基于云边协同多任务计算卸载策略

钟云峰,宋伟宁
(东华理工大学 信息工程学院,江西 南昌 330013)

摘 要:在工业互联网中,工厂的设备计算能力有限,边缘计算的出现有效缓解了现场设备的计算压力,提供低时延的计算服务。有效的计算卸载策略能够更好地提供高质量的服务,如今大多数有关计算卸载的研究都是移动边缘计算,移动边缘计算的卸载策略在工业互联网中不适用,因此研究工业互联网中基于边缘计算的计算卸载很有必要。为此,提出了基于云边协同的计算卸载框架以及系统模型;基于此系统模型,以最小化任务时延为目标,将问题形式描述为 01 整数规划问题,并提出了基于混合整数线性规划算法的计算卸载策略解决该问题。实验结果表明,与局部卸载方法和最小时延和能耗卸载方法相比,提出的基于云边协同的计算卸载方法在时延上分别降低了 4% 和 10%,提高了系统性能。

关键词:边缘计算;计算卸载;工业互联网;卸载时延;卸载决策
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2022)04-0069-05
doi:10.3969/j.issn.1673-629X.2022.04.012

Multi-task Computation Offloading Strategy Based on Cloud-side Collaboration

ZHONG Yun-feng, SONG Wei-ning
(School of Information Engineering, East China University of Technology, Nanchang 330013, China)

Abstract: In the industrial Internet, the computing power of the equipment in the factory is limited, and the emergence of edge computing has effectively alleviated the computing pressure of field devices and provided low-latency computing services. Effective computation offloading strategies can better provide high-quality services. Nowadays, most of the research on computation offloading is mobile edge computing. The offloading strategy of mobile edge computing is not applicable in the industrial Internet. Therefore, it is necessary to research the computation offloading based on edge computing in the industrial Internet. To this end, a computation offloading framework and system model based on cloud-side collaboration are proposed. Based on this system model, with the goal of minimizing task delay, the problem form is described as a 01 integer programming problem, and a mixed integer linear programming algorithm is proposed. The computation offloading strategy solves this problem. The experimental results show that compared with the partial offloading method and the minimum delay and energy offloading method, the proposed cloud-side collaborative computation offloading method reduces the delay by 4% and 10%, respectively, and improves the system performance.

Key words: edge computing; computation offloading; industrial Internet; unloading delay; offloading decision-making

0 引 言

近年来,随着互联网、无线接入技术和智能终端技术的发展,极大推动了物联网(Internet of Things, IoT)产业的发展。物联网垂直应用领域的研究也在快速推进,产生了许多的应用场景[1]。随着工业 4.0、“中国制造 2025”的提出,工业互联网成为解决智能工厂中许多问题的主要方案[2]。作为云计算的拓展和补充,边缘计算受到越来越多人的重视[3]。针对由于云计算采用集中式计算造成的网络拥塞以及传输时延较大的问题,边缘计算采用了分布式计算方式,由分布在网络中的多个服务器接受用户的计算任务,从而降低设备上传数据至云服务器的需求,减小了网络拥塞[4]。边缘计算是工业互联中的关键技术,云计算和边缘计算优势互补,共同促进整个工业 IT 和 OT 的深度融合。

目前大多数研究针对的都是移动边缘计算,计算卸载技术解决了用户等待时延高和电池电量快速消耗的问题。叶桓宇[5]利用边缘计算(Edge Computing, EC)和软件定义网络(Software Defined Networks,

收稿日期:2021-04-27 **修回日期**:2021-08-27
基金项目:国家重点研究计划(2018YFB1702700);江西省放射性地学大数据技术工程实验室开放基金(JELRGBDT201903)
作者简介:钟云峰(1997-),男,硕士,研究方向为边缘计算、计算卸载;宋伟宁,博士,讲师,研究方向为智能制造、工业互联网。

SDN)的技术特点,提出了一种新型 SDIN 架构,并在该架构的基础上,设计实现多 QoS 的优先级队列计算卸载算法。Liu Juan 等人[6]根据建立的模型,使用马尔可夫决策过程对每个任务的平均能耗和平均时延进行分析,并提出了一种有效的一维搜索算法,以找到最佳的任务调度策略,降低了任务时延。Mao Y 等人[7]提出了一种基于 Lyapunov 优化的动态计算卸载(LODCO)算法,有效降低了时延。还有 Jia M、Kao Y H[8-9]也提出了以优化时延为目标的在线任务卸载算法。以上是以优化时延为目标的,还有以权衡时延和能耗为目标的研究。例如,代美玲等人[10]以最小化任务时延和设备能耗为目标,把问题描述为资源约束下的最小化能耗和时延加权和的凸优化问题,提出基于乘子法的计算卸载与资源分配解决该问题。孟陈融[11]提出了权衡时延和能耗的任务卸载模型,设计了一种启发式算法来解决该问题,有效降低了任务的时延和设备的能耗。Nan Y、Wang W、Liu L Q[12-14]等人也提出了能耗和时延的权衡优化的卸载方案。周鹏等人[15]提出了一种工业物联网中计算任务跨域卸载模型,将资源分配与计算卸载分为两个子问题,分别得出两个子问题的最优解,最终就是整个优化问题的最优解。

以上大多数研究都是针对移动边缘计算的,在工厂中大多数设备都是有线供电,移动边缘计算的模型和卸载策略不适用。该文设计了一种基于云边协同的工业互联网环境下多任务计算卸载模型和以优化时延为目标的计算卸载策略。首先融合边缘云与远端云构建了一种面向多终端的网络架构,依此建立系统模型;在此模型基础上,构造了以优化卸载时延的目标函数;最后,设计求解算法,求出最优解,合理实施计算卸载。

1 系统模型及问题描述

1.1 系统模型

基于云边协同的计算卸载框架如图 1 所示,由一个远端的云计算中心、多个近处的边缘服务器以及多个工厂中的终端设备组成。在工厂中的边缘服务器不一定只有一个,可以根据终端设备数量或者任务请求量来划分出几个区域,在每个区域放置一个边缘服务器。远端云服务中心与边缘服务中心、边缘服务中心与终端设备分别存在着一对多的映射关系,边缘服务器通过互联网接入云服务中心,终端设备通过有线网或者无线网接入到边缘服务器。终端设备把要执行的任务信息发送到边缘服务器,包括任务需要上传的数据量大小、需要的 CPU 时钟周期数、任务的返回数据量大小等信息,服务器根据任务信息和边缘服务器的资源信息来作出决策,下发至终端。

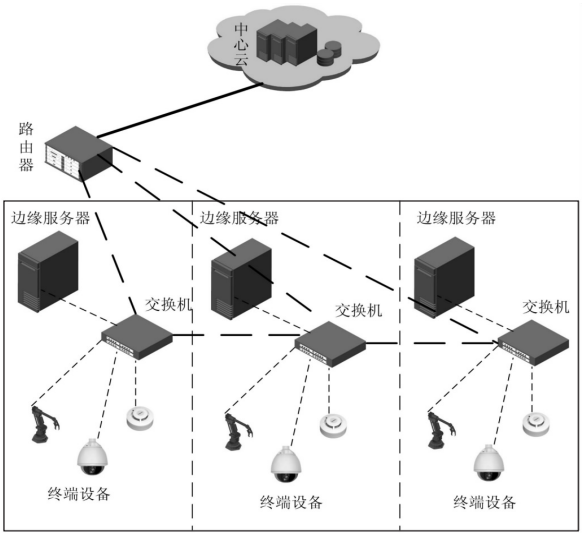

图 1 基于云边协同的计算卸载框架

1.2 时延模型

采用边缘计算的目的就是为了任务能够快速响应,所以能耗优化不是该文考虑的方面,系统优化的目标是时延。系统时延由计算时延和通信时延组成,计算时延有本地计算时延、边缘计算时延、云端计算时延;通信时延有数据上传传输时延和结果返回传输时延。由于一般回传的数据量相对较小,所以该文不考虑结果回传时延。

1.3 终端计算模型

假设一个区域内有 N 个终端设备,表示为 $N=[1,2,\cdots,n]$,其中每个终端都有 M 个任务需要执行,表示为 $M=[1,2,\cdots,m]$。每个计算任务定义为元组 $A_{i,j}=(D_{i,j},W_{i,j},R_{i,j})(i\in N,j\in M)$,$D_{i,j}$ 表示当前任务的数据量,$W_{i,j}$ 表示当前计算任务所需要的 CPU 时钟周期数,$R_{i,j}$ 表示响应数据大小。任务在终端设备上执行的时延为计算时延 t_{local},可表示为:

$$t_{\text{local}}=\frac{W_{i,j}}{C_{\text{local}}} \tag{1}$$

1.4 边缘计算模型

任务卸载到边缘服务器的时延有数据上传时延、计算时延、等待时延,所以任务在边缘服务器上执行的时延 t_{edge} 可表示为:

$$t_{\text{edge}}=\frac{D_{i,j}}{V_{\text{edge}}}+\frac{W_{i,j}}{C_{\text{edge}}}+t_{\text{wait}} \tag{2}$$

其中,t_{wait} 为基于排队论的等待时延预测[16]。

根据排队论的 Little 法则,在平衡条件下,任务在服务器等待的平均时间为系统的平均等待队长除以任务的平均进入率,即:

$$t_{\text{wait}}=\frac{\overline{N_q}}{\bar{\lambda}} \tag{3}$$

其中,$\overline{N_q}$ 为平均等待队长,$\bar{\lambda}$ 为任务的平均进入率。$\overline{N_q}$

与 $\bar{\lambda}$ 为系统上线后,通过系统设计的计数器与定时器进行自动测量而得到,具体测量与评估方法如下:

在时长为 t 的时间段内,等待的任务数为 $N_t - S$,随着时间的增加,计算平均等待队长为:

$$\overline{N_q} = \frac{\sum_{i=0}^{t} N_t - S}{t} \tag{4}$$

其中,N_t 为 t 时刻的全部任务数,S 为边缘服务器同时服务的最大任务数。同时得到任务平均进入率。

$$\bar{\lambda} = \frac{N_t - N_0}{t} \tag{5}$$

其中,N_0 为决策开始时系统内的任务数。由此,可以得到排队等待时间的预测值。

1.5　云端计算模型

任务卸载到云服务中心的时延有数据上传时延和计算时延,所以任务在云服务器上执行的时延 t_{cloud} 可表示为:

$$t_{\text{cloud}} = \frac{D_{i,j}}{V_{\text{edge}}} + \frac{D_{i,j}}{V_{\text{cloud}}} + \frac{W_{i,j}}{C_{\text{cloud}}} \tag{6}$$

1.6　问题描述

一个任务 $A_{i,j}$ 有可能在本地执行或者卸载到边缘服务器、云服务器,设 $x_{i,j}$ 为 1 表示在本地执行,$x_{i,j}$ 为 0 表示卸载执行;$y_{i,j}$ 为 0 表示没有卸载到边缘服务器,$y_{i,j}$ 为 1 表示卸载到边缘服务器;$z_{i,j}$ 为 0 表示没有卸载到云服务器,$z_{i,j}$ 为 1 表示卸载到云服务器。所以任务 $A_{i,j}$ 的执行时延 $t_{i,j}$ 可表示为:

$$t_{i,j} = x_{i,j}\frac{W_{i,j}}{C_{\text{local}}} + y_{i,j}\left(\frac{D_{i,j}}{V_{\text{edge}}} + \frac{W_{i,j}}{C_{\text{edge}}} + t_{\text{wait}}\right) + z_{i,j}\left(\frac{D_{i,j}}{V_{\text{edge}}} + \frac{D_{i,j}}{V_{\text{cloud}}} + \frac{W_{i,j}}{C_{\text{cloud}}}\right) \tag{7}$$

优化的目标是总时延最小即:

$$\min \sum_{i \in N, j \in M} t_{i,j} \tag{8}$$

约束条件为:

$$x_{i,j} + y_{i,j} + z_{i,j} = 1,\ x_{i,j}, y_{i,j}, z_{i,j} \in 0,1 \tag{9}$$

变量标识如表 1 所示。

表 1　变量标识

参数	含义
N	终端设备的数量
M	终端设备上的任务数量
$A_{i,j}$	第 i 个终端设备上的第 j 个任务
$D_{i,j}$	当前任务的数据量
$W_{i,j}$	当前计算任务所需要的 CPU 时钟周期数
$R_{i,j}$	响应数据量大小
t_{local}	本地计算总时延
t_{edge}	边缘计算总时延
t_{cloud}	云端计算总时延
C_{local}	本地设备的计算能力
C_{edge}	边缘服务器的计算能力
C_{cloud}	云服务器的计算能力
t_{wait}	边缘服务器等待处理的排队时延
V_{edge}	本地与边缘服务器节点之间的传输速率
V_{cloud}	边缘服务器节点与云服务器之间的传输速率

2　卸载策略

根据上面描述,问题变成了一个 01 整数规划问题,只需要求得最优解然后按照最优解进行任务卸载。求解 01 规划的方法有很多,一般的有分支定界法、枚举法和模拟退火法,当 M 和 N 比较大时分支定界法和枚举法会出现收敛速度慢的问题。所以该文选取了混合整数线性规划算法来求解上述问题,混合整数线性规划算法具有求解速度快且是精确解的优点。

混合整数线性规划算法求解步骤:

01 整数规划是特殊的整数规划,这类问题被分类为 NP 困难问题。混合整数线性规划算法使用此基本策略来求解混合整数线性规划。混合整数线性规划算法可以在任一阶段完成问题的求解。如果它在某个阶段成功求解了问题,算法不会执行后面的阶段。

(1)使用线性规划预处理缩减问题的规模。预处理步骤旨在消除冗余变量和约束,改善模型的尺度和约束矩阵的稀疏性,加强变量的边界,检测模型的原始和对偶不可行性。

(2)使用线性规划求解初始松弛(非整数)问题。

(3)执行混合整数规划预处理以收紧混合整数问题的 LP 松弛。混合整数规划预处理的主要目标是简化后续的分支定界计算。预处理包括快速预检查和消除一些无用的子问题候选项,以免分支定界算法对其进行分析。

(4)尝试切割生成以进一步收紧混合整数问题的 LP 松弛。

(5)尝试使用启发式方法求得整数可行解。

(6)使用分支定界算法系统地搜索最优解。

最后根据求解出来的最优解来执行卸载策略。

3　仿真结果与分析

为了验证云边协同多任务计算卸载策略的有效性,首先比较文中方法与局部卸载方法的差异,即任务只在本地或者边缘服务器执行,不卸载到云服务中心;再将文中的卸载时延策略与最小时延能耗策略[10]进

行比较;然后研究在用上述两种方法求解的情况下,计算任务总时延和任务数量的关系;最后比较模拟退火算法和混合整数线性规划算法在计算任务的总时延和算法执行时间方面的差异。

仿真实验在 Matlab 环境下进行,系统的各个仿真参数如表 2 所示。表 2 中给出了任务数据量大小、任务所需要的 CPU 时钟周期数、终端设备的计算能力、边缘服务器的计算能力、云服务器的计算能力、终端到边缘服务器的传输速率、边缘服务器到云服务器的传输速率。

表 2 系统的仿真参数设置

参数	仿真值
$D_{i,j}$ /Mbits	[0.5,20]
$W_{i,j}$ / Cycles	[0.1,6.0] 10^9
C_{local} /GHz	2
C_{edge} /GHz	10
C_{cloud} /GHz	100
V_{edge} /Mbps	30
V_{cloud} /Mbps	15

首先,通过设置不同的任务数量,来比较不同卸载策略完成任务的总时延。计算任务总时延与任务数量的关系如图 2 所示。图 2 比较了云边协同多任务计算卸载策略(文中方法)与局部卸载方法完成任务的总时延,从图中可以看出文中方法优于局部卸载方法,相比局部卸载方法时延降低了 4%。所以在实际的部署中,将边缘服务器与云服务器联动起来能够有效地降低任务时延,提高服务质量。

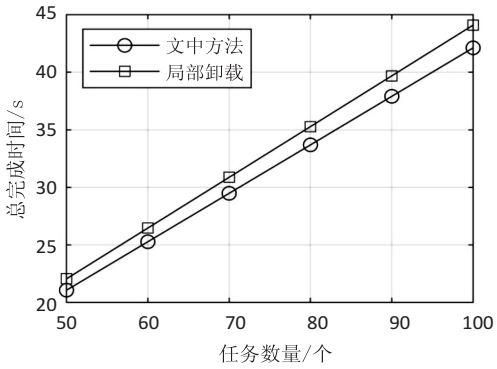

图 2 不同卸载策略下任务完成时间的比较

再将文中卸载策略与最小时延和能耗卸载策略进行比较,比较结果如图 3 所示。从图 3 可以看出,与最小时延和能耗的卸载策略相比,文中的最小时延卸载策略完成任务的时延更低,时延降低了 10%。在时延要求很高的情况下,不考虑能耗的卸载策略能够更好地降低时延。

图 4 为不同任务数量下文中策略分配任务的情况。从图中可以看到任务会卸载到不同的位置,卸载策略会根据任务的数据量大小、计算量大小等信息将任务卸载到最适合的位置来减小时延。随着任务数量的增加,边缘设备的压力会增加,所以会有更多的任务卸载到云端和设备终端。

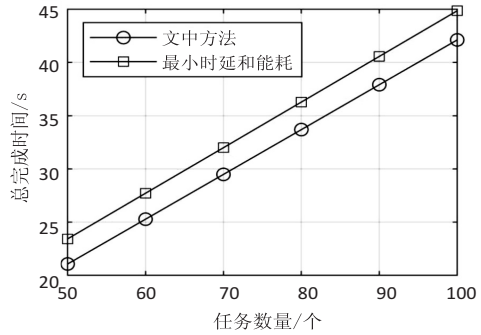

图 3 不同卸载策略下任务完成时间的比较

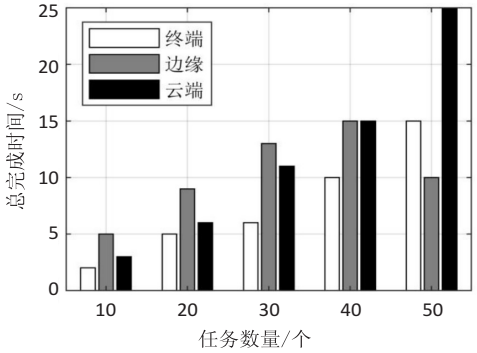

图 4 任务卸载位置分配

模拟退火算法与混合整数线性规划算法的比较如图 5 所示。从图 5 可以看出,两种算法中时延都随着任务数量的增加而增加。混合整数线性规划算法在不同任务数量下的性能都优于模拟退火算法,这是因为混合整数线性规划算法能够求解出 01 整数规划的最优解,而模拟退火算法作为启发式算法,不一定能够求出最优解。

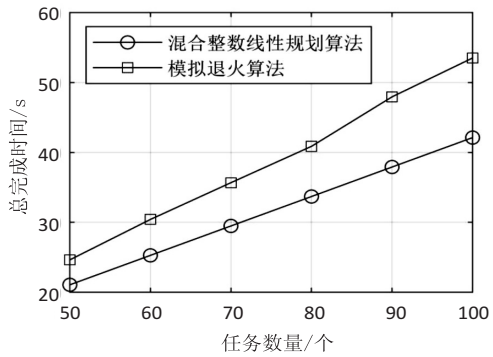

图 5 模拟退火算法与混合整数线性规划算法的比较

当设备数量增加时,假设每台设备有 10 个任务,比较模拟退火算法与混合整数线性规划算法的算法执行时间,如表 3 所示。从表 3 可以看出,模拟退火算法的执行时间随着设备数量的增多而增加,随着设备规

模的增加,混合整数线性规划算法的算法执行时间基本没太大的增加。因此,仿真结果表明混合整数线性规划算法明显优于模拟退火算法。

表 3　模拟退火算法与混合整数线性规划算法执行时间比较

设备规模/台	模拟退火算法/s	混合整数线性规划算法/s
100	2.92	0.02
150	4.07	0.03
200	5.17	0.03
250	6.47	0.05
300	7.33	0.07

4　结束语

研究了基于云边协同的多任务计算卸载问题,设计了基于云边协同的计算卸载框架;基于此系统模型,以最小化任务时延为目标,设计了一种多任务计算卸载方法。针对现实中任务排队时延计算问题,采用了一种基于排队论的等待时延预测方法。针对根据系统模型建立的 01 整数规划问题,提出了混合整数线性规划算法的解法。仿真结果表明,与局部卸载方法和最小时延和能耗卸载方法相比,提出的基于云边协同的计算卸载方法在时延上有所优化,在算法执行时间上也有很大的提升。

参考文献:

[1] 邬贺铨. 物联网技术与应用的新进展[J]. 物联网学报,2017,1(1):1-6.

[2] EMILIANO S, ABUSAYEED S, SONG H, et al. Industrial Internet of things: challenges, opportunities, and directions [J]. IEEE Transactions on Industrial Informatics, 2018, 14 (11):4724-4734.

[3] 李林哲,周佩雷,程　鹏,等. 边缘计算的架构、挑战与应用[J]. 大数据,2019,5(2):1-16.

[4] YU W, LIANG F, HE X, et al. A survey on the edge computing for the Internet of things[J]. IEEE Access, 2017, 6:6900-6919.

[5] 叶桓宇. 基于软件定义工业互联网的边缘计算技术研究[D]. 北京:北京邮电大学,2019.

[6] LIU J, MAO Y, ZHANG J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]// 2016 IEEE international symposium on information theory (ISIT). Barcelona: IEEE, 2016: 1451-1455.

[7] MAO Y, ZHANG J, LETAIEF K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605.

[8] JIA M, CAO J, YANG L. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing[C]//2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS). Toronto, Canada: IEEE, 2014: 352-357.

[9] KAO Y H, KRISHNAMACHARI B, RA M R, et al. Hermes: latency optimal task assignment for resource-constrained mobile computing[C]//2015 IEEE conference on computer communications (INFOCOM). Hong Kong: IEEE, 2015: 1894-1902.

[10] 代美玲,刘周斌,郭少勇,等. 基于终端能耗和系统时延最小化的边缘计算卸载及资源分配机制[J]. 电子与信息学报,2019,41(11):2684-2690.

[11] 孟陈融. 面向边缘计算的数据中心服务资源调度机制研究[D]. 北京:北京邮电大学,2018.

[12] NAN Y, LI W, BAO W, et al. Adaptive energy-aware computation offloading for cloud of things systems[J]. IEEE Access, 2017, 5: 23947-23957.

[13] WANG W, ZHOU W. Computational offloading with delay and capacity constraints in mobile edge[C]//2017 IEEE international conference on communications (ICC). Paris, France: IEEE, 2017: 1-6.

[14] LIU L, CHANG Z, GUO X, et al. Multi-objective optimization for computation offloading in mobile-edge computing [C]//2017 IEEE symposium on computers and communications (ISCC). Heraklion, Greece: IEEE, 2017: 832-837.

[15] 周　鹏,徐金城,杨　博. 工业物联网中基于边缘计算的跨域计算资源分配与任务卸载[J]. 物联网学报,2020,4(2): 96-104.

[16] 刘国强. 基于移动边缘计算的任务卸载策略研究[D]. 哈尔滨:哈尔滨工业大学,2018.