DOI:10.20033/j.1003-7241.(2025)05-0028-06.

存在依赖关系的边缘计算多任务调度策略研究

王晓辉[1]，张 颉[2]，季知祥[1]

（1.中国电力科学研究院有限公司，北京 100192；

2.国网四川省电力公司电力科学研究院，四川 成都 610041）

摘 要：在边缘计算场景中，提取资源的异构性特点，满足个性化需求给任务调度带来极大挑战。对于存在依赖关系的多任务调度，顺序安排不合理会导致一些任务等待时间过长。针对上述问题，提出一种异构边缘算力网络具有依赖关系的多任务调度方法，将待调度任务之间的依赖关系用有向无环图表示，以最小化任务执行总时间开销为目标，基于任务的资源需求与边缘计算设备的算力匹配情况对任务进行调度，有效减少了任务计算响应时间并提升了计算设备利用率。实验结果表明，所研究方法在不同场景下在任务调度延迟方面具有明显优势。

关键词：边缘计算；资源异构；多任务调度；系统仿真；调度

中图分类号：TP18；TP391 文献标识码：A 文章编号：1003−7241(2025)05−0028−06

Research on Edge Computing Multi Task Scheduling Strategy with Dependency

WANG Xiao-hui[1], ZHANG Jie[2], JI Zhi-xiang[1]

(1. China Electric Power Research Institute, Beijing 100192 China;

2. Sichuan Power Research Institute SGCC, Chengdu 610041 China)

Abstract: In the edge computing scenario, extracting the heterogeneous characteristics of resources and meeting personalized requirements bring great challenges to task scheduling. For multi task scheduling with dependencies, unreasonable sequencing can lead to long waiting times for some tasks. To solve the above problems, this paper proposes a multi task scheduling method with dependency relationship in heterogeneous edge computing network. The dependency relationship between the tasks to be scheduled is represented by a directed acyclic graph. With the goal of minimizing the total time cost of task execution, tasks are scheduled based on the matching between the resource requirements of tasks and the computing power of edge computing devices, which effectively reduces the response time of task computing and improves the utilization of computing devices. Experimental results show that the proposed method has significant advantages in task scheduling delays in different scenarios.

Keywords: edge computing; resource heterogeneity; multi-task scheduling; system simulation; dispatch

0 引言

移动互联网特别是5G技术快速发展，越来越多的智能设备构成了边缘网络[1]。许多对时延敏感的服务诸如电力行业、自动驾驶、交互式游戏等行业需要安全可靠、低延时的计算服务。为满足上述新的需求，传统的云计算难以胜任，需要计算资源更加接近用户，从传统的云服务器中心向边缘端转移，以最小化时间延迟。边缘计算(edge computing，EC)技术是一种支持移动设备实现实时响应的新兴计算技术，将算力资源从中心服务器转移到边缘网络，为移动设备提供更加高效的计算服务。边缘计算充分利用了分布在不同地方的边缘设备，将计算任务调度到紧邻的计算设备进行执行，一方面降低了云服务器的负载压力，另一方面也提高了任务处理时间，有利于数据安全。

随着移动端应用的快速普及，设备产生的数据规模越来越庞大，给边缘计算网络带来了越来越大的负担，也对算力资源提出了更高的要求。在边缘计算网络中，任务执行的时间开销主要取决于两个方面，一方面是数据传输的时间开销，另外一方面是任务在边缘计算设备的执行时间开销。为了使计算时延最小化，需要在移动设备与边缘计算设备之间共同分配计算资源，最近有不少针对边缘计算场景的任务卸载及计算资源分配方面的研究工作。

为了权衡能耗和计算延迟两个指标，文献[2]提出了一种基于深度强化学习的多用户边缘计算任务卸载调度

*基金项目：国家电网有限公司科技项目（5700-202155260A-0-0-00）

收稿日期：2023-11-09

方法，以最小化系统延迟和能耗。文献[3]提出一种边缘原生任务调度方法，以促进高效的任务调度及优化边缘原生应用的性能。文献[4]综合考虑了任务之间的三种约束关系，并提出了一种基于差分进化思想的启发式任务调度方法，实现时间更短、收敛速度更快的任务排序。为了提高云边协作的性能，文献[5]提出了一种新型的两阶段调度方法，利用最大流量的思想将任务划分为云−边缘部署方案，根据网络拓扑结构，利用动态编程为边缘域部署运营商。针对包含异构计算资源的边缘计算场景，文献[6]提出了一种基于修正蒙特卡洛树搜索的新型作业调度算法，以充分利用计算资源的优势。文献[7]提出了一种异构边缘系统上机器学习任务的公平调度方法，在提高任务按时完成率的同时考虑能源约束。文献[8]提出一个异构边缘平台上的任务划分和协调框架，用于计算机视觉应用在考虑到 CPU 和GPU 的异构边缘计算平台上的分区和协调。考虑到边缘节点计算资源和通信带宽的有限性，一些研究提出了解决资源分配和任务卸载的方法，旨在降低时延和设备能量消耗。文献[9]提出了一种基于动态优先级队列的卫星任务时间优化调度算法，能够缩短任务响应延迟，实现卫星集群计算资源的高效利用。文献[10]提出了一种基于决斗双深层递归Q−网络方法的多代理任务调度算法，以近似求得最佳任务调度和资源分配方案。

目前已有研究通常缺乏对异构边缘算力网络具有依赖关系的多任务调度问题的研究。针对上述问题，本文提出了一种异构边缘算力网络具有依赖关系的多任务调度方法，将待调度任务之间的依赖关系用有向无环图表示，对异构边缘算力网络中多个任务之间的依赖关系对任务进行排序，以最小化任务执行总时间开销为目标，基于任务的资源需求与边缘计算设备的算力匹配情况对任务进行调度，有效减少了任务计算响应时间并提升了计算设备的计算及存储资源利用率。

1 系统框架及问题建模

不同于一般的边缘算力网络计算资源调度不同，异构边缘算力网络中计算设备存在异构性特征，不同的计算任务对计算资源有着不同的需求，在任务调度时需要考虑任务的个性化需求[11]。与一般的异构边缘算力网络计算资源调度不同，在任务之间存在依赖关系的情况下，进行任务调度的时候不仅需要考虑不同任务对于计算资源的满足情况，同时需要考虑任务间的依赖关系。针对具有依赖关系的边缘计算多任务调度，考虑将待调度任务之间的依赖关系用有向无环图表示，根据任务的前驱任务完成情况决定当前任务能否被调度[12]。根据边缘设备的空闲情况、边缘计算网络拓扑情况以及任务对计算资源的需求情况优化任务分配，并在调度过程中动态更新待调度任务有向无环图，直至所有任务都被调度完成。

1.1 系统框架

图 1 是本文提出的存在依赖关系的边缘计算多任务调度框架，主要包括算力资源动态感知、算力资源调度、算力资源网络管理等几个模块。算力资源动态感知模块负责对当前边缘算力网络的算力资源使用情况进行管理，包括算力可用数量、算力资源当前消耗情况统计等功能。算力资源调度模块包括计算任务排序、计算任务调度等功能。算力资源网络管理模块包括算力路由控制、算力资源网络拓扑结构管理等功能。底层为边缘算力节点情况，本文研究的场景中边缘算力节点为异构算力资源，不同的节点的算力资源存在差异情况。不同计算任务可以在边缘终端节点发起，具体的任务调度需要考虑任务的优先级，并根据距离任务发起节点距离远近、边缘节点算力情况及网络拓扑由算力资源调度模块进行任务调度。

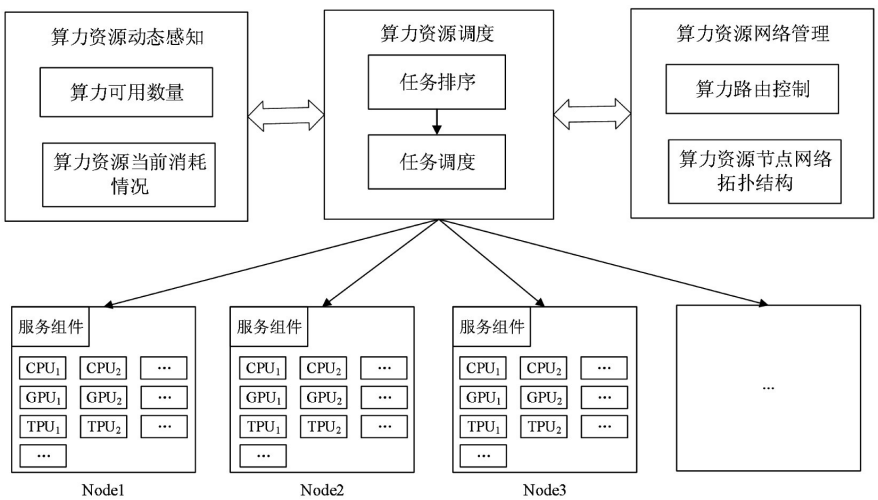

图 1 异构边缘算力网络调度框架

1.2 多任务依赖关系模型

在异构边缘计算环境中，多个计算任务之间可能存在依赖关系，一些任务的执行需要其他任务完成后才能开始。

定义1：对任一任务u来说，如果任务u在任务u'执行完成后才能开始，则称任务u'为任务u的前驱任务。

定义2：对任一任务u来说，如果任务u''在任务u执行完成后才能开始，则称任务u为任务u''的后继任务。

对异构边缘计算环境任务u，其前驱任务集合定义如下：

$$A_u=\left[u_{a1},u_{a2},...,u_{am}\right] \tag{1}$$

式中，u_{ai}为任务u的一个前驱任务，$i\in[1,m]$，m为任务u前驱任务的数量。

对异构边缘计算环境任务u，其后继任务集合定义如下：

$$B_u=\left[u_{b1},u_{b2},...,u_{bn}\right] \tag{2}$$

式中，u_{bi}为任务u的一个后继任务，$i\in[1,n]$，n为任务u后继任务的数量。

异构边缘计算环境多个任务之间的依赖关系用有向无环图(DAG)表示。图2是存在依赖关系的多个任务之间的依赖关系示意图。

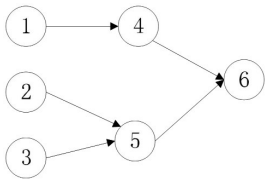

图2 存在依赖关系的多个任务依赖关系示意图

图2中任务4的前驱任务集合为A_4=[1]，任务4的后继任务集合为B_4=[6]。任务5的前驱任务集合为A_5=[2,3]，任务5的后继任务集合为B_5=[6]。对于任务1、2、3来说，由于均不存在前驱任务，因此$A_1=\varnothing$，$A_2=\varnothing$，$A_3=\varnothing$。

2 算法设计

异构边缘算力网络中多个任务之间存在依赖关系的情况下，在进行任务调度时，首先对边缘算力网络中的设备根据算力资源空闲情况进行排序，其次对计算任务根据任务之间的依赖关系进行排序，最后确定任务的调度策略，即确定每一个任务调度到哪个边缘计算设备。

2.1 边缘计算网络节点排序

异构边缘算力网络的计算资源涉及CPU、GPU等异构的硬件设备，在进行任务调度时，要考虑每个计算设备的CPU、GPU等计算资源是否满足任务需求，如果满足任务需求，可以将任务调度到该计算设备上，否则无法将任务调度到该设备上。本文将任务对单运算核心浮点运算量(floating-point operations，FLOP)作为运算量需求度量指标。

用矩阵C_u表示任务u对不同类型计算资源浮点运算量的需求矩阵，具体如下：

$$C_u=\left[c_{u1},c_{u2},...,c_{uk}\right] \tag{3}$$

式中，k为异构处理器种类数量，c_{uj}为任务u在第j类计算资源上的浮点运算次数(floating point operations，FLOPs)需求。如果任务u不需要第j类资源，则c_{uj}值为0。

对每个边缘计算设备，其性能由设备上的各类处理器的FLOPs决定，用矩阵F_h表示每个边缘计算设备各类计算资源的FLOPs矩阵。

$$F_h=\left[f_{h1},f_{h2},...,f_{hk}\right] \tag{4}$$

式中，k为计算设备的各类计算资源的数量，f_{hj}代表计算设备h上第j类计算资源的FLOPs。如果计算设备h上没有第j类计算资源，则f_{hj}值为0。

用F_h'表示计算设备h上当前空闲计算资源的FLOPs矩阵，用H表示异构边缘计算设备集合。基于F_h'对H中异构边缘计算设备按照可用算力资源从大到小排序，得到排序后的异构边缘计算设备集合H'。

2.2 待调度任务排序

异构边缘算力网络中多个任务之间存在依赖关系的情况下，在任务调度之前需要进行排序，根据排序后的任务队列进行调度。由于待调度任务存在依赖关系，这种依赖关系可以用有向无环图表示，因此待调度任务排序问题可以划归为有向无环图的拓扑排序问题，具体排序算法如算法1所示。

算法1. TaskSorting.

输入：任务依赖关系拓扑图G=(V,E)

输出：排序后的待调度任务队列Q

1. Q←Φ；
2. G'←Φ；
3. for each v∈V do
4. if indegree(v)==0
5. Q←Q∪{v}；
6. G'←G去掉定点v及与v相连的所有边；
7. end for；
8. repeat
9. for each v'∈V'do
10. if indegree(v')==0
11. Q←Q∪{v'}；
12. G'去掉定点v'及与v'相连的所有边；
13. end for；
14. until G'==Φ；
15. return Q.

算法1通过对待调度任务进行排序，最终得到排序后的待调度任务集合Q。

2.3 具有依赖关系的多任务调度算法

在计算设备h上完成任务u总的时间开销$t(u,h)$表示如下：

$$t(u,h)=t_p(u,h)+t_s \tag{5}$$

以最小化任务执行总时间开销为目标，本文对任务调度响应时间优化问题建模如下：

$$M_x=min(u,h)\times C_u\times F_h' \tag{6}$$

式中，C_u表示任务u对异构边缘计算设备的算力需求，F_h'表示边缘计算设备h可供任务u使用的计算资源大小。

根据式(5)可知，为最小化完成任务的总时间开销，需要优化任务执行时间。本文考虑基于迭代方法对任务分配进行优化，从而减少数据传输时间。

根据多任务之间的依赖关系，首先调度对其他任务不存在依赖关系的任务，也即是前端任务集合为空集的节点。

存在依赖关系的多任务调度算法如下：

算法2. TasksScheduling.

输入：全部任务集合T，可调度任务集合T'，任务依赖关系拓扑图G，边缘计算设备集合H

输出：任务调度策略Σ

1. T'←Φ；/*初始为空集合*/

2. Q←TaskSorting(G)；

3. H'←DeviceSorting(H)；/*边缘计算设备基于算力资源空闲情况排序*/

4. for each u∈Q do

5. if A_u==Φ

6. T'←T'∪u；

7. end for；

8. for each h∈H'do

9. for each u∈T'do

10. If $C_u<F_h'$

11. If<u,h>与Σ不冲突

12. Σ←Σ∪<u,h>；

13. $F_h'\leftarrow F_h'-C_u$；

14. If $B_u\neq\Phi$

15. for each u'∈B_u do

16. $A_{u'}\leftarrow A_{u'}\cup u'$；

17. end for；

18. $F_h'\leftarrow F_h'-C_u$；

19. end for；

20. end for；

21. for each u∈Q do

22. if $A_u'==A_u$

23. T'← T'∪u；

24. end for；

25. return Σ.

本算法首先根据边缘计算节点算力资源空余情况对边缘计算节点进行排序，然后根据多个任务之间的依赖关系对任务集合进行排序，根据待调度任务的前驱关系确定调度初始可以被调度的任务。对于当前待调度任务，通过任务对计算资源的需求情况与算力节点的可用情况进行比较，优先将任务调度到算力资源富裕的边缘算力节点上，有效减少任务执行中的时间开销。对于每个已调度完成的任务，及时对其后继任务集合中任务的前驱任务集合进行更新，确保任务能及时被调度。

3 仿真实验

3.1 仿真场景及参数设置

论文采用EdgeCloudSim实验平台作为边缘计算仿真工具，模拟智能电网、电力设备管理、智能电表和电力安全监测4个电力异构算力边缘计算环境。不同边缘计算环境具体情况如表1所示。

表1 电力异构算力边缘计算环境

边缘计算环境	任务数量/个	边缘计算设备数量/个	用户数/个
智能电网	1 000	100	100
电力设备管理	500	50	50
智能电表	2 000	200	200
电力安全监测	1 000	100	50

在模拟的上述边缘算力网络当中，包含10个终端节点，7个路由节点，8个边缘计算节点。边缘计算节点分别是E1、E2、E3、E4、E5、E6、E7、E8，具体参数规格如表2所示。

表2 设备参数规格表

设备	核心类别	核心FLOPs	核心数量
E1	CPU	2.5 G	4
	GPU	1.41 G	1 590
E2	CPU	3.5 G	6
E3	CPU	2.56 G	4
E4	CPU	3.8 G	4
	GPU	1.84 G	2 560
E5	CPU	2.1 G	6
	GPU	1.73 G	2 560
E6	CPU	3.4 G	8
E7	CPU	2.37 G	4
	GPU	1.5 G	1 536
E8	CPU	1.48 G	2

为评价和分析论文提出的算法性能，采用算法执行的平均时延作为评估标准。边缘算力网络的任务数量用N表示，参与运算的边缘设备数量用K表示，用户数量用U表示。

3.2 仿真结果与分析

实验中，本文提出的调度方法表示为MECD，在不同场景下与考虑负载均衡的随机调度、不考虑负载均衡的

随机调度进行对比分析。

(1) 用户数量对平均时延的影响

异构边缘算力网络中任务处理时延与用户数量的关系如图3所示。由图3可知,随着用户数量的增加,本文提出的MECD调度、考虑负载均衡的随机调度、不考虑负载均衡的随机调度三种调度方法的平均时延都呈现增加趋势,用户数量越多,其处理耗时也不断增加。不难看出,在任务数量固定的情况下,与随机调度方法相比,所提出的MECD方法的平均时延缩短明显。随着任务数量的增加,MECD方法在处理时延方面的优势更加明显。随机调度方法中,考虑负载均衡的方法与不考虑负载均衡的方法相比,在处理时延方面有一定优势。

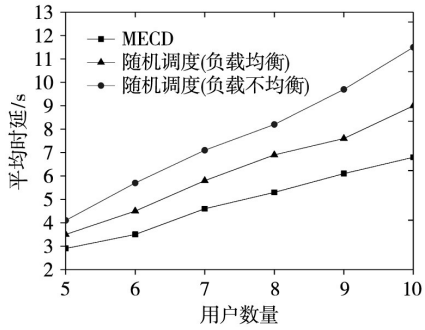

图3 任务处理时延与用户数量的关系

(2) 任务数量对平均时延的影响

异构边缘算力网络中处理时延与任务数量的关系如图4所示。由图4可知,随着任务数量的增加,MECD调度、考虑负载均衡的随机调度、不考虑负载均衡的随机调度三种调度方法的平均时延都呈现增加趋势,主要是因为任务数量的增加使得边缘计算设备平均处理任务数量增加,这直接体现在任务调度的时延上。在用户数量固定的情况下,与随机调度方法相比,所提出的MECD方法的平均时延明显较小。随着任务数量的增加,MECD方法在平均时延方面的优势更加明显。随机调度方法中,考虑负载均衡的方法与不考虑负载均衡的方法相比,在任务数量增加情况下平均时延方面有一定优势。

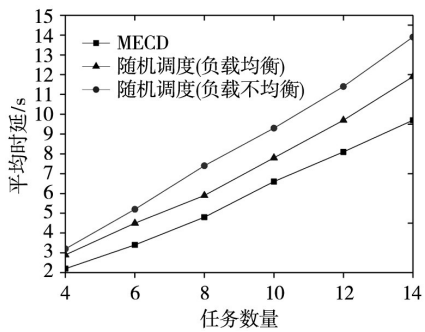

图4 任务处理时延与任务数量的关系

(3) 边缘计算设备数量对平均时延的影响

异构边缘算力网络中处理时延与边缘设备数量的关系如图5所示。由图5可知,随着边缘设备数量的增加,MECD调度、考虑负载均衡的随机调度、不考虑负载均衡的随机调度三种调度方法的平均时延都呈现降低趋势,主要是因为边缘计算设备数量的增加使得计算资源增加,在多任务情况下能够大大减少任务的平均等待时间和运行时间。在用户数量和任务数量固定的情况下,与随机调度方法相比,所提出的MECD方法的平均时延明显较小。随着边缘计算设备数量的增加,MECD方法在平均时延方面的优势更加明显。随机调度方法中,考虑负载均衡的方法与不考虑负载均衡的方法相比,在边缘计算设备数量增加情况下平均时延方面有一定优势。

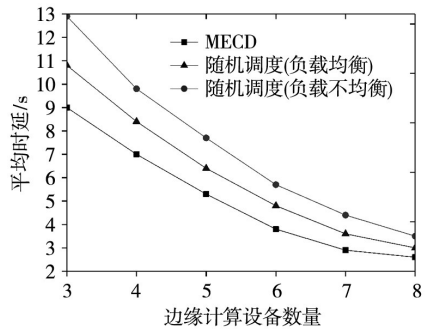

图5 任务处理时延与边缘设备数量的关系

(4) 设备负荷对平均时延的影响

异构边缘算力网络中处理时延与边缘设备负荷的关系如图6所示。由图6可知,随着边缘设备负荷的增加,MECD调度、考虑负载均衡的随机调度、不考虑负载均衡的随机调度三种调度方法的平均时延都呈现增加的趋势,主要是因为边缘计算设备负荷的增加使得可用计算资源减少,这导致任务等待和执行时间增加。在用户数量和任务数量固定的情况下,与随机调度方法相比,所提出的MECD方法的平均时延明显较小。随着边缘计算设备负荷的增加,MECD方法在平均时延方面的优势更加明显。

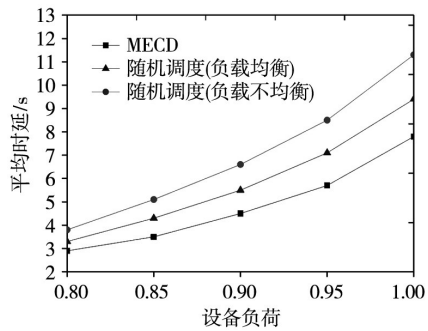

图6 任务处理时延与设备负荷的关系

4 结束语

本文针对异构边缘计算网络环境具有依赖关系的多

(下转第56页)