基于计算资源动态可用性的计算卸载切换策略

茶 青 牛 力 侯 鹏 何 敏 李 波*
(云南大学信息学院 云南 昆明 650500)

摘 要 计算卸载和切换是提升车载边缘计算的计算性能和服务质量的关键问题。但由于车辆的移动性,可用计算资源的动态性变化将会导致卸载场景不断变化,进而导致任务完成时间增加。为此,提出基于计算资源动态可用性的计算卸载切换策略,包括在路径固定情况下上传目标服务节点的选择策略和基于动态可用计算资源的切换策略。实验结果表明,相对于现有研究,所提策略能够有效地缩短任务完成时间,提升计算卸载性能。

关键词 车载边缘计算 计算卸载 计算切换 计算切换策略

中图分类号 TP393 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.05.006

COMPUTING OFFLOAD HANDOFF STRATEGY BASED ON DYNAMIC AVAILABILITY OF COMPUTING RESOURCES

Cha Qing Niu Li Hou Peng He Min Li Bo*
(*School of Information Science and Engineering*, *Yunnan University*, *Kunming 650500*, *Yunnan*, *China*)

Abstract Computing offloading and handoff are key issue to improve the computing performance and quality of service of vehicle edge computing. Due to the dynamic changes of vehicle mobility and available computing resources, the offloading scene will continually change, which will increase the task completion time. For this reason, a computing offloading and handoff strategy based on the dynamic availability of computing resources is proposed, including a selection strategy of the upload target service node under the fixed path and a computing handoff strategy based on dynamic available computing resources. The experimental results show that compared with the existing researches, the proposed strategy can effectively shorten the task completion time and improve the computing offloading performance.

Keywords Vehicle edge computing Computing offloading Computing handoff Handoff strategy

0 引 言

随着 5G 网络和物联网(Internet of Thing,IoT)设备的不断升级,车联网(Internet of Vehicles,IoV)成为智能交通领域的重要研究课题,由于车辆自身硬件设施的限制,车辆在道路上行驶时会将其任务请求(任务)发送至提供通信和计算服务的基础设施,基础设施能够与车辆进行通信并处理车辆任务。由于车辆的高速移动性,其任务请求通常只能容忍较短的端到端延迟,较长的时延将严重影响应用服务质量和用户体验。

传统的云计算模式下的计算卸载将计算任务卸载到云计算中心,即车载云计算。但由于云计算的集中性,车载云计算将产生高时延,因此提出在云计算和车辆终端之间构建边缘计算层,将边缘服务器与路侧基础设施结合起来[1]。不仅使得数量众多且广泛分布的车辆方便访问计算资源,而且车辆可以将全部/部分计算任务卸载到移动边缘计算(Mobile Edge Computing,MEC)服务器[2-3]进行计算或存储,同时克服了车辆计算、通信、存储和能源有限的限制和车载云计算的过度延迟问题[4]。因此车载边缘计算(Vehicle Edge Computing,VEC)被视为提高 IoV 计算卸载效率和卸载质量的关键技术[5]。

收稿日期:2020-11-10。国家自然科学基金项目(61562092)。**茶青**,硕士生,主研领域:移动边缘计算。**牛力**,硕士生。**侯鹏**,硕士生。**何敏**,副教授。**李波**,教授。

VEC 环境主要由车辆终端、部署了 MEC 服务器的路侧单元(Road Side Unit,RSU)(该组合可称为服务节点)和部署了云服务器的宏基站组成[6]。当车辆产生的计算任务在本地执行不能满足其要求时,任务将通过无线网络卸载至服务节点或云服务器执行,以缩短计算延迟,快速响应任务请求[7]。

文献[8]考虑车辆的移动轨迹和卸载任务的完成时间,提出了基于路径可预测的计算卸载策略,通过选择满足卸载需求的服务节点,提高计算卸载的效率,但该算法未解决移动性对计算卸载性能的影响,不适用于动态环境下的计算卸载。文献[9]提出了一种考虑了车辆的移动性的计算卸载方案,结果表明,在车载网络中将任务卸载到边缘节点可最大程度地降低相关成本。文献[10]和文献[11]考虑了从车辆终端到服务节点的任务卸载所涉及的延迟,但随着车辆不断移动,服务节点与车辆之间需经过多跳链路进行传输,增加了通信过程(包括上传过程和下载过程)的网络开销。文献[12]深入研究了车辆移动性对卸载性能的影响,提出了基于 Follow-Me Cloud 的动态卸载算法,保证了任务随着车辆终端的移动不断地处理切换过程,更新选择距离卸载任务最近的代理资源。通过不断切换减少了服务节点和车辆的通信距离。文献[13]提出了基于移动性的任务卸载方案,同时考虑了车辆的移动性和计算结果的下载延迟。但上述文献都未考虑边缘节点计算资源的动态变化性,不符合实际的网络环境。

表 1 总结了现有的工作。对现有作品的详细分析表明,尽管现有研究在针对 VEC 环境下的计算卸载和计算切换问题取得了一定的成果,但在卸载过程中要么忽略移动性,要么忽略下载延迟,要么忽略 MEC 服务器分配给每个卸载任务的可用计算能力动态变化对任务卸载服务质量的影响。在 VEC 卸载场景中,车辆的不断运动会造成卸载场景中数据传输环境不断发生改变,同时随着 MEC 服务器上并发处理的计算任务数量的变化会导致 MEC 服务器分配给每个计算任务的可用计算资源动态变化,从而造成了动态变化的 VEC 计算卸载环境[14]。为此,对 VEC 环境下的车辆运动模型以及对 ES 的可用计算资源进行建模分析是有效解决卸载场景动态变化的 VEC 环境中计算卸载性能降低的关键[15]。本文针对该问题提出基于计算资源动态可用性的计算卸载切换策略(DROH),包括在路径固定情况下上传目标服务节点的选择策略和基于动态可用计算资源的切换策略两部分,实验结果证明了本文提出的卸载切换策略能够有效地缩短任务完成时间,降低卸载环境动态变化对卸载性能的影响。

表 1 本文算法和现有最新算法的区别

文献	上传延迟	下载延迟	多跳	移动性管理	Docker	资源动态性	是否切换
文献[8]	×	×	×	√	×	×	×
文献[9]	√	×	×	√	×	×	×
文献[10-11]	√	×	×	√	×	×	×
文献[12]	√	×	×	×	×	×	√
文献[13]	√	√	×	√	×	×	√
本文算法	√	√	√	√	√	√	√

1 系统模型

车载边缘卸载环境中的计算资源类型可分为 3 层卸载架构模型[16],包括汽车终端层、边缘计算层和云计算层,如图 1 所示。

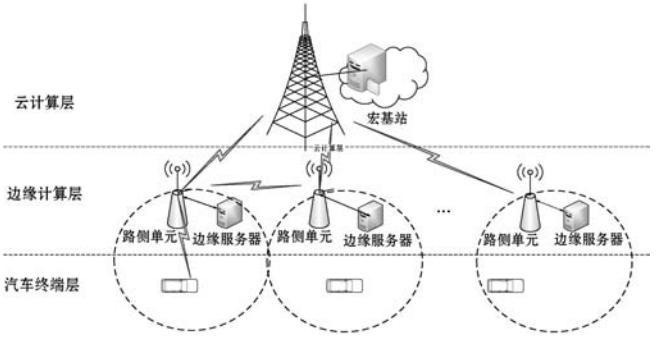

图 1 车载边缘计算架构

1.1 计算卸载模型

假设源车辆节点产生的计算任务为应急任务且可以优先调度[17],将源节点产生的任务 i 表示为向量 $\boldsymbol{Q}_i$,描述为:

$$\boldsymbol{Q}_i = (C, D_{\text{in}}, D_{\text{out}}, t_{\text{d}}) \tag{1}$$

式中:C 表示任务计算量;D_{in} 表示任务上传数据量;D_{out} 表示任务结果数据量;t_{d} 表示任务截止时间,任务完成时间超过该截止时间则表示任务处理失败。

车辆的卸载决策基于卸载环境中服务节点的计算能力、可用通信带宽状态、任务属性等诸多因素执行计算卸载判决。通常当计算任务卸载的完成时间小于本地执行时间时将执行任务卸载,定义 $\eta_i = \{0,1\}$ 为计算卸载决策变量,即当 $\eta_i = 0$ 时,计算任务 i 在源车辆节点上执行,反之当 $\eta_i = 1$ 时,源节点执行计算卸载过程,将任务卸载至边缘服务器以缩短任务完成时间,提高服务质量。

1) 当 $\eta_i = 0$ 时,计算任务 i 在车辆在本地的完成时间表示为:

$$T_i^{\text{L}} = \frac{C_i}{F_i^{\text{L}}} \tag{2}$$

式中:C_i表示任务计算量;F_i^{L}表示为源车辆节点的计算能力。

2）当 $\eta_i=1$ 时,本地完成时间大于任务截止时间,即 $T_i^{\mathrm{L}}>t_{\mathrm{d}}$,任务 i 执行计算卸载,并将任务的卸载分为数据上传、数据计算和数据下载 3 个阶段。

（1）上传阶段:由于 RSU 的服务范围有限且源车辆节点的不断运动,导致汽车节点与 RSU 节点之间的数据传输链路不断发生改变,为缩短数据传输时间,应当选择带宽最大的链路作为传输链路,因此定义最大等效带宽为 $\overline{B}$(单位:MB/s)。

当数据 D 通过多跳传输时,等效带宽由通信链路单跳带宽和跳数 $h(h=1,2,\cdots,n)$ 共同决定,当跳数为 n 时,每一跳的通信时间可以表示为 $t_i(i=1,2,\cdots,n)$,若将该过程视为一跳传输,则可以得到等效带宽为:

$$\overline{B}=\frac{D}{T}=\frac{D}{t_1+t_2+\cdots+t_n}=\frac{D}{\frac{D}{B_1}+\frac{D}{B_2}+\cdots+\frac{D}{B_n}}=\frac{1}{\frac{1}{B_1}+\frac{1}{B_2}+\cdots+\frac{1}{B_n}}\quad n\in h \tag{3}$$

当计算任务 i 上传的等效带宽为$\overline{B}_i^{\mathrm{up}}$时,可得计算任务 i 的上传时间为:

$$T_i^{\mathrm{up}}=\frac{D_i^{\mathrm{in}}}{\overline{B}_i^{\mathrm{up}}} \tag{4}$$

（2）执行阶段:实际场景中 MEC 服务器需要为多个并发的计算任务分配计算资源,每个用户的可用计算资源随着并行处理任务数量的变化而动态变化,计算资源更新间隔为 Δt 时,分配给任务 i 的可用计算资源表示为:

$$f_{ES_i}(t)=\frac{F_{ES_i}}{T_{ES_i}(n)} \tag{5}$$

式中:F_{ES_i}表示 MEC 服务器 E_{S_i}的总计算资源;$T_{ES_i}(n)$表示在第 $n\Delta t$ 时段内 E_{S_i}上的并行处理的任务数;Δt 表示计算能力更新时间间隔。可用计算能力动态变化如图 2 所示。

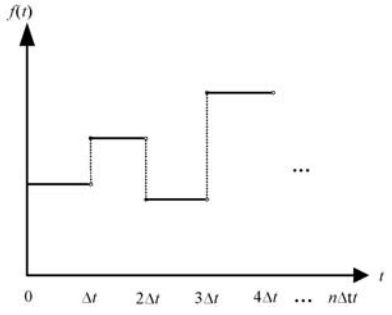

图 2　MEC 服务器可用计算资源动态变化示意图

故计算任务 i 的计算时间为:

$$T_i^{\mathrm{ex}}=\frac{C_i}{f_{ES_j}} \tag{6}$$

（3）下载阶段:下载过程与上传过程类似,下载链路等效带宽为 $\overline{B}_j^{\mathrm{down}}$,则下载时间表示为:

$$T_i^{\mathrm{down}}=\frac{D_i^{\mathrm{out}}}{\overline{B}_j^{\mathrm{down}}} \tag{7}$$

故当源车辆节点将计算任务卸载到边缘服务器时,计算任务 i 的完成时间为上传时间、计算时间和下载时间之和,可表示为:

$$T_i^{\mathrm{E}}=\frac{D_i^{\mathrm{in}}}{\overline{B}_j^{\mathrm{up}}}+\frac{C_i}{f_{ES_j}}+\frac{D_i^{\mathrm{out}}}{\overline{B}_j^{\mathrm{down}}} \tag{8}$$

1.2　计算切换模型

在卸载执行过程中,源车辆节点的不断运动会导致任务当前所在服务节点所提供的计算服务可能不再是最佳的卸载选择,即当发现有满足计算切换准则的候选服务节点时,则将剩余任务从当前服务节点迁移到候选服务节点上,满足用户对服务时延和服务质量的需求。

计算切换的触发条件可定义为:当任务在原服务节点上的剩余任务完成时间大于切换到候选服务节点上的剩余完成时间和切换开销时间之和时,则执行计算切换。即:

$$T_a>T_b+T_{a\to b}^{\mathrm{handoff}} \tag{9}$$

式中:T_a表示卸载任务在原服务节点 E_{S_a}上的剩余完成时间;T_b表示剩余卸载任务切换到候选服务节点 E_{S_b}上的预计完成时间;$T_{a\to b}^{\mathrm{handoff}}$表示触发切换所需要的切换时间开销。根据文献[18]对基于 Docker 的计算切换的过程,切换时间开销可表示为:

$$T_{a\to b}^{\mathrm{handoff}}=T_C^{\mathrm{p}}\times\frac{C_{\mathrm{re}}}{C}+\frac{D_{\mathrm{p}}}{r_{a\to b}}+T_C^{\mathrm{r}}\times\frac{C_{\mathrm{re}}}{C} \tag{10}$$

式中:C_{re}表示卸载任务触发计算切换前的剩余计算量;T_C^{p}表示卸载任务的计算量为 C 时的总打包时间;D_{p}表示打包后的数据量;$r_{a\to b}$表示切换过程中打包文件的传输速率;T_C^{r}表示卸载任务的计算量为 C 时的总重启时间。

假设有 m 个满足计算切换准则的候选服务节点时,选择最小剩余完成时间对应的候选服务节点为切换的目标节点。

1.3　优化目标与约束

计算卸载和切换的目标是使得任务的完成时间最小,将其表示为:

$$\min\ T_i = \min(T_i^{\mathrm{up}} + T_i^{\mathrm{ex}} + T_i^{\mathrm{down}}) \tag{11}$$

$$\text{s.t.}\quad \mathrm{C1}:\eta_i = [0,1] \quad \forall i \in N_V$$
$$\mathrm{C2}:F_i^{\mathrm{L}} < F_j^{\mathrm{E}} \quad \forall i \in N_V, j \in N_{ES}$$
$$\mathrm{C3}:F_j^{\mathrm{E}} < F^{\mathrm{C}} \quad j \in N_{ES}$$
$$\mathrm{C4}:T_{ES_j} \leqslant T_{ES_j}^{\max} \quad \forall j \in N_{ES}$$

式中：约束 C1 表示二进制计算卸载决策，基于当前卸载环境中代理资源的计算能力以及任务属性判断是否执行计算卸载过程，当 $\eta_i = 1$ 时执行计算卸载，当 $\eta_i = 0$ 时任务由源节点在本地执行；C2 表示车辆节点的本地计算能力小于边缘服务器 E_{S} 节点的计算能力；C3 表示服务节点计算能力小于云服务器计算能力；C4 表示每个服务节点上可并行处理的任务数不能超过其最大可并行处理任务数。

2 策略设计

车辆具有有限的计算资源，对于时延敏感型任务，须将其卸载到边缘服务器上以缩短任务完成时间，提高服务质量。在此过程中，周围可用计算资源动态变化，一旦触发计算切换判决准则，则执行计算切换。

1）上传过程。执行卸载切换首先需要链接至服务节点，通常以随机选择和就近选择[12]，但是该方法无法保证数据传输时间最小，因此本文提出基于固定路径的上传选择策略。即在路径固定的情况下，通过收集车辆可能途径的所有服务节点信息，卸载决策模块通过将卸载任务映射到途经的所有服务节点上，获取相应的任务上传时间，并从中选择对应上传时间最短的服务节点作为目标上传节点，进一步降低任务的上传时间开销。具体如下：

在路径固定的情况下，源车辆节点获取卸载环境中的所有服务资源的当前时刻可用计算能力、数据传输带宽、途径 RSU 节点等相关信息，得到待卸载任务在各个服务节点上的预计上传时间：

$$T = (t_i, \cdots, t_j, \cdots, t_k) \tag{12}$$

基于当前时刻上传时间最小的上传原则，选择最小预计上传时间 T_i 对应节点作为目标上传服务节点，即

$$T^{\mathrm{up}} = \min(t_i, \cdots, t_j, \cdots, t_m) \tag{13}$$

待确定目标服务节点后，选择等效带宽最大的传输链路作为上传链路，并假设在一个时间段 t_μ 内任务上传等效带宽 $\overline{B}_{t_\mu}^{\mathrm{up}}$ 保持不变，则在时间段 t_μ 内任务可上传数据量可表示为：

$$D_{t_\mu}^{\mathrm{in}} = \overline{B}_{t_\mu}^{\mathrm{up}} \times t_\mu \tag{14}$$

从 0 时刻到 t_s 时刻，当任务上传累加量等于任务输入数据量，即满足式(15)时，任务上传数据量传输阶段完成，累积上传时间为 $T_i^{\mathrm{up}} = t_s$。

$$\left(\sum_{t_\mu=0}^{t_\mu=t_s} D_{t_\mu}^{\mathrm{in}}\right) = D_i^{\mathrm{in}} \tag{15}$$

2）执行过程。任务上传完成后，服务节点开始执行计算任务，计算切换判决模块实时更新服务节点的选择。假设在 t_x 时刻，计算任务的剩余计算量为 $C_{t_x}^{\mathrm{re}}$，则任务当前时刻剩余任务的预计完成时间 T_i^{re} 为：

$$T_i^{\mathrm{re}} = \frac{C_{t_x}^{\mathrm{re}}}{f_{ES_i}(t_x)} + \frac{D_i^{\mathrm{out}}}{\overline{B}_{RSU_i}^{\mathrm{down}}} \tag{16}$$

式中：$f_{ES_i}(t_x)$ 表示在 t_x 时刻卸载任务在 E_{S_i} 节点可用的计算能力；$\overline{B}_{RSU_i}^{\mathrm{down}}$ 表示当前节点可为卸载任务提供的网络带宽。

同样可得到剩余任务在其他可用代理资源上的预计完成时间：

$$T_j^{\mathrm{re}} = \frac{C_{t_x}^{\mathrm{re}}}{f_{ES_j}(t_x)} + \frac{D_i^{\mathrm{out}}}{\overline{B}_{RSU_j}^{\mathrm{down}}} \quad i,j \in \{1,2,\cdots,m\}, i \neq j \tag{17}$$

基于式(10)可得在 t_x 时刻卸载任务从 E_{S_i} 节点切换到 E_{S_j} 节点的切换时间开销，即 $T_{t_x}^{\mathrm{handoff}}$。则当满足 $T_i^{\mathrm{re}} > T_j^{\mathrm{re}} + T_{t_x}^{\mathrm{handoff}}$ 时，卸载任务从 E_{S_i} 节点切换到 E_{S_j} 节点以降低任务的完成时间开销，并选择剩余完成时间最小的服务节点作为切换目标节点，即：

$$T_j^{\mathrm{re}} = \min(T_{\mathrm{re}}) \tag{18}$$

$$T_{\mathrm{re}} = (T_i^{\mathrm{re}}, \cdots, T_j^{\mathrm{re}}, \cdots, T_m^{\mathrm{re}}) \tag{19}$$

类似地，假设在 t_μ 内任务所在服务节点的计算能力不变，得到在时间 t_μ 内的已完成任务计算量 C_{t_μ}，假设在 t_r 时刻，各服务节点上已完成计算量累加等于任务的计算量，则意味着任务执行完成。则任务在执行过程中的执行时间开销为：

$$T_i^{\mathrm{ex}} = \sum_{j=T_i^{\mathrm{up}}}^{j=t_r} t_j^{\mathrm{ex}} + \sum_{j=T_i^{\mathrm{up}}}^{j=t_r} t_j^{\mathrm{handoff}} \tag{20}$$

3）下载过程。将执行完成时刻任务所在的服务节点作为下载节点，在下载节点到源车辆节点之间找到可用等效带宽最大的传输链路，得到相应的结果下载时间 T_i^{down}。

综合上述三个阶段，形成了完整的基于动态可用计算资源的计算卸载切换策略，如图 3 所示。首先，源节点根据式(13)选择上传时间最短的服务节点作为目标上传节点，随着边缘服务器的并行处理任务数不断变化，计算资源随之改变，根据式(9)判断是否执行切换过程。当剩余计算量为零，执行下载过程，任务结束。

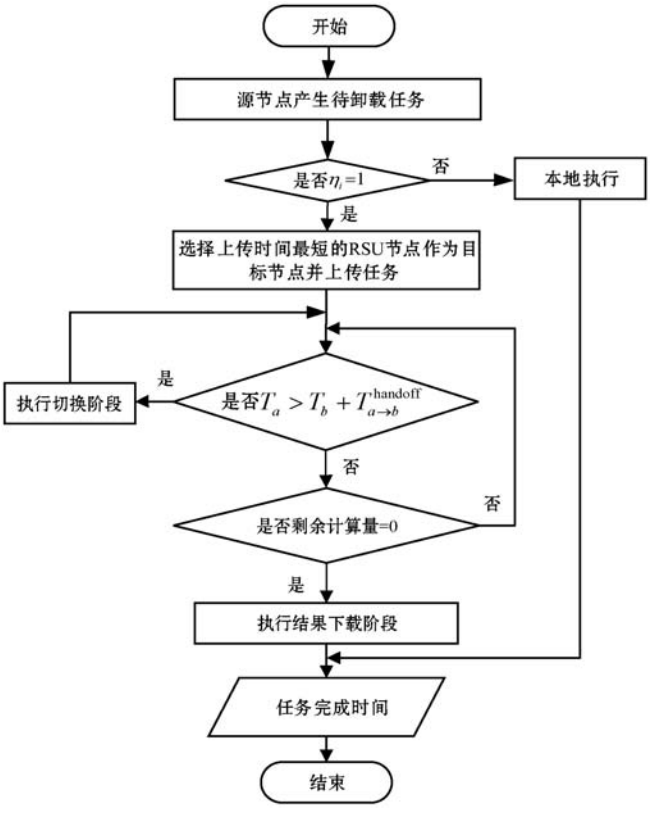

图 3　基于可用计算能力变化的计算卸载切换策略

3　实验仿真与结果分析

3.1　仿真环境与参数设置

实验仿真场景如图 4 所示,其中:星形为 RSU 位置;圆圈为 RUS 通信范围;方块为汽车节点当前位置。每秒对卸载场景中的汽车节点位置信息、代理资源计算能力等相关信息进行采集以及计算切换判决。源节点产生的计算任务可选择在本地执行,或者卸载至 ES 或云端服务器执行,不考虑卸载至其他车辆执行的 V2V 卸载模式。每个 ES 节点可为一定数量的计算任务并发提供计算服务并平均分配其计算能力,代理资源计算能力更新时隙为 20 s;计算切换模式仅考虑 ES 之间的计算切换,不涉及云边切换的模式。实验仿真涉及到的参数如表 2 所示。

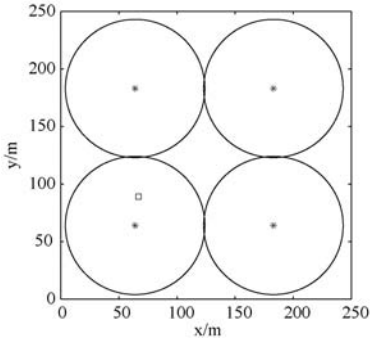

图 4　仿真实验场景

表 2　仿真参数

参数	数值	参数	数值
仿真次数	1 000	车辆计算能力/MIPS	26 499 × [1 2]
RSU 半径/m	[50 100]	ES 计算能力/MIPS	26 499 × [13 16]
RSU 数目	4	V2R 基础带宽/(Mbit · s^{-1})	15
ES 数目	4	R2R 基础带宽/(Mbit · s^{-1})	12
汽车数目	10	V2R 延时/ms	[20 30]
汽车速度/(m · s^{-1})	[15 20]	R2R 延时/ms	[200 300]
任务计算量	[6,8] × 10^{12}	并行任务数	[3 4 5]
任务数据量/× 10^6	[90,110]	更新时隙/s	20

3.2　性能指标

设定仿真次数为 N,以平均任务上传时间、平均任务通信时间、平均任务计算时间、平均任务完成时间为性能指标,并将其定义如下:

平均任务上传时间:卸载任务通过计算卸载的方式将任务上传至代理资源的时间开销,表示为:

$$\overline{T}_{\text{up}} = \frac{\sum_{i=1}^{E_N} (T_{\text{up}}^i)}{N} \tag{21}$$

式中:T_{up}^i表示第 i 次实验的任务上传时间;E_N 为仿真次数。

平均任务通信时间:卸载任务通过计算卸载的方式将任务上传至代理资源的时间开销以及结果下载至终端设备的时间开销,表示为:

$$\overline{T}_{\text{com}} = \frac{\sum_{i=1}^{E_N} (T_{\text{com}}^i)}{N} \tag{22}$$

式中:T_{com}^i表示第 i 次实验的任务通信时间。

平均任务执行时间:卸载任务在代理资源上的完成计算的时间跨度,表示为:

$$\overline{T}_{\text{ex}} = \frac{\sum_{i=1}^{E_N} (T_{\text{ex}}^i)}{N} \tag{23}$$

式中:T_{ex}^i表示第 i 次实验的任务执行时间。

平均任务完成时间:卸载任务执行卸载过程所需要的总体时间开销,包括通信时间开销以及执行时间开销,表示为:

$$\overline{T}_{\text{sum}} = \frac{\sum_{i=1}^{E_N} (T_{\text{sum}}^i)}{N} \tag{24}$$

式中：T^{i}_{sum}表示第 i 次实验的任务完成时间。

3.3 实验结果与分析

为评估本文所提的 DROH 策略的有效性，将其与以下三种算法进行比较。

1）无切换卸载策略 OWH（Offloading Without Han-doff）：源车辆将任务卸载到完成时间最小的服务节点后，不再进行切换，始终由该节点完成计算任务[8]。

2）基于距离最小的切换策略 MDTH（Minimum Co-mmunication Distance-based Handoff）：源车辆将任务卸载到距离最近的服务节点，随着车辆移动，始终将任务迁移到距车辆最近的节点执行计算任务[12]。

3）基于最小完成时间的切换策略 METH（Minimum Execution Time-based Handoff）：源车辆将任务卸载到当前完成时间最小的服务节点，随着计算资源动态变化，始终选择将任务迁移到当前时刻执行时间最短的代理资源上执行，降低任务的执行时间[10]。

在仿真实验环境，通过统计本文提出的 DROH 算法和上述三种算法的仿真实验结果，得到这些算法的性能。

1）平均任务上传时间性能。

由图 5 可知在计算能力动态变化的卸载场景中，OWH 算法和 METH 算法在上传时间开销上相近，这是因为这两种算法都是选择将任务卸载到任务生成时刻预计完成时间最小的代理资源上执行。而 MDOH 算法将任务卸载到距车辆最近的服务节点，所以其平均上传时间小于 OWH 算法和 METH 算法。另外，本文所提 DROH 算法的平均上传时间相较于其他三种算法显著减小。因为 DROH 策略在路径固定的情况下，始终监测车辆的位置信息以及途径的 RSU 节点信息，从而得到源车节点到途径 RSU 节点在每时刻的等效带宽最大的数据传输链路，将任务映射到途径的 RSU 节点上获取相应的任务上传时间，并将对应最小上传时间的 RSU 节点设为目标上传节点，以节省上传时间。

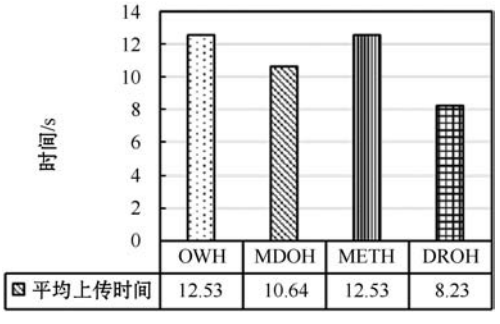

图 5 平均任务上传时间

2）平均通信时间性能。

从图 6 可以看出本文所提的 DROH 策略，平均通信时间相较于 OWH 策略、MDOH 策略和 METH 策略分别缩短了 21.28%、14.17% 和 18.22%，这是因为本文算法能够根据任务的链路带宽，对通信路径进行自适应调整，减小了任务传输的通信开销，优化了用户服务体验。而其他三种算法并未考虑移动路径对任务通信时间的影响，为此其通信开销较大。

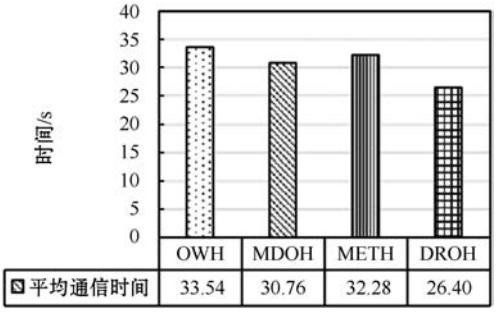

图 6 平均通信时间

3）平均任务完成时间性能。

从平均完成时间的角度看 DROH 策略、MDOH 策略和 METH 策略相较于 OWH 策略分别缩短了 21.94%、15.51% 和 13.41%，如图 7 所示。随着车辆的不断运动，卸载场景中数据传输环境不断发生改变；同时随着 MEC 服务器上并发处理的计算任务数量的变化，MEC 服务器分配给每个计算任务的可用计算资源也在动态变化，OWH 策略仍由当前服务节点提供服务，导致任务执行时间增加，进而导致任务完成时间增加。MDOH 策略将任务迁移到最近的服务节点，但未考虑节点的可用计算资源，所以任务执行时间增加。而 DROH 算法在计算卸载执行过程中，实时监测卸载环境中的资源状态，一旦触发计算切换准则就执行计算切换过程，将任务切换到目标服务器进行计算，降低了计算能力动态变化对计算卸载性能的影响。同时 DROH 策略对比 METH 策略提升 9.86%，这是因为尽管 METH 策略始终由完成时间最小的节点提供计算服务。但仅考虑了服务节点计算能力变化对执行时间的影响，未考虑不同通信链路的不同的通信带宽所产生的影响。而本文所提的 DROH 策略综合考虑任务通信时间和任务执行时间，降低了任务的完成时间。

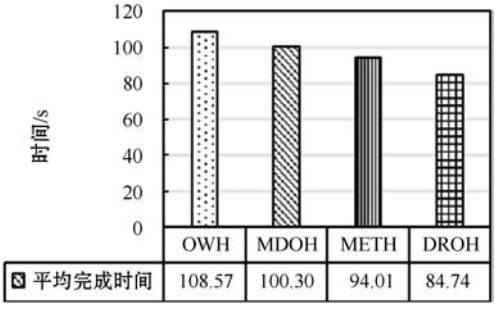

图 7 平均任务完成时间

通过与其他三种算法的对比,可以得出本文所提 DROH 算法的优越性,通过减小通信时间和任务执行时间,最大程度降低任务完成时间。

4 结　语

MEC 技术的引入为解决车载计算任务卸载提供了新的解决方向,然而随着应用场景的不断拓展以及车载用户对实时性服务的需求日益高涨,如何有效解决动态变化的车载环境对计算卸载性能的影响是本文的研究重点。提出在固定路径情况下基于动态可用计算资源的计算卸载切换策略,并通过实验仿真证明了该策略的有效性。未来的工作将集中在多任务和多属性下车-边-云三者的协同,通过三者的协同更好地制定卸载策略,满足任务对时延、能耗和可靠性的要求。

参 考 文 献

[1] Tang C, Wei X, Zhu C, et al. Mobile vehicles as fog nodes for latency optimization in smart cities[J]. IEEE Transactions on Vehicular Technology,2020,69(9):9364 - 9375.

[2] Li B, Hou P, Wang K, et al. Deployment of edge servers in 5G cellular networks[J]. Transactions on Emerging Telecommunications Technologies,2020,32(8):e3937.

[3] 查易艺,袁烨,李金湖,等. 一种移动边缘云计算任务卸载算法[J]. 计算机应用与软件,2020,37(6):135 - 141.

[4] 方加娟,李凯. 基于边缘云和移动辅助设备的计算卸载优化方案[J]. 计算机应用与软件,2020,37(12):6 - 12.

[5] Ning Z, Wang X, Huang A. Mobile edge computing-enabled 5G vehicular networks: Toward the integration of communication and computing[J]. IEEE Vehicular Technology Magazine,2019,14(1):54 - 61.

[6] Li B, Hou P, Wu H, et al. Placement of edge server based on task overhead in mobile edge computing environment[J]. Transactions on Emerging Telecommunications Technologies, 2020,32(9):e4196.

[7] Nasrin W, Xie J. A joint handoff and offloading decision algorithm for mobile edge computing (MEC)[C]//2019 IEEE Global Communications Conference,2019:1 - 6.

[8] Zhang K, Mao Y, Leng S, et al. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading[J]. IEEE Vehicular Technology Magazine,2017,12(2):36 - 44.

[9] Guo S, Liu J, Yang Y, et al. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing[J]. IEEE Transactions on Mobile Computing,2019,18(2):319 - 333.

[10] Chen M, Hao Y. Task offloading for mobile edge computing in software defined ultra-dense network[J]. IEEE Journal on Selected Areas in Communications,2018,36(3):587 - 597.

[11] Zheng J, Cai Y, Wu Y, et al. Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach[J]. IEEE Transactions on Mobile Computing, 2019,18(4):771 - 786.

[12] Taleb T, Ksentini A, Frangoudis P. Follow-me cloud: When cloud services follow mobile users[J]. IEEE Transactions on Cloud Computing,2019,7(2):369 - 382.

[13] Misra S, Bera S. Soft-VAN: Mobility-aware task offloading in software-defined vehicular network[J]. IEEE Transactions on Vehicular Technology,2020,69(2):2071 - 2078.

[14] Rejiba Z, Masip-Bruin X, Marín-Tordera E. A survey on mobility-induced service migration in the fog, edge, and related computing paradigms[J]. ACM Computing Surveys, 2019,52(5):1 - 33.

[15] Meng Z, Xu H, Huang L, et al. Achieving energy efficiency through dynamic computing offloading in mobile edge-clouds [C]//2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems,2018:175 - 183.

[16] Huang X, Xu K, Lai C, et al. Energy-efficient offloading decision-making for mobile edge computing in vehicular networks[J]. EURASIP Journal on Wireless Communications and Networking,2020,2020:35.

[17] 董思岐,李海龙,胡磊,等. 面向优先级用户的移动边缘计算任务调度策略[J]. 计算机应用研究,2020,37(9):1 - 5.

[18] Ma L, Yi S, Li Q. Efficient service handoff across edge servers via docker container migration[C]//2nd ACM/IEEE Symposium on Edge Computing,2017:1 - 13.

(上接第 14 页)

[17] 王诚文,董青秀,穗志方,等. 自然语言处理评测数据集质量评估研究[J]. 中文信息学报,2023,37(2):26 - 40.

[18] 陈茂林,杜欣为,陆兆辉,等. 建立先天性心脏病专科数据库的实践与思考[J]. 中国卫生信息管理杂志,2021,18(5):691 - 695.

[19] 邓嘉乐,胡振生,连万民,等. 基于 RoBERTa-CRF 的肝癌电子病历实体识别研究[J]. 医学信息学杂志,2023,44(6):42 - 47.

[20] 蒋嘉浩,赵国钰,马英博,等. 语义在人脑中的分布式表征:来自自然语言处理技术的证据[J]. 心理科学进展, 2023,31(6):1002 - 1019.

[21] 黄晓芳,陈剑秋,周祖宏,等. 基于 BERT 的电子病历实体关系联合抽取研究[J]. 医学信息学杂志,2023,44(2):28 - 34.