一种基于综合匹配度的边缘计算系统任务调度方法

郑守建[1),3)] 彭晓晖[1),3)] 王一帆[1)] 任祖杰[2)] 高 丰[2)]

[1)](中国科学院计算技术研究所 北京 100190)

[2)](之江实验室 杭州 311122)

[3)](中国科学院大学 北京 100049)

摘 要 边缘计算模式满足数据的实时和低功耗处理需求,是缓解当前网络数据洪流实时处理问题的有效方法之一. 但边缘设备资源的异构与多样性给任务的调度与迁移带来极大的困难与挑战. 目前，边缘计算任务调度研究主要集中在调度算法的设计与仿真，这些算法和模型通常忽略了边缘设备的异构性和边缘任务的多样性，不能使多样化的边缘任务与异构的资源能力深度匹配. 本文针对边缘计算系统资源异构且受限的特性，研究边缘任务与目标设备资源深度匹配的有效方法，提出基于任务资源匹配、负载均衡和任务公平性的综合匹配度评估方法（integrative matching evaluation degree method, IMDE），并设计基于网络流的在线多任务调度算法（IMDE and network flow based online multi-task scheduling algorithm, IMD-FLOW）来验证该方法的有效性. 同时，研究边缘计算的仿真系统，将实际环境中用户、任务和设备等若干实体抽象成多个角色和组件，构建符合边缘环境异构特征的 EdgeSimPy 离散事件仿真平台. 在该平台上的实验结果表明，提出的 IMD-FLOW 调度算法相较于轮询、主资源公平（dominant resource fairness, DRF）、Quincy 等其他算法，至少降低 6.26%的任务响应延迟与 7.53%的网络通信开销，在集群超负荷的情况下，系统失效时间平均延缓 1.24 倍.

关键词 边缘计算；资源异构；设备匹配；任务调度；系统仿真

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2022.00485

An Integrative Matching Degree Based Task Scheduling Method for Edge Computing System

ZHENG Shou-Jian[1),3)] PENG Xiao-Hui[1),3)] WANG Yi-Fan[1)] REN Zu-Jie[2)] GAO Feng[2)]

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

[2)](*Zhejiang Lab, Hangzhou* 311122)

[3)](*University of Chinese Academy of Sciences, Beijing* 100049)

Abstract With the rapid growth of intelligent devices, massive amounts of perception data are generated at the network edge. The cloud computing model for Internet of Things (IoT) systems has brought a lot of problems, including high response latency, high transmission energy consumption, privacy leak, etc. Meanwhile, the growth of computing power in data centers cannot meet the exponentially increasing data volume gradually. The edge computing paradigm can satisfy the demands of real-time and low-power data processing, and it is an effective solution for dealing the data deluge in real time. However, there are significant differences in resource types and performance in edge computing environments. The diversity and heterogeneity of computing resources in edge computing systems brings difficulty and challenges to the

收稿日期：2020-08-31；在线发布日期：2021-07-26. 本课题得到国家自然科学基金(62072434，U19B2024)和之江实验室开放课题(2020KE0AB02)资助. **郑守建**，硕士研究生，中国计算机学会（CCF）学生会员（C2368G），主要研究领域为边缘计算、资源管理. E-mail: zhengshoujian20g@ict.ac.cn. **彭晓晖**（**通信作者**），博士，副研究员，中国计算机学会(CCF)会员（69356M），主要研究领域为分布式计算系统架构与计算模型. Email:pengxiaohui@ict.ac.cn. **王一帆**，博士，助理研究员，中国计算机学会(CCF)会员（55224M），主要研究领域为边缘计算系统、系统性能量化分析. **任祖杰**，博士，副部长，中国计算机学会（CCF）会员（24502M），主要研究领域为边缘计算、智能计算. **高 丰**，博士，研究员，中国计算机学会(CCF)专业会员（30906M），主要研究领域为分布式计算、操作系统.

task scheduling and migration among edge devices. At present, the research on task scheduling in edge computing mainly focuses on the design and simulation of scheduling algorithms. They usually simply consider only one or two computing resources, leading to the mismatch of diverse edge tasks and the heterogeneous resource capabilities. To address this problem, we propose effective mechanisms for matching edge tasks and the resources of target devices deeply based on an integrative matching evaluation degree method (IMDE) which includes task and resource matching degree, device load balance degree and task fairness. This method analyzes the correlation between edge tasks and computing devices from multiple perspectives, and finally uses the integrative matching degree to represent the relevance of each task and the target device at the current moment of edge system. Then, in order to verify the effectiveness of this method, we design and develops an online multi-task scheduling algorithm based on IMDE and network flow (IMD-FLOW), which aims to maximize the matching degree of the decision set. This algorithm maps the integrative matching degree between tasks and devices to the network flow graph, assigns appropriate weights and capacities to the edges in the graph, uses the minimum cost flow algorithm which can solve the global optimal problem to obtain the initial scheduling decision, detects conflicts and extracts the final scheduling decisions. In addition, according to task's data requirements and device's network communication capabilities, we construct a fine-grain network communication graph to describe the network environment and data distribution in a simulated edge computing system, and proposes a bandwidth allocation algorithm, with the goal of minimizing data transfer time, to allocate the device bandwidth for one migrated task optimally. We also design a simulation system, named EdgeSimPy, for edge computing includes entities of users, devices, and tasks. It does not limit the kinds of computing resources, and supports distributed data storage to simulates actual edge computing systems. Experimental results on this platform show that IMD-FLOW reduces the task response delay by at least 6.26% and the network communication overhead by at least 7.53% compared with round-robin, random, dominant resource fairness (DRF), Quincy algorithms, and the online algorithm for the multi-component application placement problem (MCAPP-IM). The system failure time is delayed by 1.24 times in average when the edge cluster is overload.

Keywords edge computing; resource heterogeneity; task and device matching; task scheduling; system simulation

1 引 言

随着联网的智能物端设备快速增多，面向物理世界的边缘网络产生了海量的感知数据[1]. 在物联网的云计算模式中，边缘设备（如智能硬件、智能手机或个人 PC 等）将收集的感知数据全部传输至云中心处理. 这种计算模式带来了严重的响应延迟、传输能耗和隐私泄露等问题[2]，数据中心的算力增长速度逐渐无法满足指数级增长的数据实时处理需求，边缘计算模式的提出为解决此类问题提供了一个可选的方案[2-5]. 边缘计算旨在利用分布在网络边缘海量设备的算力，将计算任务调度迁移至靠近数据源的位置执行，与云计算模式互补，可有效缓解物联网的数据洪流问题.

在边缘计算系统中，任务调度与迁移在很大程度上影响了系统的计算能效，因此受到各界的广泛关注. 例如，在学术界中，Chen 等人提出应用于边缘环境的分布式智能视频监控（distributed intelligent video surveillance, DIVS）系统，将卷积神经网络（convolutional neural network，CNN）中隐含层与输出层中的计算任务分割成若干子任务，并分别调度迁移至合适的边缘设备执行[6]，保证了神经网络在边缘的高效执行. 在工业界中，Kubernetes 是目前应用最普遍的容器编排系统，被广泛应用在云数据中心、边缘云、边云协同等计算场景，并被雅虎、网易、华为和京东等多家公司部署在生产环境中. 该系统最主要的功能是完成容器任务的调度与集群资源管理，通过高效的调度算法与自动化部

署减轻了系统运维的压力，有效提升了整个系统的计算效率[7-8].

复杂的边缘环境下，高效的任务调度是边缘计算系统亟待突破的学术问题之一. 边缘系统中设备的计算、存储、网络等资源相对于云中心是有限的，且资源的种类与性能存在显著差异. 例如，处理器有CPU、GPU、NPU 等不同种类，同类处理器间的核数、缓存大小等也存在较大差异[9]. 同时，为提升执行效率，一个边缘任务可能会请求多种加速资源，且执行所需数据可能分布在多台设备上. 因此，设备资源异构且受限、任务多样性和数据分布存储等特点给边缘任务调度带来了很大的挑战.

近年来，大量的边缘任务调度研究工作将该问题建模成多任务多目标的优化问题，并使用规则匹配[10-11]、线性规划[12-14]、机器学习[15-16]、图论[17-19]、博弈论[20-24]等方法进行求解. 但是，这些调度策略中的任务模型与系统资源模型通常只涉及 CPU 的频率和通信协议的速率等少数资源的性能[24-25]，而忽略边缘计算系统中异构计算资源和通信媒介的种类与性能差异，不能对多样化的边缘任务与异构资源能力进行深度需求匹配.

针对上述问题，本文提出综合匹配度评估方法（integrative matching degree evaluation, IMDE），并设计边缘计算系统基于网络流的在线多任务调度算法（IMDE and network flow based online multi-task scheduling algorithm, IMD- FLOW）验证评估方法的有效性. 首先，本文从任务与资源匹配程度、设备负载均衡度和任务公平性三方面分析计算任务与边缘设备的相关性，构建细粒度评估矩阵，并对不同角度的矩阵赋予权重，形成综合匹配度评价矩阵，细粒度、深层次地评估计算任务与边缘设备的匹配程度. 其次，以最大化决策集合的匹配度为调度目标，设计在线多任务调度算法 IMD-FLOW，并采用最小费用流算法求解初始决策集合，检测并去除集合中冲突的决策对，提取最终有效的调度决策. 最后，为验证调度策略的有效性，设计开发了面向边缘计算系统的 EdgeSimPy 离散事件仿真平台，并实现了 6 种分别基于规则匹配、线性规划和图论等不同方法的调度算法，对提出的 IMD-FLOW 算法进行仿真实验. 结果表明该算法与主资源公平（dominant resource fairness, DRF）、Quincy[18]等调度算法相比，至少降低 6.26%的任务响应延迟与 7.53%的网络通信开销，在集群超负荷的情况下，系统失效时间平均延缓 1.24 倍.

2　相关工作

任务调度是分布式并行计算领域重点关注的问题之一，一直广受国内外学术和产业界的关注. 近年来，大量的研究工作针对云中心内、云边协同、边缘协同等多种计算场景研究任务调度策略，提出了基于规则匹配、线性规划、机器学习、图论、博弈论等多种方法的解决方案.

一些研究针对特定计算场景提出的调度方法，显著优化计算时延等指标. 例如，Zhang 等人针对视频流的调度问题，实现了基于 Apache Storm 的延迟敏感计算平台 Hetero-Edge，其系统调度策略根据集群任务情况动态调整[11]，使计算延迟显著降低，但其资源模型是针对视频流类特殊应用设计，不适用于多样化的边缘任务. Bahreini 等人将多组件应用放置（multi component application placement problem, MCAPP）抽象为混合整数线性规划问题，并提出一种启发式在线搜索算法（MCAPP-IM），该研究将设备移动性与通信质量添加至资源模型，通过两次匹配计算来调整应用内组件的调度分配决策[14]，使应用整体执行效率得到提升. 这类研究在固定场景中取得了显著的调度效果，但仅针对单一的计算负载或系统环境，难以适用于复杂的边缘环境.

对于广泛存在的边云、边缘协同等分布式计算场景，部分研究工作针对系统中计算、存储、通信、移动性等资源要素中的一个或几个方面展开研究. 例如，在产业界广泛应用的 Hadoop YARN[26]资源管理框架采用了先入先出（first in first out, FIFO）、公平（fairness）①、计算能力（capacity）②等多个基于规则匹配的调度策略，适用于不同业务场景的云数据中心，但该框架仅涉及 CPU 和内存两类资源. Habak 等人针对设备移动性问题构建了 FemtoCloud 边缘计算系统，并提出一种以最大化设备资源利用率为目标的调度算法[12]，使系统吞吐量、资源和带宽利用率等指标得到提升，但其实验过程仅关注 CPU 一种计算资源与端到端的可用带宽，没有考虑设备间通信链路质量情况. Ma 等人提出了一种云加持的边缘计算框架（cloud assisted mobile edge computing, CAME），通过使用云中心、边缘云的计算资源，缓解边缘任务过重、设备算力不足的

① Apache Hadoop 3.2.1–Hadoop: Fair Scheduler, https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/FairScheduler.html

② Apache Hadoop 3.2.1–Hadoop: Capacity Scheduler, https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html

问题，构建以系统计算延迟和成本最小化为目标的优化函数，利用不等式约束的线性性质进行求解[12]，然而该框架的资源模型忽略了设备间的资源种类差异，不能利用设备内专用资源对任务进行加速. Yang 等人提出一种基于贝叶斯优化的任务调度算法，依据任务有向无环图（directed acyclic graph, DAG）对贝叶斯网络进行初始化学习，捕获不同任务之间的依赖关系，并生成任务迁移调度决策[15]，该算法降低任务间通信开销，但没有关注 DAG 子任务在目标设备上的计算开销. Tang 等人为缓解通信资源竞争问题，设计了基于展望理论（prospect theory, PT）的非合作博弈模型，对任务迁移制定分流决策[20]，其资源模型中没有涉及设备算力属性. 这些研究针对分布式环境中某一方面的资源建模，致力于达成提升资源利用率、优化通信开销等目标，但针对的环境大多是同构的，没有关注异构系统中资源的种类和性能差异.

随着计算环境中资源种类的不断增加，越来越多的研究关注异构资源环境. 例如，针对早期云计算的多维资源环境，Ghodsi 等人提出了 DRF 调度策略方法，将用户在服务器分配中占最大比率的资源设为主资源，并采用最大最小公平分配策略，保证所有用户的主资源份额相等[10]，该方法改变了早期基于资源槽的分配方式，提升了系统资源利用率和分配公平性，但却将集群系统看作单台设备，忽视了系统资源的分布式特性（即系统内设备性能差异）. Zhang 等人提出了一种新型资源管理框架 HeteroEdge，将计算资源抽象为工人（worker）模型，每个工人的能力描述符包含能力指标、资源类型等，便于用户配置资源共享信息，并将调度问题转化为多源最短路径问题进行求解[17]，降低计算延迟与设备能耗. 然而该框架提供的资源能力评价指标忽视了处理器的种类和体系结构带来的设备性能差异，例如，仅使用核心数表示 CPU 能力等级，忽略了 CPU 频率、架构等对性能的影响. Isard 等人提出一种基于网络流的调度框架 Quincy，将任务与设备作为连通图的节点，传输开销作为边权，采用最小费用流算法进行调度决策[18]. 该调度框架具有优良的可扩展性，并行考虑所有任务的通信需求，提升了全局公平性. 但该框架忽视了计算开销对调度质量的影响.

综上所述，传统分布式并行计算领域的任务迁移调度研究已十分丰富，具有非常好的理论指导作用，但是由于传统计算环境大多相对同构，在调度时仅考虑 CPU、内存等少数资源的容量与端到端的网络通信质量，忽略了处理器的类型和体系结构等因素对不同计算任务的加速能力差异，以及边缘计算环境下通信协议的速率、带宽、稳定性和功耗等因素对应用服务质量（quality of service, QoS）的影响. 因此，需要细粒度和多维度地分析边缘任务对资源的需求和边缘设备的资源能力，才能使边缘任务与异构资源深度匹配，作出高效的任务调度决策.

3 综合匹配度评估方法

边缘计算系统任务调度的本质是多样化的任务与异构资源的深度匹配问题，以达到优化一个或多个 QoS 指标的目的. 本文通过建立细粒度、深层次的任务与设备相关性评估方法（IMDE），进而提出基于任务与设备深度匹配的调度算法（IMD-FLOW）. 评估方法一方面需要考虑任务与设备在计算、存储、网络等资源方面的相关性，另一方面还应考虑系统稳定性、设备资源利用率、负载均衡度和任务公平性等多个因素，从有利于系统长期运转的角度，根据系统 QoS 要求与用户体验质量（quality of experience, QoE）要求，评估任务与设备的匹配程度. 任务与设备之间不存在绝对相关性，即无法孤立地评估单一任务与单一设备的相关性. IMDE 方法是对当前环境状态下多任务与多设备的相对相关性进行评估，获取任意两者的相对匹配程度，以辅助调度算法在满足 QoS 和 QoE 要求下选取全局最优的匹配.

本节提出的 IMDE 方法包含任务与资源匹配程度、设备负载均衡和任务公平性三个方面，其中，资源匹配度与任务公平性主要从系统和应用两个角度关注任务延迟，设备负载均衡则主要从系统角度关注设备负载压力，三者相互影响，综合考量系统 QoS 与用户 QoE 的要求. 该方法首先构建资源匹配度、负载均衡度与任务公平度三个相关性矩阵，精确评估各任务与各设备在这三方面的相对匹配程度，然后对三个相关性矩阵赋予不同的权重系数，使系统可根据 QoS 和 QoE 需求调整各方面的重要程度，最终形成综合匹配度评价矩阵. IMDE 评估方法的资源模型覆盖了边缘计算场景中任务、设备与系统的多项环境信息，利用该方法可细粒度分析任务与设备的相关性.

3.1 资源匹配度

边缘系统中，计算资源种类丰富、性能差异巨大. 计算任务为取得更好地加速效果往往包含复杂多样的资源需求，大大增加了评估任务与资源相关

性的难度. 任务与设备在资源方面的匹配度与执行时间具有较强的相关性，执行时间是指任务在某设备从开始执行到结束的整体时间，包含数据传输和计算两方面，执行时间越短，则资源匹配度越高. 本文将设备能力与任务需求进行细粒度划分，从计算资源和数据资源两方面分析任务与设备的相关性，并通过预估计算和传输时间表征两方面的匹配程度，最终计算预估执行时间，构建资源匹配度矩阵.

计算资源相关性. 计算资源主要考虑 CPU、GPU、NPU 等处理器资源，内存、硬盘等资源则作为任务迁移的基本条件，当设备剩余资源不能满足任务需求时，该任务无法迁移至该设备，两者匹配度为 0. 当设备剩余资源满足计算任务基本需求时，计算任务在该设备上的计算时间越短，则两者在计算资源方面的相关度越大.

边缘计算场景中包含大量的机器学习类任务，例如人脸识别开门、语音识别交互、火灾检测等，浮点运算指令在这些任务中占有很大比例. 并且，计算任务的计算时间与其所需的浮点运算量（floating-point operations, FLOPs）成正比. 同时，计算任务在执行时并不会占用设备的全部计算资源，占用比例与其最大占用核心数相关. 因此，任务对某类计算资源的需求可由 FLOPs 与任务最大占用核心数共同表示. 由于边缘计算环境中通常具有多种计算资源，且计算任务对不同资源的需求各异. 为此，本文采用 2 个向量 $\boldsymbol{Q}_u$ 和 $\boldsymbol{P}_u$ 分别表示任务 u 在各类计算资源上的 FLOPs 和最大占用核心数，可表示如下：

$$\boldsymbol{Q}_u = [q_{u1}, q_{u2}, ..., q_{uk}] \tag{1}$$

$$\boldsymbol{P}_u = [p_{u1}, p_{u2}, ..., p_{uk}] \tag{2}$$

其中，k 为计算资源种类数量，q_{ui} 表示任务 u 在第 i 类资源上的 FLOPs 需求，p_{ui} 表示任务 u 在第 i 类资源上的最大占用核心数. 当任务不需要该类计算资源时，则对该类资源的 FLOPs 和最大占用核心数为 0.

处理器的频率经常被用作处理器算力的评价标准[24-25]，但其受流水线架构影响很大，不能精确反应设备的计算能力. 而每秒浮点运算次数（floating-point operations per seconds， FLOPS）向下能够表示硬件能力，向上直接承接数据计算，具有屏蔽硬件架构，直接面向应用的优点. 同时，现代处理器通常包含多个计算核心，例如 NVIDIA GPU 通常拥有多个 CUDA 核心. 计算核心是运行任务的直接实体，依据任务最大占用核心数的不同，计算设备核心会被不同程度占用. 因此，与计算任务相对应的，边缘设备处理器的计算能力由单核心 FLOPS 与核心数量共同表示. 采用 2 个向量 $\boldsymbol{C}_v$ 和 $\boldsymbol{E}_v$ 分别表示设备 v 拥有的各类计算资源的单核心 FLOPS 和核心数量，可表示如下：

$$\boldsymbol{C}_v = [c_{v1}, c_{v2}, ..., c_{vk}] \tag{3}$$

$$\boldsymbol{E}_v = [e_{v1}, e_{v2}, ..., e_{vk}] \tag{4}$$

其中，k 为计算资源种类数量，c_{vi} 表示设备 v 上第 i 类资源的 FLOPS，e_{vi} 表示设备 v 上第 i 类资源的核心数量. 当设备不具有该类计算资源时，则对该类资源的 FLOPS 和核心数量为 0. 另外，设备 v 的剩余可用核心数 $\boldsymbol{\Phi}_v$ 可表示为核心数量与执行任务已占用核心总数之差，公式如下：

$$\boldsymbol{\Phi}_v = \boldsymbol{E}_v - \sum_{u' \in \boldsymbol{T}_v} \boldsymbol{P}_{u'} \tag{5}$$

其中，$\boldsymbol{T}_v$ 表示当前系统状态下在设备 v 上执行的任务集合.

以上采用的 FLOPs 和 FLOPS 两个指标分别表示任务的浮点运算量和设备处理器的浮点运算性能. 本文采用任务在目标设备上的预估计算时间作为两者在计算资源方面的相关性指标，并使用任务计算量 FLOPs 与其占用算力之比作为预估计算时间的计算方式，其中占用算力由任务最大占用核心数与设备核心算力 FLOPS 相乘得到. 当设备 v 剩余资源不能满足任务 u 基本要求时，如对于某种计算资源，v 的剩余容量无法满足 u 的最大占用核心数要求，即在当前时刻不适宜将 u 调度至 v，则认为 u 在 v 上的计算时间为无穷大，否则可对计算时间 t_{comp} 进行预估. 任务 u 在设备 v 上的预估计算时间可表示如下：

$$t_{comp}(u,v) = \begin{cases} \boldsymbol{\theta}^T \cdot \dfrac{\boldsymbol{Q}_u}{\boldsymbol{P}_u * \boldsymbol{C}_v}, & \boldsymbol{P}_u \leqslant \boldsymbol{\Phi}_v \\ \infty, & \boldsymbol{P}_u > \boldsymbol{\Phi}_v \end{cases} \tag{6}$$

其中，$\boldsymbol{\theta}$ 为元素全为 1 的向量. 利用式（6）分别计算各任务在各设备上的预估计算时间，获得预估计算时间矩阵 $\boldsymbol{T}_{comp}$. 预估计算时间体现了任务与设备在计算资源方面的相关性，计算时间越短，说明任务对计算资源的需求与设备拥有的计算资源算力相关性越高，反之，相关性越低.

数据资源相关性. 边缘计算场景中，任务执行和所需数据可能不在同一台设备，且任务所需数据很可能分布在多台设备，数据传输是影响任务响应时间的重要因素，同时也是造成能耗的关键因素之一. 本文依据任务的数据需求和系统通信质量两方面，预估数据传输时间，以此判断任务与设备在数据资源方面的相关性.

任务 u 的数据需求 $\boldsymbol{D}_u$ 表示为所需数据在各设备上的分布情况，表示如下：

$$\boldsymbol{D}_u=[d_{u1},d_{u2},...,d_{un}] \tag{7}$$

其中，n 为系统内设备数量，d_{ui} 表示该任务 u 所需数据在第 i 台设备上的存储量，当该值为 0 时，表示该设备没有存储任务所需数据.

边缘计算系统中，设备间网络通信质量存在强异构性，例如蓝牙与 Wi-Fi 等不同通信方式具有不同的峰值带宽和传输功耗，并且设备与网络信号发射源的相对位置也对传输速率有很大影响. 因此，实际网络中设备通信的有效带宽难以达到理论峰值. 本文将实际有效带宽作为系统通信能力的关键要素，以此刻画设备间通信质量. 设备 v 与相邻设备通信能力向量记为 $\boldsymbol{W}_v$，可表示为

$$\boldsymbol{W}_v=[w_{v1},w_{v2},...,w_{vn}] \tag{8}$$

其中，w_{vi} 表示该设备与第 i 台设备的有效带宽. 当两设备在网络上不相邻时，则该值为 0. 设备与自身的有效带宽设为磁盘读写带宽.

依据系统中所有设备的通信能力向量可以构建如图 1 所示的系统网络通信完全图. 详细刻画了系统内两两设备的通信链路质量的实际情况，相比传统仅考虑端到端的通信质量，该图有助于提升任务与资源深度匹配的准确性.

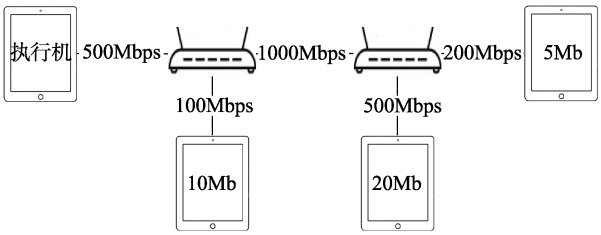

图 1 系统网络通信完全图

利用网络通信完全图可分析两两设备当前可用带宽，结合具体任务数据需求可对数据传输时间进行预估. 当设备资源匮乏，如不能获取到任务所需数据或耗时较长时，则认为该任务在该设备上的数据传输时间为无穷大，否则任务在特定设备上执行时所需的数据传输时间为该设备与所有设备传输耗时中的最大值. 满足条件时，任务 u 在设备 v 上的数据传输时间，可表示为

$$\begin{aligned} t_{comm}(u,v,\boldsymbol{B})&=\max\frac{\boldsymbol{D}_u}{\boldsymbol{B}}\\ &=\max_{v'\in\boldsymbol{M}}\frac{d_{uv'}}{b_{v'}},\boldsymbol{P}_u\leqslant\boldsymbol{\Phi}_v \end{aligned} \tag{9}$$

其中，$\boldsymbol{M}$ 表示系统中所有设备集合. 带宽分配向量 $\boldsymbol{B}$ 表示设备 v 为任务 u 分配的与其他设备通信的上限带宽. $b_{v'}$ 为在设备 v 上执行任务 u 时与设备 v' 的通信带宽，该值满足系统网络通信完全图中链路的带宽限制. 带宽向量具体分配由任务调度算法中的自定义带宽分配算法给出，并保证 $d_{uv'}$ 不为 0 时，$b_{v'}$ 也不为 0. 利用式（9）分别计算各任务在各设备上的预估数据传输时间，可以获得预估传输时间矩阵 $\boldsymbol{T}_{comm}$. 预估传输时间体现了任务与设备在数据资源方面的相关性，数据传输时间越短，说明任务的数据需求与设备的通信能力相关性越高，反之，相关性越低.

资源匹配度. 通过预估计算时间 $\boldsymbol{T}_{comp}$ 与预估数据传输时间 $\boldsymbol{T}_{comm}$，计算两者之和可得预估执行时间 $\boldsymbol{T}_{total}$，该时间是指任务在某设备上执行的整体时间，包含传输与计算两个阶段，执行时间越短，则资源匹配程度越高，体现任务与设备在资源方面的整体相关性. 将 $\boldsymbol{T}_{total}$ 进行规范化，并与 1 做差，得到的规范化矩阵即为资源匹配度矩阵 $\boldsymbol{\mathcal{R}}$：

$$\begin{aligned} \boldsymbol{\mathcal{R}}&=1-\widehat{\boldsymbol{T}_{total}}\\ &=1-\mathrm{Normalize}(\boldsymbol{T}_{comp}+\boldsymbol{T}_{comm}) \end{aligned} \tag{10}$$

矩阵元素 r_{uv} 即表示任务 u 与设备 v 的资源匹配度. 该数值越大，说明任务与设备在资源方面的相关性越高，反之，相关性越低. 当任务不能被迁移至该设备时，例如设备剩余资源不满足任务执行的基本条件，则两者匹配度为 0.

3.2 负载均衡度

边缘设备具有资源受限的特性，若资源分配不合理，容易造成因一种资源过度分配导致设备不可用的情况发生. 为避免系统可用资源产生剧烈波动，本文定义设备负载均衡度来评估任务与设备在负载均衡方面的匹配度.

本文将单一计算资源利用率定义为该资源已占用核心数与总核心数之比，当设备上某类资源不存在或不可访问时，则该类资源的已占用核心数和利用率都设为 0. 当设备单一资源被过分占用，而其他资源有较大空闲时，即资源间利用率相差很大，剩余资源难以被进一步分配，导致设备资源利用不充分. 若使设备内各资源的利用率维持在相同水平，则表示设备资源占用十分均衡，能有效避免设备不可用，有利于提升设备整体资源利用率，增强系统稳定性. 但维持相同水平难以实现，通过表示各利用率的离散程度是衡量均衡情况的有效方式. 设备各资源利用率的离散程度越低，则利用率越相近，设备负载均衡度越高，故设备各资源利用率的离散程度与设备内资源负载均衡度直接相关. 方差是衡

量一组数据离散程度的度量，因此，本文使用资源利用率的方差计算设备内负载均衡度. 例如，已知任务 u 的最大占用核心数 $\boldsymbol{P}_u$，能够预估设备 v 执行任务 u 后的资源利用率方差 $\boldsymbol{\sigma}_{uv}$. 可表示为

$$\boldsymbol{\sigma}_{uv}=\begin{cases}\mathrm{D}\left(\dfrac{\boldsymbol{P}_u+\sum\limits_{u'\in\boldsymbol{T}_v}\boldsymbol{P}_{u'}}{\boldsymbol{E}_v}\right), & \boldsymbol{P}_u\leqslant\boldsymbol{\Phi}_v\\ \infty, & \boldsymbol{P}_u>\boldsymbol{\Phi}_v\end{cases}\tag{10}$$

当 $\boldsymbol{\sigma}_{uv}$ 越大时，设备内资源利用率的方差越大，离散程度越高，资源负载均衡度越低，反之，当 $\boldsymbol{\sigma}_{uv}$ 越小时，资源负载均衡度越高. 利用式（11）分别计算各设备被分配各任务后的资源利用率方差，可得方差矩阵 $\boldsymbol{\Sigma}$，对该矩阵进行规范化，并与 1 做差，最终得到的规范化矩阵即为负载均衡度评估矩阵 $\boldsymbol{\mathcal{L}}$，计算方法可表示如下：

$$\boldsymbol{\mathcal{L}}=1-\hat{\boldsymbol{\Sigma}}\tag{11}$$

矩阵元素 l_{uv} 即表示任务 u 与设备 v 在资源负载均衡方面的匹配度. 设备接受待迁移任务后的资源负载均衡度越高，则该数值越大，反之，数值越小. 当任务不能被迁移至该设备时，两者匹配度为 0.

3.3　任务公平度

资源匹配度与负载均衡匹配度没有关注任务提交的先后顺序对执行效率的影响. 本节以任务提交时间为依据，计算各任务权重作为迁移优先级，构建任务公平度评估矩阵.

将任务提交时间 $\boldsymbol{T}_{submit}$ 规范化，并与 1 做差，得到一组任务权重数据. 依据任务与设备的适配关系 $\boldsymbol{S}$，可构建任务公平度评估矩阵 $\boldsymbol{\mathcal{F}}$，其计算方法如下：

$$\boldsymbol{\mathcal{F}}=\left(1-\widehat{T_{submit}}\right)*\boldsymbol{S}\tag{12}$$

其中，$\boldsymbol{S}$ 为任务与设备的适配矩阵，s_{uv} 为 1 表示设备 v 能够满足任务 u 的资源需求，为 0 则表示不能满足. 矩阵元素 f_{uv} 为 0 时表示任务 u 不能迁移至设备 v，否则表示任务 u 的权重. 任务提交时间越早，越需要被优先迁移，则任务权重值越大，反之，权重值越小. 任务公平度评估矩阵可为筛选长期不被调度的任务提供依据.

3.4　综合匹配度

对上述三个评估矩阵分别赋予不同的权重并相加，形成如下综合匹配度评价矩阵.

$$\boldsymbol{\mathcal{I}}=\alpha\boldsymbol{\mathcal{R}}+\beta\boldsymbol{\mathcal{L}}+\gamma\boldsymbol{\mathcal{F}}\tag{13}$$

其中，$\alpha+\beta+\gamma=1$，α、β、γ 分别是资源匹配度、设备负载均衡和任务公平性三方面的权重系数. 针对不同边缘计算场景与调度目标，调整三者初始化比例以满足 QoS 和 QoE 的具体要求，并在运行时根据负载变化情况进行动态调整，以提升边缘计算系统的运行效率. i_{uv} 表示任务 u 与设备 v 的综合匹配度. 当任务与设备三方面综合相关性越高时，该数值越大，反之，数值越小. 当任务不能被迁移至该设备时，两者匹配度为 0.

综上所述，细粒度的资源模型、灵活调整的权重系数和简洁直观的矩阵表现形式使 IMDE 方法能够有效表征边缘任务与计算设备的深度匹配情况，依据综合匹配度评价矩阵可进一步辅助调度算法设计与决策.

4　基于综合匹配度的多任务调度算法

本节基于综合匹配度评估方法（IMDE）中的网络通信完全图，提出一种最优带宽分配算法. 该算法以最小化数据传输时间为目标，定义带宽分配最优化公式，求解最优带宽分配向量. 同时提出一种基于 IMDE 方法的在线多任务调度算法（IMD-FLOW），结合最优带宽分配算法，计算综合匹配度评价矩阵，以最大化决策集合匹配度为目标构建网络流图， 并采用最小费用流算法提取有效调度决策.

4.1　最优带宽分配算法

边缘计算系统中的网络协议十分异构，带宽分配不合理将急剧增加数据传输时间. 本文综合匹配度评估中的网络通信完全图详细刻画边缘网络通信实际状态，给优化带宽分配带来了可能. 传输时间由任务所需数据的分布情况和设备分配的带宽共同预测，其中带宽的分配则需要由具体的分配算法给出.

首先，依据任务数据需求向量更新网络通信完全图，在当前网络拓扑关系中，确定各设备的数据存储量，如图 1 所示. 然后，以最小化数据传输时间为目标，依据更新后的网络完全图构建带宽分配最优化公式，求解最优带宽分配向量 $\boldsymbol{B}$，计算方式如下：

$$\begin{aligned}&\min_{\boldsymbol{B}}\quad t_{comm}(u,v,\boldsymbol{B})\\&\text{s.t.}\quad b_{v'}\leqslant b_{min}(v,v'),v'\in\boldsymbol{V}\end{aligned}\tag{14}$$

其中，$b_{min}(v,v')$ 表示为设备 v 与设备 v' 通信路径上的最小剩余带宽. 约束条件表示给设备 v 与其他设备分配的带宽应满足两者通信路径上可用带宽上限的约束.

本文设计一种迭代优化的算法求解带宽分配最优化公式，伪代码如算法 1 所示. 该算法依据任务 u

的数据需求 $\boldsymbol{D}_u$，迭代计算迁移设备 v 到其他目标设备 v'中的最大传输时间 t_{max} 及相应路径 $p_{\max}$，然后在已分配带宽矩阵 $\boldsymbol{B}_{alloc}$ 中调整该路径上的带宽分配，以降低该路径上传输时间. 反复迭代优化最大耗时传输路径，逐步逼近最优解，直至优化效果满足指定条件 ε. 最终从 $\boldsymbol{B}_{alloc}$ 中提取设备 v 到其他设备 v' 的带宽向量 $\boldsymbol{B}$.

算法 1. 最优带宽分配算法

输入：任务 u；设备 v；网络通信剩余带宽矩阵 $\boldsymbol{B}_{remain}$；

输出：带宽分配向量 $\boldsymbol{B}$；

1. $\delta \leftarrow 10$ /*步长*/
2. WHILE true
3. $p_{max}, t_{max} \leftarrow findMaxTcommPath(\boldsymbol{B}_{alloc})$
4. IF $0 \leqslant t_{last} - t_{max} < \varepsilon$ THEN /*已经足够逼近最小值*/
5. BREAK
6. END IF
7. IF $t_{max} \geqslant t_{last}$ THEN /*已越过目标值，缩小步长*/
8. $\delta \leftarrow \delta/2$
9. ENDIF
10. $\boldsymbol{B}_{alloc} \leftarrow updateAllocBW(\boldsymbol{B}_{alloc}, p_{max})$
11. $t_{last} \leftarrow t_{max}$ /*更新上轮迭代最大耗时*/
12. END WHILE
13. $\boldsymbol{B} \leftarrow$ extract the bandwidth from v to other machines from $\boldsymbol{B}_{alloc}$

过程 1. $findMaxTcommPath(\boldsymbol{B}_{alloc})$

1. FOR v' IN $\boldsymbol{M}$
2. IF $d_{uv'} = 0$ THEN
3. CONTINUE
4. END IF
5. $p \leftarrow$ find the shortest path between v and v'
6. $b_{min} \leftarrow$find the minimum allocated bandwidth from $\boldsymbol{B}_{alloc}$ along p
7. IF $b_{min} = 0$ THEN /* 未分配带宽 */
8. $t_{curr} \leftarrow$ INF
9. ELSE
10. $t_{curr} \leftarrow d_{uv'} / b_{min}$
11. END IF
12. IF $t_{curr} > t_{max}$ THEN
13. $t_{max} \leftarrow t_{curr}$
14. $p_{max} \leftarrow p$
15. END IF
16. END FOR
17. RETURN t_{max}, p_{max}

过程 2. $updateAllocBW(\boldsymbol{B}_{alloc}, p)$

1. FOR each edge <u',v'> **IN** p
2. IF $\boldsymbol{B}_{alloc}(u',v') + \delta > \boldsymbol{B}_{remain}(u',v')$ THEN /*剩余带宽不满足变化步长时，从相关最短耗时分支抽调带宽*/
3. $\delta' \leftarrow \delta + \boldsymbol{B}_{remain}(u',v') - \boldsymbol{B}_{alloc}(u',v')$
4. $\delta \leftarrow \delta - \delta'$
5. $p_{min} \leftarrow$ find the minimum transfer time path contains <u',v'>
6. $\boldsymbol{B}_{alloc} \leftarrow$ subtract δ' from $\boldsymbol{B}_{alloc}$ from v' to destination along p_{min}
7. END IF
8. $\boldsymbol{B}_{add}(u',v') \leftarrow \delta$
9. END FOR
10. $\boldsymbol{B}_{alloc} \leftarrow \boldsymbol{B}_{alloc} + \boldsymbol{B}_{add}$
11. RETURN $\boldsymbol{B}_{alloc}$

通过带宽最优分配算法求解得到的分配向量保证数据传输时间达到最短，然后通过调整冗余带宽可以进一步减少带宽占用. 最优带宽分配向量中，除最长传输时间所处的路径外，其他路径传输时间都小于或等于最长时间，通过缩减其他路径带宽，使其传输耗时接近最长时间，可以在保证传输时间最短的情况下，占用更少的带宽.

4.2 基于网络流的多任务调度算法

解决边缘计算系统调度问题的关键在于将迁移的任务与目标资源深度匹配，而本文提出的 IMDE 评估方法对待调度任务与设备剩余资源进行了匹配度分析. 本节利用 IMDE 评估方法，从决策整体的角度，将最大化决策集合的匹配度作为调度目标，提出一种基于网络流的在线多任务调度算法. 该算法将任务与设备的综合匹配度关系映射到网络流图中，通过为该图中的边赋予适当的权重和容量，使用最小费用流算法求解全局最优的分配策略.

最小费用流算法在经济学、管理学等领域应用广泛，其目标是获取在最大流量情况下花费最小的解决方案. 边缘计算系统某一时刻调度作业可以编码为如图 2 所示的网络流图. 该图主要包含任务节点、设备节点和虚拟节点等三类节点，每个任务节点 u_i 或设备节点 v_j 分别对应于一个计算任务或边缘设备，虚拟节点有源点 *Src* 和汇点 *Sin*，网络流图中所有流量都从 *Src* 流入，并从 *Sin* 流出. 网络流图的边主要包含容量和成本两个属性. 容量表示该条边最多通过的流量大小，一般为正整数；成本表示单位流量通过该条边所需的开销. 在本文提出的 IMD-FLOW 方法中，边的成本与综合匹配度相关，通过 IMDE 评估方法可计算综合匹配度评价矩阵，

该矩阵中各元素表示当前环境状态下任务与设备的匹配程度，匹配度越高，成本应越低. 因此，本文将 1 与综合匹配度的差作为 u_i 与 v_j 边的成本. *Src* 到 u_i 与 v_j 到 *Sin* 的边仅起到连通作用，所以成本为 0. 由于任务只可被迁移至一台设备，并一次性完成迁移，因此将 *Src* 与 u_i、u_i 与 v_j 的容量都设为 1. 综合匹配度具有相对性，当设备接收新的任务后，与该设备相关的综合匹配度已不具有参考价值，因此一台设备一次调度仅允许迁移一个任务，v_j 与 *Sin* 的容量也设为 1. 使用最小费用流算法对上述构建的网络流图进行求解，可以获取整体匹配度最高，且迁移任务尽可能多的决策集合.

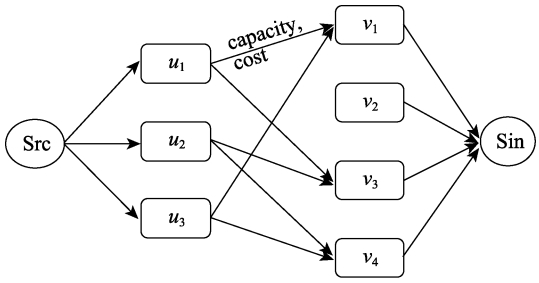

图 2　IMD-FLOW 算法的网络流图

但此时的决策集合中可能包含大量相互冲突的决策对，例如，当一个任务迁移至集群中的某一设备时，系统中网络状态发生改变，先前的综合匹配度评价矩阵中的部分元素失效. 因此，对决策集合进行过滤，依据综合匹配度从大到小的顺序，依次遍历初始决策集合，检测决策对是否失效，并将有效决策对加入最终决策集合.

算法 2. IMD-FLOW 调度算法

输入：待调度任务集合 $\boldsymbol{T}$；可调度设备集合 $\boldsymbol{M}$；网络通信完全图 $\boldsymbol{G}$；

输出：迁移决策集合 $\boldsymbol{\Omega}$；

1. $\boldsymbol{B}_{remain}$←extract the remain bandwidth from $\boldsymbol{G}$
2. FOR each v IN $\boldsymbol{M}$
3. $\quad \boldsymbol{E}_{used} \leftarrow$ get the number of used cores of v
4. $\quad$ FOR each u IN $\boldsymbol{T}$
5. $\quad\quad$ IF v can meet the requirements of u THEN
6. $\quad\quad\quad \boldsymbol{B} \leftarrow allocOptBW(u,v,\boldsymbol{B}_{remain})$ /*利用最优带宽分配算法求解分配向量*/
7. $\quad\quad\quad \boldsymbol{T}_{comp}(u,v) \leftarrow sum(\boldsymbol{Q}_u / (\boldsymbol{P}_u * \boldsymbol{C}_v))$
8. $\quad\quad\quad \boldsymbol{T}_{comm}(u,v)$←get the maximum in the division of each element of $\boldsymbol{D}_u$ and $\boldsymbol{B}$
9. $\quad\quad\quad \boldsymbol{\sigma}(u,v) \leftarrow Var((\boldsymbol{P}_u + \boldsymbol{E}_{used}) / \boldsymbol{E}_v)$
10. $\quad\quad\quad \boldsymbol{W}_{task}(u,v) \leftarrow \boldsymbol{T}_{submit}(u)$
11. $\quad\quad$ ELSE
12. $\quad\quad\quad \boldsymbol{T}_{comp}(u,v), \boldsymbol{T}_{comm}(u,v), \boldsymbol{\sigma}(u,v), \boldsymbol{W}_{task}(u,v) \leftarrow$ INF
13. $\quad\quad$ END IF
14. $\quad$ END FOR
15. END FOR
16. $\mathcal{R} \leftarrow 1 - Normalize(\boldsymbol{T}_{comp} + \boldsymbol{T}_{comm})$
17. $\mathcal{L} \leftarrow 1 - Normalize(\boldsymbol{\sigma})$
18. $\mathcal{F} \leftarrow 1 - Normalize(\boldsymbol{W}_{task})$
19. $\mathcal{I} \leftarrow \alpha\mathcal{R} + \beta\mathcal{L} + \gamma\mathcal{F}$
20. $\boldsymbol{N}$←build a network flow graph based on $\mathcal{I}$
21. $\boldsymbol{S}$←get the initial decisions by min cost flow algorithm on $\boldsymbol{N}$
22. FOR each pair <u,v> from the maximal matching degree to minimal in $\boldsymbol{S}$
23. $\quad$ IF <u,v> not conflict with $\boldsymbol{\Omega}$ THEN
24. $\quad\quad \boldsymbol{\Omega} \leftarrow \boldsymbol{\Omega} \cup$ <u,v>
25. $\quad$ END IF
26. END FOR

算法 2 是 IMD-FLOW algorithm 的伪代码. 首先，从网络通信完全图 $\boldsymbol{G}$ 中提取各设备间剩余带宽 $\boldsymbol{B}_{remain}$. 遍历所有任务与设备组合，按照 IMDE 评估方法计算所有组合的计算时间 $\boldsymbol{T}_{comp}$、通信时间 $\boldsymbol{T}_{comm}$、资源利用率方差 $\boldsymbol{\sigma}$、任务优先权重 $\boldsymbol{W}_{task}$ 等. 对所有结果构成的矩阵进行规范化，得到资源匹配度 $\mathcal{R}$、负载均衡度 $\mathcal{L}$ 和任务公平度 $\mathcal{F}$ 三个相关性矩阵，代入各矩阵权重计算得到综合匹配度评价矩阵 $\mathcal{I}$. 然后，基于该矩阵按照上文描述的构建方法建立网络流图 $\boldsymbol{N}$，并使用最小费用流算法求解初步决策集合 $\boldsymbol{S}$. 最后，依据决策集合中综合匹配度，从大到小依次判断决策对<u,v>是否与最终决策集合 $\boldsymbol{\Omega}$ 冲突，冲突条件包括任务已迁移、设备已分配、带宽冲突等，若不存在冲突，则将决策对加入最终决策集合 $\boldsymbol{\Omega}$，该集合即为算法作出的调度决策.

5　EdgeSimPy 仿真平台

现有大多数仿真平台资源模型粒度较粗，一般仅包含 CPU 频率、内存大小和通信速率三类能力指标，不能满足本文提出的评估方法（IMDE）的仿真实验. 针对边缘环境的异构特性，本节设计并实现了 EdgeSimPy 仿真平台和本文提出的基于 IMDE 方法的网络流在线多任务调度算法（IMD-FLOW），以及多个基准对比调度算法.

5.1　平台架构

EdgeSimPy 是针对边缘计算场景基于 SimPy 仿

真框架[①]使用 Python 开发的离散事件仿真平台，可以与 Python 支持的深度学习框架很好的结合，例如 TensorFlow[27]、PyTorch[28]等，有助于研究基于机器学习或深度学习的任务调度方法.

平台整体架构如图 3 所示，包含 Core 和 Algorithm 两个部分. Core 模块对边缘计算场景任务调度问题中的各个实体进行建模，共抽象出 Scheduler、Broker 等 6 个角色和 Monitor、Algorithm 等 3 个组件. Algorithm 模块用于实现自定义的任务调度决策算法. 本文在该平台上实现了提出的 IMD-FLOW 算法，以及轮询、随机、DRF、MCAPP-IM 和 Quincy 5 个对比基准调度算法.

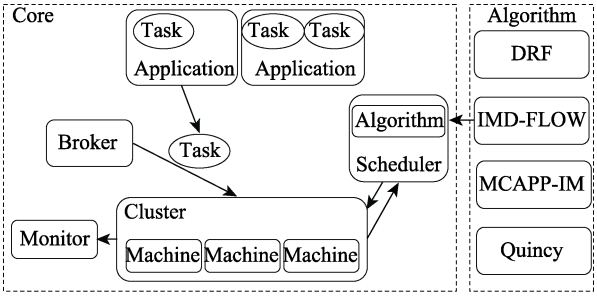

图 3 EdgeSimPy 仿真平台架构图

EdgeSimPy 仿真平台的设计方案如图 4 所示，共包括 9 个过程：

（1）Application 选中需执行的 Task；

（2）Broker 将 Task 提交至 Cluster；

（3）Cluster 调用 Scheduler 决策算法；

（4）Scheduler 作出迁移决策；

（5）Cluster 将 Task 迁移至决策指定 Machine；

（6）Machine 分配计算资源与通信资源，并执行 Task；

（7）Machine 将 Task 结束信息通知 Cluster；

（8）Cluster 将 Task 结束信息通知 Broker；

（9）Broker 将 Task 结束信息通知 Application.

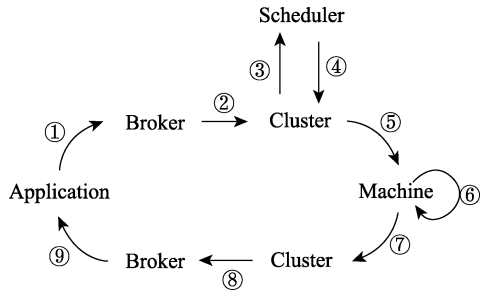

图 4 任务生命周期图

① SimPy: Discrete-Event Simulation for Python, https://simpy.readthedocs. io/en/latest/

5.2 模块设计

Core 模块中各角色与组件的主要职责如表 1 所示. 其中，Machine 是仿真平台中设备的抽象，具有单核心 FLOPS、核心数量等细粒度资源属性，通过设定不同组合的属性配置，可以模拟具有不同计算资源的边缘设备，刻画边缘环境资源异构的特性. Task 是仿真平台迁移调度的任务实体，包含 FLOPs、最大占用核心数、数据分布等资源需求参数，能够表示对不同资源的细粒度需求. 通过所有角色与组件的协作，以及任务与设备细粒度的资源表述，EdgeSimPy 仿真平台突出了边缘异构特征，模拟真实的边缘计算环境.

表 1 Core 模块角色与组件主要职责

角色/组件	主要职责
Application	应用抽象；任务生成；
Task	任务抽象；计算时间、传输时间预估；
Machine	边缘设备抽象；提供计算、通信资源；
Cluster	集群抽象；任务、设备、网络管理；决策执行；
Broker	构建应用；提交任务；
Scheduler	调度器抽象；调度决策；
Algorithm	自定义调度算抽象类，定义调度接口；
Monitor	状态监测；
Simulation	仿真事件管理；

Algorithm 模块包含实验所需的自定义调度算法，具体算法继承 Core 模块中的 Algorithm 抽象类，以满足接口规范.

6 实验与分析

本节首先对实际边缘设备的性能参数和计算任务的资源需求等信息进行测量. 然后，在 EdgeSimPy 仿真平台上，依据实际测量数据，构建边缘计算仿真集群环境和 2 个任务数据集. 最后，对不同调度算法进行实验，并从任务响应延迟、网络通信效率和系统稳定性 3 方面对实验结果进行分析.

6.1 实验环境

为使仿真环境更加贴近实际，本文对 3 台边缘设备的性能参数和多种计算任务的资源需求进行了实际测量，通过实际测量数据构建仿真环境与任务数据集.

本文提取了 Sumsung 550R5L、Nubia Z11 和 OPPO A57 等 3 台设备的详细参数. 它们拥有不同型号和架构的 CPU 处理器，其中 Sumsung 550R5L 拥有 GPU 处理器，使用 LINPACK[29]、NVIDIA Visual

Profiler[①]等工具分别测量三者不同处理器的 FLOPS、核心数量等参数. 此外，依据寒武纪官方数据和实际测量数据向模拟平台添加了包含 NPU 处理器的虚拟边缘设备. 各设备参数规格如表 2.

表 2　设备参数规格表

设备	核心类型	核心型号	单核心 FLOPS	核心数量
Samsung 550R5L	CPU	Intel Core i5-6200U	2.37G	2
	GPU	GeForce 940MX	2.48G	384
Nubia Z11	CPU	Qualcomm Snapdragon 820	1.56G	4
OPPO A57	CPU	Qualcomm Snapdragon 435	180.9M	8
NPU Machine	CPU		2G	4
	NPU	MLU220-M.2[②]	4T	1

本文使用以上实际测量的设备参数在 EdgeSimPy 平台上构建了包含 10 台模拟边缘设备的仿真实验集群，如图 5 所示，其中编号 1~8 的设备为计算设备，能够接受与处理任务，编号 9、10 两台设备为网络设备，不具备数据处理能力. 该集群中包含 CPU、GPU、NPU 等 3 类计算资源，并且 CPU 计算能力存在巨大差异，充分体现边缘环境中资源的异构性特征.

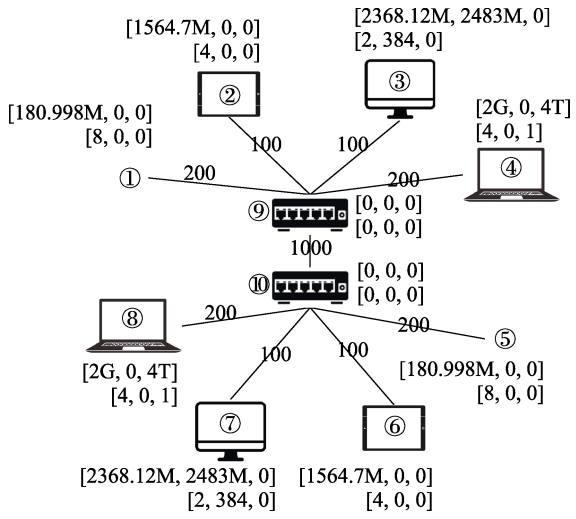

图 5　仿真集群环境

在边缘计算场景中，人工智能类负载占有重要比重[4]，并且该类负载通常需要请求大量异构计算资源，例如，使用 GPU 加速神经网络的训练和推理. 本文使用人脸识别作为主要边缘应用负载，该应用包含人脸检测、特征提取、人脸比较等 3 类计算任务. 任务类型包含 CPU 密集型、GPU 密集型和 CPU/GPU 混合型 3 类. 依据 GPU 密集型任务虚拟构建了 NPU 密集型计算任务的资源需求. 使用 perf[③]、nvprof[④] 等工具测量不同类型计算任务在不同种类计算资源下的计算量 FLOPs、最大占用核心数等资源需求数据. 计算任务的资源需求量随输入数据不同会有较大差异，本文以相同图像输入测量各任务信息，如表 3.

表 3　任务资源需求表

任务	类型	占用核心	FLOPs	最大占用核心数
人脸检测	CPU 密集型	CPU	190.36M	1
CNN 人脸检测	GPU 密集型	CPU	7.69M	1
		GPU	10.12G	192
CNN 人脸检测	NPU 密集型	CPU	7.69M	1
		NPU	10.12G	1
特征提取	CPU 密集型	CPU	192.85M	1
人脸比较	CPU/GPU 混合型	CPU	2.22G	1
		GPU	12.54G	192

本文假设单位时间内提交的应用数量遵循泊松分布（Poisson distribution）. 依据实际测量的任务资源需求数据，构建 2 个仿真数据集，分别为定量数据集与非定量数据集. 定量数据集用于分析提交固定数量任务时系统的任务响应时间、网络通信开销、设备内负载均衡等指标，包含 500 个人脸识别应用负载，每个应用包含 3 个子任务，共 1500 个计算任务. 应用提交的频率服从 $\lambda = 0.5$ 的泊松分布，即平均每 2 秒开始执行 1 个应用，共持续 1000 秒，任务集合中 CPU 密集型、GPU 密集型、NPU 密集型和 CPU/GPU 混合型满足 4:1:1:3 的比例. 在非定量数据集中，增大集群负载压力，使单位时间提交应用的频率服从 $\lambda = 2$ 的泊松分布，即平均每 1 秒随机产生 2 个应用并执行，且不限制运行时间. 通过非定量数据集可以观察不同调度算法在集群高负荷情况下的表现情况.

以上仿真实验的设备集群与数据集完全基于实际测量数据构建，包含边缘环境的资源异构性和任务多样性特征. 数据集提交应用的频率遵循泊松分布，符合实际环境的随机性特征. 同时，采用针对边

① NVIDIA Visual Profiler, https://developer.nvidia.com/nvidia-visual-profiler

②MLU220-M.2, http://www.cambricon.com/index.php?m=content&c=index&a=lists&catid=57

③Perf Wiki, https://perf.wiki.kernel.org/index.php/Main_Page

④ Profiler: CUDA Toolkit Documentation, https://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvprof-overview

缘环境设计实现的 EdgeSimPy 作为仿真平台，使实验数据的合理性与实验环境的真实性得到充分保证.

6.2 结果分析

本文在 EdgeSimPy 仿真平台上使用构建的任务数据集进行对比实验，考虑多种类型的调度算法，包括基于规则匹配的轮询、随机和 DRF 等 3 种调度算法，基于线性规划的 MCAPP-IM 调度算法和基于图论的 Quincy 调度算法. 由于后两种算法原本的资源模型都仅支持一种存在性能差异的计算资源，不能适用于本文针对的异构环境. 因此，将本文采用的资源模型适配到 Quincy 和 MCAPP-IM 算法中，使其利用该资源模型对计算时间和通信数据量等参数进行预估，并采用原本的调度策略进行决策. 本文最终实现以上 5 种调度算法，以及本文提出的基于综合匹配度评估方法（IMDE）的网络流在线多任务调度算法（IMD-FLOW），对这 6 种算法进行了综合对比实验. 其中，计算综合匹配度评估矩阵的参数 α、β、γ 分别设为 0.7、0.2、0.1，目前该参数支持人工初始化配置、运行时调整，以此适应系统环境的动态变化. 接下来将从任务响应延迟、网络通信效率和系统稳定性 3 方面对实验结果进行了深入分析.

任务响应延迟. 任务响应延迟是指其从提交到结束花费的总时间. 该指标是调度问题中最重要的指标之一，对用户体验与系统运行状态有重要影响. 本文使用定量数据集对 4 种调度算法进行仿真实验，首先从任务响应延迟的角度分析实验结果.

当系统应用不同调度算法时，所有任务及各类型任务的平均响应延迟如图 6 所示. 对于所有任务平均响应延迟，本文提出的 IMD-FLOW 算法相较于其他调度算法至少缩短 6.26%. 观察各类型计算任务的平均响应延迟可以看出，本文提出的调度算法对各类计算任务的表现大多都优于其他算法. 特别是对于 CPU 密集型任务，该调度算法的表现更加突出，能够使平均响应延迟降低至少 8.9%.

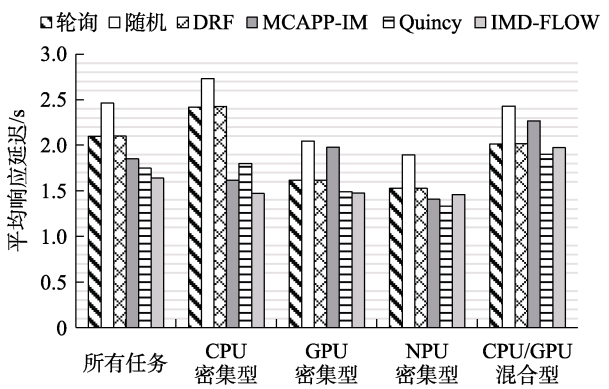

图 6 各类型及所有任务平均响应延迟

这是因为在本文模拟的集群环境中，GPU、NPU 处理性能没有差异，这导致 GPU 或 NPU 密集型任务在各设备上的执行时间大致相同，响应延迟仅受数据传输时延、调度顺序影响. 但各设备 CPU 处理器的性能存在巨大差异，本文提出的 IMDE 方法能够有效感知该类性能差异，判断出边缘任务与异构设备的相关性. 基于 IMDE 的 IMD-FLOW 算法能利用相关性信息作出有效决策，显著降低任务响应延迟. 适配后的 Quincy 和 MCAPP-IM 也能在一定程度感知性能差异，但是 Quincy 仅考虑数据分布异构，MCAPP-IM 忽略负载均衡问题，并且两者都没有关注弱任务长时间不被执行的情况，造成了平均响应延迟高于 IMD-FLOW 算法. 这说明 IMDE 方法能够有效利用环境异构信息，分析设备与任务的深度匹配情况，缓解边缘设备资源异构带来的调度难题，使基于 IMDE 的 IMD-FLOW 算法在面对边缘环境中资源性能差异较大的情况时，能够取得优良的效果.

通过从各个方面分析任务响应延迟可以发现，传统广泛应用的 DRF 调度策略在边缘环境中没有表现出优势，与轮询调度算法表现相当. DRF 调度算法是依据任务主资源占集群总资源的比重对任务排序并调度，这种只关注集群整体能力，不考虑设备异构特性的方式导致资源难以被合理分配，致使任务执行效率不高.

网络通信效率. 边缘环境中网络通信质量通常较差，设备间的有效带宽存在较大差异，这给数据传输管理带来了很大困难. 同时，网络通信在设备能耗中占很大比重，提高网络通信效率有利于提升设备效能.

应用不同调度算法时，系统所有任务的平均数据传输时间和数据传输总量如图 7 所示，结果显示本文提出的 IMD-FLOW 算法在构建的任务数据集上相较于轮询、随机、DRF 和 MCAPP-IM 算法至少降低 7.53%的数据传输时间，减少 6.83%的数据传输量. Quincy 算法以最小化数据传输时延为目标，忽略其他因素对调度的影响，因此其传输时间与传输量低于 IMD-FLOW 算法. 理论上，减少传输量有利于降低设备执行任务时数据传输阶段造成的能耗，降低数据传输时间和任务响应时间. IMD-FLOW 使数据传输时间能够得到有效降低，一方面得益于任务被优先迁移至数据存储量高的设备，这通过数据传输总量的减少得以体现；另一方面得益于最优带宽分配算法对系统环境中通信资源的合理分配. 轮询、随机、DRF 和 Quincy 调度算法在应用不同带

宽分配算法时的平均数据传输时间如图 8 所示，其中平均带宽分配算法是将链路可用带宽平均分配给需要通信的各个设备. 可以看出最优带宽分配算法使各算法的数据传输时间降低 1.23%至 1.49%. MCAPP-IM 算法由于其通信成本设定的限制，不能应用最优带宽分配算法进行优化.

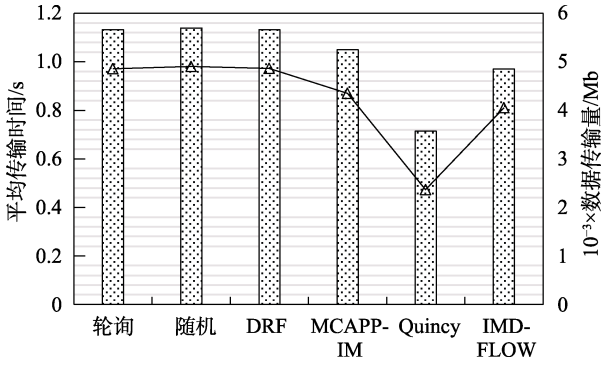

图 7　网络通信效率

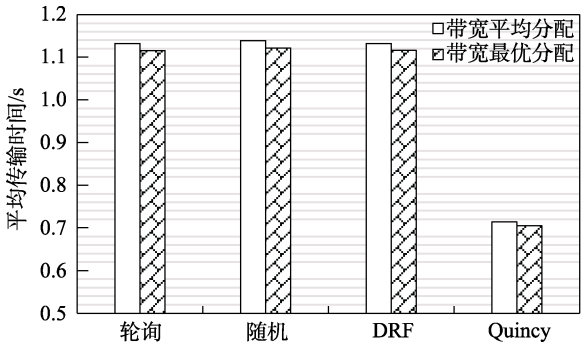

图 8　带宽分配算法比较

通过以上分析可以看出，本文提出的 IMD-FLOW 算法能够降低数据传输时间与网络通信量，有助于解决边缘环境网络异构的问题.

系统稳定性. 边缘设备存在资源受限的特性，容易因单一计算资源过度分配导致设备不可用，造成系统资源可用性波动较大，计算系统稳定性不足的问题. 本文定义的设备内资源负载均衡度是描述资源利用率离散程度的指标，提升负载均衡度可以缓解上述设备不可用问题，并有助于提升资源利用率. 负载均衡度采用利用率的方差作为表征方式，方差越低，则均衡度越高.

系统应用不同调度算法时，所有设备资源利用率方差在各时刻与全过程的平均值如图 9 所示，通过观察散列分布的数据点和全过程的平均值可以看出，本文提出的 IMD-FLOW 在大多数时刻资源利用率的方差低于其他算法，全过程平均方差低约 12.43%. Quincy 资源利用率方差最高，原因在于该算法进行调度决策时主要考虑减少数据传输量，不关心设备负载压力，调度分配具有偏向性，造成了设备负载压力不均衡. 这就表明 IMDE 方法中对负载均衡度的分析是准确有效的，且起到了重要作用，IMD-FLOW 算法优先迁移匹配度高的决策对，有效提升了资源负载均衡度，减少资源分配不均导致设备不可用的情况发生，增强了系统稳定性.

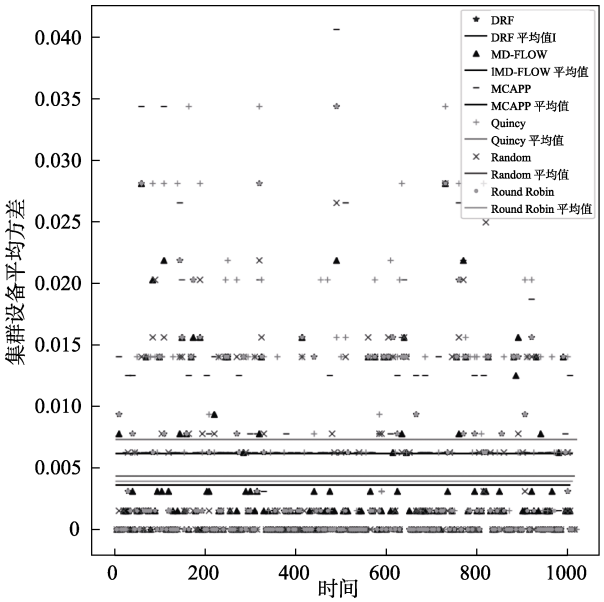

图 9　平均资源利用率方差

当集群环境处于超负荷状态时，更稳定的系统在更短的时间内可以处理更多的任务，因此有效延缓系统拥塞的时间. 本文为评测各调度算法在集群超负荷状态下的系统稳定性，采用非定量数据集对不同调度算法进行比较实验，分析达到不同拥塞程度所需的时间. 本文以待迁移任务数量作为拥塞程度的指标，分为 10、20、50、100 等 4 个级别，分别记录各调度算法到达各拥塞程度的时间，结果如表 4 所示. 其中，由于 MCAPP-IM 算法的计算复杂度太高，没有获取到其达到拥塞程度 100 的时间. 通过分析该表可以看出，本文提出的 IMD-FLOW 算法到达各拥塞程度耗费的时间平均是其他算法的 1.24 倍. 这意味着在集群超负荷时，能够延缓系统拥塞时间，说明该算法使系统稳定性得到了增强. 系统拥塞时间与任务响应延迟、设备负载均衡度都有重要关系，拥塞时间的延缓是一个算法综合性能较强的一个表现.

本章从任务响应延迟、网络通信效率和系统稳定性三个方面对本文提出的 IMD-FLOW 算法进行实验分析. 结果表明，该算法在各个方面都优于其

他调度算法，尤其在针对异构资源时任务响应延迟得到显著降低，说明通过 IMDE 方法能够有效判断多样化的边缘任务与异构资源的深度匹配程度，有助于解决边缘环境的任务调度问题.

表 4 系统不同程度失效时间

算法	10	20	50	100
Random	8.05	14.4	33	60.05
Round Robin	19	30.75	67.05	180
DRF	17	52.05	102.05	199.35
Quincy	10.4	17.0	78.05	193
MCAPP-IM	9.05	26.4	43.05	
IMD-FLOW	29	59.05	106	221

7 总结与展望

本文针对边缘计算系统任务调度问题，提出综合匹配度评估方法（IMDE），从任务与资源匹配程度、设备负载均衡和任务公平性三个角度分别定义任务与设备的相关性评估矩阵，并构建边缘环境网络通信完全图，精确评估边缘环境下计算任务与异构设备的匹配度. 设计了基于细粒度网络通信完全图的最优带宽分配算法，优化边缘环境带宽分配，最小化数据传输时间. 同时，为验证 IMDE 方法的有效性，提出了基于网络流的在线多任务调度算法（IMD-FLOW），以最大化决策集合的综合匹配度总和为目标，采用最小费用流算法获取初始决策集合，并检测冲突，提取有效调度决策. 此外，基于 SimPy 仿真框架设计并实现了 EdgeSimPy 仿真平台，该平台分为 Core 和 Algorithm 2 个模块，Core 模块包含对边缘计算场景任务迁移调度问题中各个实体的抽象，Algorithm 模块用于实现自定义调度决策算法. 最后，在 EdgeSimPy 平台上实现了规则匹配、线性规划和图论等 3 类调度算法，并对提出的 IMD-FLOW 算法进行实验评测. 实验结果表明本文提出的调度算法相较于 DRF、Quincy 等其他算法至少降低 6.26%的任务响应延迟与 7.53%的网络通信开销，针对异构计算资源时，提升效果更加明显,并在集群超负荷的情况下延缓系统失效时间 1.24 倍.

本文主要针对边缘计算系统的异构性特征提出了 IMDE 评估方法，并设计相应算法进行仿真验证. 但该方法在设备能效、调度算法参数选取、任务依赖性、任务预分配等方面存在不足，在实际环境中部署存在通信路径不确定、计算与通信初始化时间占比高、处理器调度不均衡等问题，后续工作计划针对以上挑战在实际环境中开展进一步研究.

参 考 文 献

[1] Reinsel D, Gantz J, Rydning J. Data age 2025: The evolution of data to life-critical. Framingham, USA: International Data Corporation, IDC White Paper, 2017

[2] Shi Weisong, Sun Hui, Cao Jie, et al. Edge computing: An emerging computing model for the internet of everything era. Journal of Computer Research and Development, 2017, 54(5): 907-924 (in Chinese)
(施巍松, 孙辉, 曹杰等. 边缘计算：万物互联时代新型计算模型. 计算机研究与发展, 2017, 54(5): 907-924)

[3] Shi Weisong, Cao Jie, Zhang Quan, et al. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 2016, 3(5): 637-646

[4] Chao Lu, Peng Xiaohui, Xu Zhiwei, et al. Ecosystem of things: Hardware, software, and architecture. Proceedings of the IEEE, 2019, 107(8): 1563-1583

[5] Lin Li, Liao Xiaofei, Jin Hai, et al. Computation offloading toward edge computing. Proceedings of the IEEE, 2019, 107(8): 1584-1607

[6] Chen Jianguo, Li Kenli, Deng Qingying, et al. Distributed deep learning model for intelligent video surveillance systems with edge computing. IEEE Transactions on Industrial Informatics, 2019, 1-1

[7] Zhang Weiguo, Ma Xilin, Zhang Jin-zhong. Research on kubernetes' resource scheduling scheme//Proceedings of the 8th International Conference on Communication and Network Security. Qingdao, China, 2018: 144-148

[8] Song Shengbo, Deng Lelai, Gong Jun, et al. Gaia scheduler: A kubernetes-based scheduler framework//Proceedings of the 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications. Melbourne, Australia, 2018: 252-259

[9] Sanaei Z, Abolfazli S, Gani A, et al. Heterogeneity in mobile cloud computing: Taxonomy and open challenges. IEEE Communications Surveys Tutorials, 2013, 16(1): 369-392

[10] Ghodsi A, Zaharia M, Hindman B, et al. Dominant resource fairness: Fair allocation of multiple resource types//Proceedings of the USENIX Symposium on Networked Systems Design and Implementation. Boston, USA, 2011: 323–336

[11] Zhang W, Li S, Liu L, et al. Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds// Proceedings of the IEEE Conference on Computer Communications. Paris, France, 2019: 1270-1278

[12] Habak K, Ammar M, Harras K A, et al. Femto clouds: Leveraging mobile devices to provide cloud service at the edge//Proceedings of the IEEE International Conference on Cloud Computing. New York, USA, 2015: 9-16

[13] Ma X, Zhang S, Li W, et al. Cost-efficient workload scheduling in cloud assisted mobile edge computing//Proceedings of the IEEE/ACM International Symposium on Quality of Service. Vilanova i la Geltru, Spain, 2017: 1-10

[14] Bahreini T, Grosu D. Efficient placement of multi-component applications in edge computing systems//Proceedings of the 2nd

ACM/IEEE Symposium on Edge Computing. San Jose, USA, 2017: 1-11

[15] Yang Jiadong, Xu Hua, PanLi, et al. Task scheduling using bayesian optimization algorithm for heterogeneous computing environments. Applied Soft Computing, 2011, 11(4): 3297-3310

[16] Wang Yuandou, Liu Hang, Zheng Wanbo, et al. Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning. IEEE Access, 2019, 7: 39974-39982

[17] Zhang D Y, Rashid T, Li X, et al. Heteroedge: Taming the heterogeneity of edge computing system in social sensing// Proceedings of the International Conference on Internationalernet of Things Design and Implementation. Montreal, Canada, 2019: 37-48

[18] Isard M, Prabhakaran V, Currey J, et al. Quincy: Fair scheduling for distributed computing clusters//Proceedings of the 22nd ACM Symposium on Operating Systems Principles. Big Sky, USA, 2009: 261-276

[19] Gog I, Schwarzkopf M, Gleave A, et al. Firmament: Fast, centralized cluster scheduling at scale//Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. Savannah, USA, 2016: 99-115

[20] Tang Ling, He Shibo. Multi-user computation offloading in mobile edge computing: A behavioral perspective. IEEE Network, 2018, 32(1): 48-53

[21] Zhang D, Ma Y,Zhang Y, et al. A real-time and non-cooperative task allocation framework for social sensing applications in edge computing systems//Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium. Porto, Portugal, 2018: 316-326

[22] Zhang D, Ma Y, Zheng C, et al. Cooperative-competitive task allocation in edge computing for delay-sensitive social sensing// Proceedings of the 3rd IEEE/ACM Symposium on Edge Computing. Seattle, USA, 2018: 243-259

[23] Zhang D Y,Wang D. An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems//Proceedings of the IEEE Conference on Computer Communications. Paris, France, 2019: 766-774

[24] Chen Lixing, Zhou Sheng, Xu Jie. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. IEEE/ACM Transactions on Networking, 2018, 26(4): 1619-1632

[25] Chen Xu, Jiao Lei, Li Wenzhong, et al. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ ACM Transactions on Networking, 2016, 24(5): 2795-2808

[26] Vinod K V, Arun C M, Chris D, et al. Apache hadoop yarn: Yet another resource negotiator//Proceedings of the 4th Annual Symposium on Cloud Computing. Santa Clara, USA, 2013: 1-16

[27] Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467, 2016

[28] Paszke A, Gross S, Chintala S, et al. Automatic differentiation in pytorch//Proceedings of the 2017 Neural Information Processing Systems Workshop. Los Angeles, USA, 2017, 1-4

[29] Dongarra J J, Luszczek P, Petitet A. The linpack benchmark: Past, present and future. Concurrency and Computation: practice and experience, 2003, 15(9):803-820

ZHENG Shou-Jian, M.S. candidate. His main research interests include edge computing and resource management.

PENG Xiao-Hui, Ph.D., associate professor. His main research interests include computing architecture and models of distributed computing systems.

WANG Yi-Fan, Ph.D., assistant professor. His main research interests include edge computing systems and system performance evaluation and modeling.

REN Zu-Jie, Ph. D., vice minister. His main research interests include edge computing and intelligent computing.

GAO Feng, Ph. D., professor. His main research interests include distributed computing and operating system.

Background

Task scheduling is a fundamental problem of edge computing. There are a lot of heterogeneous resources in the edge computing environment, as well as the complex and various resource requirement in edge tasks. Efficient task scheduling algorithm can suitably match of task and device to improve the efficiency of the execution. But the task scheduling is a NP-hard problem and difficult to obtain the optimal solution. Thus, the most of researches focus on the approximate optimal solution.

Recently, there are a large number of researches on this scheduling problem, which adopt various methods including rule matching, machine learning, graph theory, game theory to model and solve it. These algorithms aim to reduce computing delay, communication overhead and other indicators but typically focus on a few resources and ignore performance differences between devices. To solve this problem, we propose an integrative matching degree evaluation method (IMDE) that evaluates the relevance between tasks and devices from three perspectives of resource matching degree, device load balance degree and task fairness. And we design an online multi-task scheduling algorithm based on IMDE and network flow, mapping the matching relationship between task and device to the network flow graph, aiming at maximizing the matching degree of decision set, and using the minimum cost flow algorithm to solve the problem. On the three evaluation aspects of the task response delay, the network communication efficiency and the system stability, IMD-FLOW performs better than the other 5 state-of-the-art algorithms.

The research is supported by the Nation Natural Science Foundation of China (No. 62072434 and No. U19B2024) and Zhejiang Lab (No. 2020KE0AB02).