabled internet of vehicles[J]. IEEE Internet of Things Journal, 2022, 10(4): 3215–3230.
5. Yadav R, Zhang W, Kaiwartya O, et al. Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing[J]. IEEE Transactions on Vehicular Technology, 2020, 69(12): 14198–14211.
6. Saleem U, Liu Y, Jangsher S, et al. Latency minimization for D2D-enabled partial compu- tation offloading in mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2020, 69(4): 4472–4486.
7. Yuan L, He Q, Chen F, et alPipeEdge: A trusted pipelining collaborative edge training based on blockchain. // Proceedings of the ACM Web Conference 2023[C]. New York, USA: 2023: 3033–3043.
8. Al-Habob A A, Dobre O A, Armada A G, et al. Task scheduling for mobile edge computing using genetic algorithm and conflict graphs[J]. IEEE Transactions on Vehicular Technology, 2020, 69(8): 8805–8819.
9. Rahimzadeh P, Lee J, Im Y, et alSPARCLE: Stream processing applications over dispersed computing networks. // 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)[C]. Singapore, Singapore: 2020: 1067–1078.
10. Zhou X, He D, Ning J, et al. Aadec: Anonymous and auditable distributed access control for edge computing services[J]. IEEE Transactions on Information Forensics and Security, 2022, 18: 290–303.
11. Xiao Y, Jia Y, Liu C, et al. Edge computing security: State of the art and challenges[J]. Proceedings of the IEEE, 2019, 107(8): 1608–1631.
12. Chen Y, Yang Q, He S, et al. Ftpipehd: A fault-tolerant pipeline-parallel distributed training approach for heterogeneous edge devices[J]. IEEE Transactions on Mobile Computing, 2023, 23(4): 3200–3212.
13. Deng X, Li J, Ma C, et al. Low-latency federated learning with DNN partition in distributed industrial IoT networks[J]. IEEE Journal on Selected Areas in Communications, 2022, 41(3): 755–775.
14. Fang M, Cao X, Jia J, et alLocal model poisoning attacks to Byzantine-Robust federated learning. // 29th USENIX security symposium (USENIX Security 20)[C]. Boston, USA: 2020: 1605–1622.

{ }

1. Lyu X, Han Y, Wang W, et alPoisoning with cerberus: Stealthy and colluded backdoor at- tack against federated learning. // Proceedings of the AAAI Conference on Artificial Intelli- gence[C]. Washington DC, USA: 2023: 9020–9028.
2. Liu K, Dolan-Gavitt B, Garg SFine-pruning: Defending against backdooring attacks on deep neural networks. // International symposium on research in attacks, intrusions, and defenses[C]. 2018: 273–294.
3. Shubha S S, Shen HTrustworthy Distributed Deep Neural Network Training in an Edge De- vice Network. // 2022 IEEE International Conference on Big Data (Big Data)[C]. Osaka, Japan: 2022: 1570–1575.
4. Luo S, Li H, Wen Z, et al. Blockchain-based task offloading in drone-aided mobile edge computing[J]. IEEE Network, 2021, 35(1): 124–129.
5. Wang M, Xu C, Chen X, et al. BC-mobile device cloud: A blockchain-based decentralized truthful framework for mobile device cloud[J]. IEEE Transactions on Industrial Informatics, 2020, 17(2): 1208–1219.
6. Turgay S. Blockchain management and federated learning adaptation on healthcare manage- ment system[J]. International Journal of Intelligent Systems and Applications (IJISA), 2022, 14(5): 1–13.
7. Li J, Li D, Zhang X. A secure blockchain-assisted access control scheme for smart healthcare system in fog computing[J]. IEEE Internet of Things Journal, 2023, 10(18): 15980–15989.
8. Rawat D B. Fusion of software defined networking, edge computing, and blockchain technol- ogy for wireless network virtualization[J]. IEEE Communications Magazine, 2019, 57(10): 50–55.
9. Zhu Y, Liu J, Wang Z, et alWhen cloud meets uncertain crowd: An auction approach for crowdsourced livecast transcoding. // Proceedings of the 25th ACM international conference on Multimedia[C]. Mountain View California, USA: 2017: 1372–1380.
10. Zavodovski A, Bayhan S, Mohan N, et alDeCloud: Truthful decentralized double auction for edge clouds. // 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)[C]. Dallas, TX, USA: 2019: 2157–2167.
11. Qu Y, Pokhrel S R, Garg S, et al. A blockchained federated learning framework for cognitive computing in industry 4.0 networks[J]. IEEE Transactions on Industrial Informatics, 2020, 17(4): 2964–2973.
12. Cheng H, Hu Q, Zhang X, et al. Trusted resource allocation based on smart contracts for blockchain-enabled internet of things[J]. IEEE Internet of Things Journal, 2021, 9(11): 7904– 7915.
13. Yang P, Lyu F, Wu W, et al. Edge coordinated query configuration for low-latency and accurate video analytics[J]. IEEE Transactions on Industrial Informatics, 2019, 16(7): 4855–4864.
14. Jo M, Maksymyuk T, Strykhalyuk B, et al. Device-to-device-based heterogeneous radio ac- cess network architecture for mobile cloud computing[J]. IEEE Wireless Communications, 2015, 22(3): 50–58.
15. Zhang X, Xue G, Yu R, et al. Keep your promise: Mechanism design against free-riding and false-reporting in crowdsourcing[J]. IEEE Internet of Things Journal, 2015, 2(6): 562–572.
16. Nakamoto S, Bitcoin A. A peer-to-peer electronic cash system[J]. Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf, 2008, 4(2): 15.
17. Wang X, Garg S, Lin H, et al. A secure data aggregation strategy in edge computing and blockchain-empowered internet of things[J]. IEEE Internet of Things Journal, 2020, 9(16): 14237–14246.
18. WiKi. Proof of Authority[EB/OL]. [https://github.com/paritytech/parity/wiki/ Proof-of-Authority-Chains](https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains). [Accessed March 24, 2024].
19. Neely M. Stochastic network optimization with application to communication and queueing systems[M]. Springer Nature, 2022.
20. Kaur K, Garg S, Aujla G S, et al. Edge computing in the industrial internet of things envi- ronment: Software-defined-networks-based edge-cloud interplay[J]. IEEE communications magazine, 2018, 56(2): 44–51.
21. Qiu T, Li B, Qu W, et al. TOSG: A topology optimization scheme with global small world for industrial heterogeneous Internet of Things[J]. IEEE Transactions on Industrial Informatics, 2018, 15(6): 3174–3184.
22. Li H, Savkin A V. Wireless sensor network based navigation of micro flying robots in the in- dustrial internet of things[J]. IEEE Transactions on industrial informatics, 2018, 14(8): 3524– 3533.
23. Kong X, Xia F, Li J, et al. A shared bus profiling scheme for smart cities based on hetero- geneous mobile crowdsourced data[J]. IEEE Transactions on Industrial Informatics, 2019, 16(2): 1436–1444.
24. Wang M, Xu C, Chen X, et al. Differential privacy oriented distributed online learning for mobile social video prefetching[J]. IEEE Transactions on Multimedia, 2019, 21(3): 636–651.
25. Zhang F, Deng R, Liang H. An optimal real-time distributed algorithm for utility maximiza- tion of mobile ad hoc cloud[J]. IEEE Communications Letters, 2018, 22(4): 824–827.
26. Wang X, Chen X, Wu WTowards truthful auction mechanisms for task assignment in mo- bile device clouds. // IEEE INFOCOM 2017-IEEE Conference on Computer Communica- tions[C]. Atlanta, GA, USA: 2017: 1–9.
27. Poon J, Buterin V. Plasma: Scalable autonomous smart contracts[J]. White paper, 2017: 1– 47.
28. Wang M, Xu C, Chen X, et al. Design of multipath transmission control for information- centric Internet of Things: A distributed stochastic optimization framework[J]. IEEE Internet of Things Journal, 2019, 6(6): 9475–9488.
29. Jiao Y, Wang P, Niyato D, et al. Auction mechanisms in cloud/fog computing resource al- location for public blockchain networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(9): 1975–1989.
30. King S, Nadal S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. self-published paper, August, 2012, 19(1).
31. Larimer D. Delegated proof-of-stake (dpos)[J]. Bitshare whitepaper, 2014, 81: 85.
32. Jang I, Choo S, Kim M, et al. The software-defined vehicular cloud: A new level of sharing the road[J]. ieee vehicular technology magazine, 2017, 12(2): 78–88.
33. Johnson D B, Maltz D A. Dynamic source routing in ad hoc wireless networks[J]. Mobile computing, 1996: 153–181.
34. Network O. Plasma-MVP[EB/OL]. <https://github.com/omisego/plasma-mvp>. [Accessed March 24, 2024].
35. Wood G, others. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1–32.
36. Yu J, Chen H, Wu K, et al. Centipede: Leveraging the distributed camera crowd for cooper- ative video data storage[J]. IEEE Internet of Things Journal, 2021, 8(22): 16498–16509.
37. Khan A, Gupta S, Gupta S K. Unmanned aerial vehicle-enabled layered architecture based solution for disaster management[J]. Transactions on Emerging Telecommunications Tech- nologies, 2021, 32(12): e4370.
38. Kumar S, Kumar S, Lobiyal D K. MWLP-DP: Mobile war-fighters location prediction for dark phase in Internet of Battlefield Things[J]. Transactions on Emerging Telecommunica- tions Technologies, 2022, 33(4): e4397.
39. Dimitrakopoulos G, Demestichas P. Intelligent transportation systems[J]. IEEE Vehicular Technology Magazine, 2010, 5(1): 77–84.
40. Drolia U, Guo K, Narasimhan PPrecog: Prefetching for image recognition applications at the edge. // Proceedings of the Second ACM/IEEE Symposium on Edge Computing[C]. San Jose California, USA: 2017: 1–13.
41. Wang B, Tao F, Fang X, et al. Smart manufacturing and intelligent manufacturing: A com- parative review[J]. Engineering, 2021, 7(6): 738–757.
42. Stokkink Q, Ileri C U, Epema D H J, et al. Web3 Sybil avoidance using network latency[J]. Comput. Networks, 2023, 227: 109701.
43. Zalta E N, Nodelman U, Allen C, et al. Stanford encyclopedia of philosophy[M]. Metaphysics Research Lab, Center for the Study of Language and Information, 1995.
44. Simon R L. The Blackwell guide to social and political philosophy[M]. John Wiley & Sons, 2008.
45. Lin Y, Gao Z, Du H, et al. A unified blockchain-semantic framework for wireless edge intel- ligence enabled web 3.0[J]. IEEE Wireless Communications, 2023: 1–9.
46. Enciso A R, Skarmeta A F. A multi-layer guided reinforcement learning-based tasks offload- ing in edge computing[J]. Comput. Networks, 2023, 220: 109476.
47. Hu N, Tian Z, Du X, et al. Deep-Green: A dispersed energy-efficiency computing paradigm for green industrial IoT[J]. IEEE Transactions on Green Communications and Networking, 2021, 5(2): 750–764.
48. Wang J, Hu J, Min G, et al. Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning[J]. IEEE Communications Magazine, 2019, 57(5): 64–69.
49. Zhang Y, Li R, Zhao Y, et alA request scheduling optimization mechanism based on deep Q-learning in edge computing environments. // IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)[C]. Vancouver, BC, Canada: 2021: 1–2.
50. Bi X, Sun X, Lyu Z, et al. A back adjustment based dependent task offloading scheduling algo- rithm with fairness constraints in VEC networks[J]. Comput. Networks, 2023, 223: 109552.
51. Mlika Z, Cherkaoui S. Network slicing with MEC and deep reinforcement learning for the Internet of Vehicles[J]. IEEE Network, 2021, 35(3): 132–138.
52. Werner S, Perez D, Gudgeon L, et alSok: Decentralized finance (defi). // Proceedings of the 4th ACM Conference on Advances in Financial Technologies[C]. Cambridge, USA: 2022: 30–46.
53. Wikipedia. Decentralized Autonomous Organization[EB/OL]. [https://en.wikipedia.org/wiki/ Decentralized\_autonomous\_organization](https://en.wikipedia.org/wiki/Decentralized_autonomous_organization). [Accessed March 24, 2024].
54. Eyal I, Sirer E G. Majority is not enough: Bitcoin mining is vulnerable[J]. Communications of the ACM, 2018, 61(7): 95–102.
55. Hafid A, Hafid A S, Samih M. Scaling blockchains: A comprehensive survey[J]. IEEE access, 2020, 8: 125244–125262.
56. Peng B, Hosseini M, Hong Z, et alR-storm: Resource-aware scheduling in storm. // Proceed- ings of the 16th annual middleware conference[C]. Vancouver BC, Canada: 2015: 149–161.
57. Kao Y H, Krishnamachari B, Ra M R, et al. Hermes: Latency optimal task assignment for resource-constrained mobile computing[J]. IEEE Transactions on Mobile Computing, 2017, 16(11): 3056–3069.
58. Lai Z, Hu Y C, Cui Y, et alFurion: Engineering high-quality immersive virtual reality on to- day’s mobile devices. // Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking[C]. Snowbird Utah, USA: 2017: 409–421.
59. Androulaki E, Barger A, Bortnikov V, et alHyperledger fabric: a distributed operating system for permissioned blockchains. // Proceedings of the thirteenth EuroSys conference[C]. Porto, Portugal: 2018: 1–15.
60. Wang S, Ye D, Huang X, et al. Consortium blockchain for secure resource sharing in vehicular edge computing: A contract-based approach[J]. IEEE Transactions on Network Science and Engineering, 2020, 8(2): 1189–1201.
61. Tian J, Tian J, Xu H. TSBFT: A scalable and efficient leaderless byzantine consensus for consortium blockchain[J]. Comput. Networks, 2023, 222: 109541.
62. Sompolinsky Y, Zohar ASecure high-rate transaction processing in bitcoin. // Financial Cryp- tography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19[C]. Bristol, UK: 2015: 507–527.
63. Dai Y, Xu D, Maharjan S, et al. Joint computation offloading and user association in multi- task mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2018, 67(12): 12313–12325.
64. Zhao G, Xu H, Zhao Y, et al. Offloading tasks with dependency and service caching in mobile edge computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(11): 2777–2792.
65. OMNeT++. Internet Topology Generator[EB/OL]. [https://inet.omnetpp.org](https://inet.omnetpp.org/). [Accessed March 24, 2024].
66. Xiao H, Xu C, Ma Y, et al. Edge intelligence: A computational task offloading scheme for dependent IoT application[J]. IEEE Transactions on Wireless Communications, 2022, 21(9): 7222–7237.
67. Jain R K, Chiu D M W, Hawe W R, et al. A quantitative measure of fairness and discrimi- nation[J]. Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA, 1984, 21.
68. Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877–1901.
69. Anil R, Dai A M, Firat O, et al. Palm 2 technical report[J]. arXiv preprint arXiv:2305.10403, 2023.
70. Liu X F, Zhan Z H, Deng J D, et al. An energy efficient ant colony system for virtual machine placement in cloud computing[J]. IEEE transactions on evolutionary computation, 2016, 22(1): 113–128.
71. Dong C, Wen W, Xu T, et al. Joint optimization of data-center selection and video-streaming distribution for crowdsourced live streaming in a geo-distributed cloud platform[J]. IEEE Transactions on Network and Service Management, 2019, 16(2): 729–742.
72. Li Y, Yang C, Zhao P, et alTowards real-time segmentation on the edge. // Proceedings of the AAAI Conference on Artificial Intelligence[C]. Washington DC, USA: 2023: 1468–1476.
73. Jiang K, Sun C, Zhou H, et al. Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook[J]. IEEE Network, 2021, 35(5): 74–82.
74. Unger C, Jia Z, Wu W, et alUnity: Accelerating DNN training through joint optimization of algebraic transformations and parallelization. // 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)[C]. Carlsbad, USA: 2022: 267–284.

{ }

1. Huang Y, Cheng Y, Bapna A, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism[J]. Advances in neural information processing systems, 2019, 32.
2. Narayanan D, Harlap A, Phanishayee A, et alPipeDream: Generalized pipeline parallelism for DNN training. // Proceedings of the 27th ACM Symposium on Operating Systems Prin- ciples[C]. Huntsville Ontario, Canada: 2019: 1–15.
3. Li M, Andersen D G, Park J W, et alScaling distributed machine learning with the parameter server. // 11th USENIX Symposium on operating systems design and implementation (OSDI 14)[C]. Broomfield, USA: 2014: 583–598.
4. Wang D, Shen L, Luo Y, et al. FedABC: Targeting Fair Competition in Personalized Federated Learning[J]. arXiv preprint arXiv:2302.07450, 2023.
5. Fan S, Rong Y, Meng C, et alDAPPLE: A pipelined data parallel approach for training large models. // Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming[C]. Virtual Event Republic of Korea: 2021: 431–445.
6. Iandola F N, Moskewicz M W, Ashraf K, et alFirecaffe: near-linear acceleration of deep neu- ral network training on compute clusters. // Proceedings of the IEEE conference on computer vision and pattern recognition[C]. Las Vegas, USA: 2016: 2592–2600.
7. Zheng L, Li Z, Zhang H, et alAlpa: Automating inter-and Intra-Operator parallelism for distributed deep learning. // 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)[C]. Carlsbad, USA: 2022: 559–578.

{ }

1. Mai L, Li G, Wagenländer M, et al KungFu : Making training in distributed machine learn- ing adaptive. // 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)[C]. San Diego, USA: 2020: 937–954.

{ }

1. Tian Z, Cui L, Liang J, et al. A comprehensive survey on poisoning attacks and countermea- sures in machine learning[J]. ACM Computing Surveys, 2022, 55(8): 1–35.
2. Ma Z, Ma J, Miao Y, et al. ShieldFL: Mitigating model poisoning attacks in privacy- preserving federated learning[J]. IEEE Transactions on Information Forensics and Security, 2022, 17: 1639–1654.
3. Li L, Xu W, Chen T, et alRSA: Byzantine-robust stochastic aggregation methods for dis- tributed learning from heterogeneous datasets. // Proceedings of the AAAI Conference on Artificial Intelligence[C]. Hawaii, USA: 2019: 1544–1551.
4. Blanchard P, El Mhamdi E M, Guerraoui R, et al. Machine learning with adversaries: Byzan- tine tolerant gradient descent[J]. Advances in neural information processing systems, 2017, 30.
5. Metzen J H, Genewein T, Fischer V, et al. On detecting adversarial perturbations[J]. arXiv preprint arXiv:1702.04267, 2017.
6. Wu Z, Xu C, Chen X, et al. BC-MetaCast: A Blockchain-Enhanced Intelligent Computing Framework for Metaverse Livecast[J]. IEEE Network, 2023, 37(4): 161–168.
7. Chen Y, Yang Q, He S, et al. FTPipeHD: A Fault-Tolerant Pipeline-Parallel Distributed Train- ing Approach for Heterogeneous Edge Devices[J]. IEEE Transactions on Mobile Computing, 2023, 23(4): 3200–3212.
8. Shejwalkar V, Houmansadr AManipulating the byzantine: Optimizing model poisoning at- tacks and defenses for federated learning. // NDSS[C]. held virtually: 2021.
9. Wikipedia. Roulette wheel selection[EB/OL]. [https://computersciencewiki.org/index.php/ Roulette\_wheel\_selection](https://computersciencewiki.org/index.php/Roulette_wheel_selection). [Accessed March 24, 2024].
10. Hu J S, Kuai T, Waslander S LPoint density-aware voxels for lidar 3d object detection. // Pro- ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition[C]. New Orleans, USA: 2022: 8469–8478.
11. Wikipedia. Full width at half maximum[EB/OL]. [https://en.wikipedia.org/wiki/Full\_width\_ at\_half\_maximum](https://en.wikipedia.org/wiki/Full_width_at_half_maximum). [Accessed March 24, 2024].
12. Ikotun A M, Ezugwu A E, Abualigah L, et al. K-means clustering algorithms: A comprehen- sive review, variants analysis, and advances in the era of big data[J]. Information Sciences, 2023, 622: 178–210.
13. Micali S, Rabin M, Vadhan SVerifiable random functions. // 40th annual symposium on foundations of computer science (cat. No. 99CB37039)[C]. New York, USA: 1999: 120– 130.
14. Gilad Y, Hemo R, Micali S, et alAlgorand: Scaling byzantine agreements for cryptocurren- cies. // Proceedings of the 26th symposium on operating systems principles[C]. Shanghai, China: 2017: 51–68.
15. Meyn S P, Tweedie R L. Markov chains and stochastic stability[M]. Springer Science & Business Media, 2012.
16. Pytorch. TORCH.DISTRIBUTED[EB/OL]. <https://pytorch.org/docs/stable/distributed.html>. [Accessed March 24, 2024].
17. Zhou H. ETDataset[EB/OL]. <https://github.com/zhouhaoyi/ETDataset>. [Accessed March 24, 2024].
18. Irvine U. ElectricityLoadDiagrams[EB/OL]. [https://archive.ics.uci.edu/dataset/321/ electricityloaddiagrams20112014](https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014). [Accessed March 24, 2024].
19. Zhou H, Zhang S, Peng J, et alInformer: Beyond efficient transformer for long sequence time-series forecasting. // Proceedings of the AAAI conference on artificial intelligence[C]. held virtually: 2021: 11106–11115.
20. Li Y, Yu R, Shahabi C, et al. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting[J]. arXiv preprint arXiv:1707.01926, 2017.
21. Yuan L, He Q, Chen F, et alPipeEdge: A Trusted Pipelining Collaborative Edge Training based on Blockchain. // Proceedings of the ACM Web Conference 2023[C]. Austin TX, USA: 2023: 3033–3043.
22. Zhang Z, Cao X, Jia J, et alFLDetector: Defending federated learning against model poi- soning attacks via detecting malicious clients. // Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining[C]. Washington DC, USA: 2022: 2545–2555.

## 附录 **A** 第四章中相关定理证明

* 1. 定理[**4.1**](#_bookmark65)的证明

证明： 定理[**4.1**](#_bookmark65)的证明与文献[[55](#_bookmark203)] 中定理 4.1 的证明类似。主要思想是，假设存

在一个随机决策序列 {*xi* (*t*) |*i* ∈ *N* }*T* ，它满足 *d*′ (*t*) ≤ ε*,* ∀ *j* ∈ *J*。给定 *x*∗ (*t*) 是

*t*=1 *j i*

**P1** 的最优解，我们有以下不等式：

∆*Lq* (*t*) + *V* ∑ *Ci* (*xi*∗ (*t*))

*i*∈*N*

≤*B* + *V* ∑ *Ci* (*xi* (*t*)) + ∑ *Qj* (*t*) *d*′ (*t*)

*j*

(A-1)

*i*∈*N j*∈*J*

≤*B* + *V* ∑ *Ci* (*xi* (*t*)) + ε ∑ *Qj* (*t*)

*i*∈*N j*∈*J*

将所有时间段的 [(A-1](#_bookmark286)) 求和并重新排列，我们由此证明了定理[**4.1**](#_bookmark65)。 □

* 1. 定理[**4.2**](#_bookmark66)的证明

证明： 因为 *xi*∗ (*t*) 是 **P3** 的优化，我们有以下不等式:

∆*Lq* (*t*) + *V* ∑ *Ci* (*xi*∗ (*t*))

*i*∈*N*

≤*B* + *V* ∑ *Ci* (*xi*∗ (*t*)) + ∑ *Qj* (*t*) *d j* (*t*)

(A-2)

*i*∈*N j*∈*J*

≤*B* + *V* ∑ *Ci* （*xopt* ） + ∑ *Qj* (*t*) *dopt* (*t*)

*i*

*i*∈*N*

将所有时间段的 ([A-2](#_bookmark288)) 求和并除以 *T* :

*j*

*j*∈*J*

1 *L T*

1 *T VC*

*x*∗ *t*

*T q* (

) + ∑

*t*=1

*T*

*i* ( *i* ( ))

(A-3)

∑ （ *opt* ）

≤*TB* + *V*

*i*∈*N*

*Ci*

*xi*

*a*

由于∑*T* 1 *dopt* (*t*) = ∑*T* 1 （*Rj* (*t*) − ∑ *xi j* (*t*)＼，不等性 *a* 成立。因为 *Lq* (*T* ) ≥ 0, 此

*t*= *j t*=

*i*∈*N*

定理得证。 □

* 1. 定理[**4.3**](#_bookmark67)的证明

证明： 首先考虑 *i* 的四种不同情况：(1) 如果 *xi*∗(*t*) *>* 0，工作者 *i* 提交 *Ci*′ (*x*) 来替代 *Ci* (*x*), 其中 *Ci*′ (*x*) *< Ci* (*x*). 根据算法 **1**，如果工作者 *i* 在时间 *t* 内被选中提供资源，则 *xi*∗ (*t*) = *Hi* (*t*) 用于最小化总成本。假设 *i* 在 *Ci* (*.*) 时已贡献了他的所有资源，我们有 *xi*′ (*t*) = *xi*∗ (*t*)，其中 *xi*′ (*t*) 是在应用 *Ci*′ (*.*) 时贡献的资源。因为奖励方案使用 *Ct* (*.*) 来计算总奖励，我们有

*R* (*t*) = *R* (*t*) *xi*∗ (*t*)

*xi*∗ (*t*)

−

= *R* (*t*) *xi*∗′ (*t*) − *xi*∗′ (*t*) = *R* (*t*)

*i* ∑

*i*∈*W* (*t*)

*xi*∗ (*t*)

*Hi* (*t*)

∑

*i*∈*W* (*t*)

*xi*∗ (*t*)

*Hi* (*t*) *i*

因此，*i* 通过提交较小的成本因子 *Ci*′ (*t*) 无法获得任何好处。

1. 如果 *xi*∗(*t*) = 0，工作者 *i* 设定其提交的 *Ci*′ (*x*) *> Ci* (*x*)。因为 *Ci*′ (*x*) ≤ *Ct* (*x*)， 我们仍然有 *Ri* (*t*) = *R*′*i* (*t*)。即，*i* 通过提交更大的成本因子 *Ci*′ (*.*) 无法获得任何好处。
2. 如果 *xi*∗(*t*) = 0，工作者 *i* 设定其提交的 *Ci*′ (*x*) *< Ci* (*x*)。回忆 *xi*′ (*t*) 表示在伪造的成本-可靠因子 *Ci*′ (*x*) 下贡献的资源量。如果 *xi*′ (*t*) *>* 0，即，*i* 通过伪造成本-可靠因子被选为工作者。根据奖励方案，*i* 的实际收入可以由下式给出：

*Ct* (*xi*∗ (*t*)) −*Ci* (*xi*∗ (*t*)) *<* 0

这是因为 *Ci* (*.*) *> Ct* (*.*)，否则 *xi*∗(*t*) *>* 0。因此，这种情况下的奖励是负的。

1. 如果 *xi*∗(*t*) = 0，工作者 *i* 设定其提交的 *Ci*′ (*x*) *> Ci* (*x*)。显然，考虑到任务分配算法总是选择成本-可靠因子较小的工作者，所以 *xi*′(*t*) = 0。因此，这种情况下的收入为 0。

基于上述分析，我们可以得出结论，任何工作者通过操纵 *Ci* (*.*) 都无法增加其收入，因此，定理[**4.3**](#_bookmark67)成立。 □

附录 **B** 缩略语表

5G Fifth-Generation Mobile Communication Technology，第五代移动通信技术AWGN Additive White Gaussian Noise，加性白高斯噪声

D2D Device to Device，设备到设备 DAG Directed Acyclic Graph，有向无环图

DNN Deep Neural Network，深度神经网络DPoS Delegated Proof of Stake，权益委托证明DQN Deep Q-Network，深度 Q 网络

LAN Local Area Network，局域网

MDP Markov Decision Process，马尔可夫决策过程MDPP Min-Drift-Plus-Penalty，最小漂移加惩罚MSE Mean Square Error，均方误差损失

NCP Network Computing Point，网络计算节点P2P Peer to Peer，点对点

PoS Proof of Stake，权益证明

PoW Proof of Work，工作量证明

TPS Transaction Per Second，每秒交易数VR Virtual Reality，虚拟现实

XR Extended Reality，增强现实

## 致 谢

## 攻读学位期间取得的“创新成果”目录

期刊论文

1. **Wu Zhonghui** , Xu C, Chen X, Ma Y, Zhong L, Zhang H, et al. BC-MetaCast: A Blockchain- Enhanced Intelligent Computing Framework for Metaverse Livecast[J]. 2023, 37(4): 161–168.

（JCR 一区，中科院二区，IF：9.3）.

1. 马云霄, 吴忠辉 , 徐祖云, 衷璐洁, 许长桥. 算力网络中面向计算重用的任务调度优化 [J]. 2023, 44(11): 129–142. （中国卓越期刊梯队期刊）.
2. Ma Y, **Wu Zhonghui** , Yang S, Zhuang Y, Xiao H. DMUST360: Distortion-aware multicast- unicast superposed transmission for multiuser 360-degree video[J]. 2024, 35(3): e4954. （SCI 收录，中科院四区）.
3. Chen X, Xu C, Wang M, **Wu Zhonghui** , Zhong L, Grieco L A. Augmented queue-based trans- mission and transcoding optimization for livecast services based on cloud-edge-crowd integra- tion[J]. 2020, 31(11): 4470–4484. （中科院一区，IF：8.4）.
4. Wang M, Xu C, Chen X, Zhong L, **Wu Zhonghui** , Wu D O. BC-mobile device cloud: A blockchain-based decentralized truthful framework for mobile device cloud[J]. 2020, 17(2): 1208–1219. （中科院一区，IF: 12.3）.
5. **Wu Zhonghui** , Xu C, Wang M, Ma Y, Huang Z, Liu J, et al. Blockchain-enabled Dispersed Computing Paradigm in Web 3.0 Metaverse[J]. （JCR 一区，中科院二区，IF: 5.6，小改）.

会议论文

1. **Wu Zhonghui** , Xu C, Wang M, Ma Y, Wu Z W, Xiaohou Z, et alPatronus:Countering Model Poisoning Attacks in Edge Distributed DNN Training. // 2024 IEEE Wireless Communications and Networking Conference (WCNC)[C]. Dubai, United Arab Emirates: 2024. （EI，CCF C）

.

1. **Wu Zhonghui** , Chen X, Xu C, Yang SNesterov Accelerated Gradient-based Algorithm for the Allocation Problem of NFV Service. // 2020 International Conference on Networking and Network Applications (NaNA)[C]. Hainan, China: 2020: 213–220. （EI）.
2. Chen X, Xu C, Wang M, **Wu Zhonghui** , Yang S, Zhong L, et alA universal transcoding and transmission method for livecast with networked multi-agent reinforcement learning. // IEEE INFOCOM 2021-IEEE Conference on Computer Communications[C]. Vancouver, Canada: 2021: 1–10. （CCF A）.
3. Liu J, **Wu Zhonghui** , Li Z, Li M, Yang S, Xiao H, et alDLCCB: A Dynamic Labeling Based Covert Communication Method on Blockchain. // 2023 International Wireless Communications and Mobile Computing (IWCMC)[C]. Marrakesh, Morocco: 2023: 168–173. （EI）.
4. Jia Y, Xu C, **Wu Zhonghui** , Feng Z, Chen Y, Yang SMeasuring decentralization in emerging public blockchains. // 2022 International Wireless Communications and Mobile Computing (IWCMC)[C]. Dubrovnik, Croatia: 2022: 137–141. （EI）.

专利

1. 许长桥, 吴忠辉, 陈雅馨, 贾永璞, 严炜琦, 杨树杰. 对等网络的通信方法及系统 [P]. 中国: CN113556286B, 2021.10.26.
2. 许长桥, 季翔, 杨树杰, 吴忠辉 , 王炜, 陈世武, et al. 对等网络的通信方法及系统 [P]. 中国: CN113572615B, 2021.10.29.