计算机科学
COMPUTER SCIENCE

服务缓存约束下优化用户设备执行成本的任务卸载策略

张俊娜, 陈家伟, 鲍想, 刘春红, 袁培燕

引用本文

张俊娜, 陈家伟, 鲍想, 刘春红, 袁培燕. 服务缓存约束下优化用户设备执行成本的任务卸载策略[J]. 计算机科学, 2023, 50(10): 275-281.

ZHANG Junna, CHEN Jiawei, BAO Xiang, LIU Chunhong, YUAN Peiyan. Cost-minimizing Task Offload Strategy for Mobile Devices Under Service Cache Constraint [J]. Computer Science, 2023, 50(10): 275-281.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220900185

服务缓存约束下优化用户设备执行成本的任务卸载策略

张俊娜[1,2]　陈家伟[1]　鲍　想[1]　刘春红[1]　袁培燕[1]
1 河南师范大学计算机与信息工程学院　河南 新乡 453007
2 河南师范大学智慧商务与物联网技术河南省工程实验室　河南 新乡 453007
(jnzhang@htu.edu.cn)

摘　要　边缘计算通过在网络边缘侧提供更优的计算和存储能力,能够有效降低用户设备的执行时延和能耗。随着应用程序对计算和存储资源的需求越来越大,任务卸载作为消除用户设备固有限制的一种有效手段,成为了主要的研究热点之一。然而,在已有的任务卸载研究中,常常忽略不同类型的任务对服务需求的多样性以及边缘服务器服务缓存有限的情形,从而导致不可行的卸载决策。因此,在服务缓存约束下,研究了能够使得用户设备执行成本最优的任务卸载问题。首先设计了云服务器、边缘服务器和本地设备的协同卸载模型,用于平衡边缘服务器的负载问题,同时借助云服务器弥补边缘服务器有限的服务缓存能力。然后,提出了适用于云边端协同的任务卸载算法,优化用户设备的执行成本。当任务被卸载时,先采用改进的贪婪算法选择最佳的边缘服务器,再通过比较任务在不同位置上的执行成本,来确定任务的卸载决策。实验结果表明,所提算法相比对比算法能够有效降低用户设备的执行成本。

关键词:边缘计算;任务卸载;云边端协同;服务缓存;卸载策略优化
中图法分类号　TP302

Cost-minimizing Task Offload Strategy for Mobile Devices Under Service Cache Constraint

ZHANG Junna[1,2],CHEN Jiawei[1],BAO Xiang[1],LIU Chunhong[1] and YUAN Peiyan[1]
1 School of Computer and Information Engineering,Henan Normal University,Xinxiang,Henan 453007,China
2 Engineering Lab of Intelligence Business & Internet of Things,Henan Normal University,Xinxiang,Henan 453007,China

Abstract　Edge computing provides more computing and storage capabilities at the edge of the network to effectively reduce execution delay and power consumption of mobile devices. Since applications consume more and more computing and storage resources,task offloading has become one of effective solutions to address the inherent limitations in mobile terminals. However,existing researches on task offloading often ignore the diversity of service requirements for different types of tasks and that edge servers have limited services capabilities,resulting in infeasible offloading decisions. Therefore,we study the task offloading problem that can optimize the execution cost of mobile devices under service cache constraints. We first design a collaborative offloading model integrated remote cloud,edge server and local device to balance the load of edge server. Meanwhile,cloud server is used to make up for the limited-service caching capacity of the edge server. Secondly,a task offloading algorithm suitable for cloud-edge-device collaboration is proposed to optimize the execution delay and energy cost of mobile devices. When the task is offloaded,the improved greedy algorithm is used to select the best edge server. Then,the offload decision of the task is determined by comparing the execution cost of the task at different locations. Experimental results show that the proposed algorithm can effectively reduce the execution cost of mobile devices compared with the comparison algorithms.

Keywords　Edge computing,Task offloading,Cloud-Edge-Device collaboration,Service caching,Offloading strategy optimization

1　引言

物联网和用户设备的广泛使用,以及第五代移动通信技术的飞速发展,推动了时延敏感型和资源密集型应用的发展,如自动驾驶、人脸识别、虚拟和增强现实、远程医疗等[1]。然而,由于用户设备的通信带宽、计算和存储能力有限,在执行上述应用时会产生较大延迟和能耗[2],且仅依靠云计算模式难以满足应用程序对时延和能耗的需求。为了提升用户体验质量,解决因云服务器离用户设备较远而带来的高延迟和高能耗问题[3],边缘计算(Edge Computing,EC)应运而生。

到稿日期:2022-09-17　返修日期:2022-12-06
基金项目:国家自然科学基金(61902112,62072159);广西密码学与信息安全重点实验室课题(GCIS202115)
This work was supported by the National Natural Science of China(61902112,62072159) and Guangxi Key Laboratory of Cryptography and Information Security(GCIS202115).
通信作者:陈家伟(chenjiawei@stu.htu.edu.cn)

通过将云端的计算、存储等资源优势“下沉”到移动网络边缘侧[4]，用户设备产生的任务能够直接卸载到邻近的边缘服务器执行，从而降低任务的传输时延和执行能耗。

任务卸载作为消除用户设备固有限制的一种有效手段，成为了边缘计算领域主要的研究热点之一。然而，在已有的任务卸载研究中，还存在一些局限性。1)假设 EC 服务器缓存有任务所需的所有服务[2,5]。边缘服务器因有限的存储能力只能缓存有限类服务，而每个被卸载任务往往需要特定的服务支持才能够被执行，即任务只能被卸载到缓存有相应服务的边缘服务器。例如，在人脸识别应用程序中，“特征提取”任务只有被卸载到配置有机器学习模型的边缘服务器上才能够被成功执行。因此，由边缘服务器的服务缓存决定有效的卸载策略。2)假设边缘服务器和用户设备协同卸载模型就能满足用户需求[6-7]。例如，文献[6]和文献[7]考虑到边缘服务器有限的处理资源，将超出边缘服务器计算能力的任务交由用户设备处理；文献[8]还借助于相邻的用户设备。虽然在一定程度上提高了任务的执行效率，但用户设备和边缘服务器的计算和存储能力均有限，当用户设备对资源需求量较大时，边缘服务器和用户设备将不能满足对资源的需求[9]，从而导致不可行的卸载决策或更长的执行时间。

针对上述局限性，本文研究在边缘服务器拥有有限服务缓存和计算能力的情形下，如何获得高效的任务卸载决策，以优化用户设备的计算时延和能耗(即执行成本)。本文的主要贡献如下：

1)提出了基于云服务器、边缘服务器和本地设备协同的任务卸载模型。借助于云服务器强大的存储能力，弥补边缘服务器有限的服务缓存能力，确保任务在服务约束下能够被成功执行。

2)从用户设备角度提出了一种适用于云边端协同的任务卸载算法。该算法综合考虑了不同任务所需服务种类的不同以及边缘服务器的计算和存储能力，以最小化用户设备的执行成本为目标，获取最佳的任务卸载策略。

3)为了评估本文所提方案的有效性，分别与其他的卸载策略进行对比实验。实验结果表明，本文提出的基于时延和能耗成本最小化的任务卸载算法能够有效降低用户设备的执行成本。

本文第 2 章回顾了相关工作；第 3 章对卸载模型进行了描述，并对问题进行了形式化建模；第 4 章阐述了在云边端协同的卸载模型下任务卸载算法的具体实现；第 5 章进行了实验验证，评估了本文模型和算法的性能；最后总结全文。

2 相关工作

近年来，作为 EC 研究领域的热点，任务卸载引起了广泛的关注和研究。许多学者提出了不同的任务卸载模型和方法，并取得了较好的研究成果。

文献[2]研究了 EC 场景中任务可划分为多个子任务情形时，单用户和多用户的边端协同任务卸载问题，提出了一种启发式计算卸载算法，实现了用户设备的整体性能优化。文献[5]研究了超密集组网的 EC 场景下的任务卸载问题，将卸载决策和资源分配作为一个联合优化问题，在用户最大时延约束条件下，提出了一种有效的卸载方案。然而，上述研究在卸载任务时均没有考虑边缘服务器的负载问题，可能存在服务器超载或宕机的风险。

为了兼顾 EC 服务器的负载问题，文献[6]将超出边缘服务器计算能力的任务交由本地设备处理，在满足能量限制的情况下，提出了一种基于李雅普诺夫优化的协同实时在线算法，高效地执行任务卸载和数据缓存决策。文献[7]通过协同本地设备与边缘服务器，提出了基于深度模型的任务调度策略，充分利用本地设备的便捷性和边缘服务器的计算能力，来提升任务的执行效率，降低应用程序的执行时延和能耗。此外，为了合理利用附近空闲设备的处理资源，缓解边缘服务器计算能力有限的问题，文献[8]设计了一个 D2D 辅助的 EC 系统，在此系统中任务能够卸载到本地设备、相邻设备和边缘服务器中进行处理，并且提出了基于背包问题的预分配算法和新的交替优化算法，实现了对任务的卸载决策和传输功率的优化。然而，上述研究虽然考虑到了边缘服务器有限的处理资源和能力，但均只建立了边端协同卸载模型，只适合应用于中小型应用程序，而大型应用程序对计算和资源的需求量很大，只通过边端协同卸载模型可能无法满足用户对时延和能耗的需求。

为了弥补边端协同卸载模型的不足，文献[10-11]建立了云边端协同卸载模型，将云服务器也作为任务的卸载节点之一。文献[10]为云服务器与 EC 的协同做出了早期努力，并在此基础上研究了 EC 资源约束和多用户之间干扰下的任务卸载问题，提出了一种迭代启发式算法求解最优的卸载策略，以满足最优的任务执行延迟。文献[11]在考虑边缘服务器能力有限的情况下，设计了一个兼顾计算时延和能耗的云边端协同的系统模型，并提出了基于强化学习的任务卸载决策算法。然而，上述研究均假定边缘服务器缓存有所有类型的服务。在实际情况中，边缘服务器因有限的存储能力只能缓存有限种类的服务。因此，在制定卸载决策时需考虑边缘服务器上的服务缓存，以确保任务被成功执行。

本文综合考虑了任务请求服务类型的多样性(即不同类型的任务所需的服务种类不同)和边缘服务器有限的计算能力和服务缓存，从用户设备的角度出发，提出了一种云边端协同的任务卸载模型和基于时延和能耗成本最小化的任务卸载算法，以期获得最优的卸载策略，优化用户设备的执行成本。

3 系统模型与问题建模

3.1 系统模型

本文建立的系统模型如图 1 所示，该系统模型包含 3 类计算资源：用户设备、边缘服务器、云服务器。3 类计算资源有着不同的计算处理能力与数据通信能力。

用户设备的集合用 $N=\{1,2,\cdots,n\}$ 表示，虽然用户设备的处理能力较弱，但任务从用户设备产生，可以直接在用户设备上执行，即不会产生通信延迟。

用户设备附近部署有多个边缘服务器，用 $M=\{1,2,\cdots,m\}$ 表示边缘服务器的集合，每个边缘服务器缓存有不同类型的服务。边缘服务器之间通过局域网或蜂窝网相互连接。用户设备可以通过无线接入点或移动网络接入边缘计算网络。

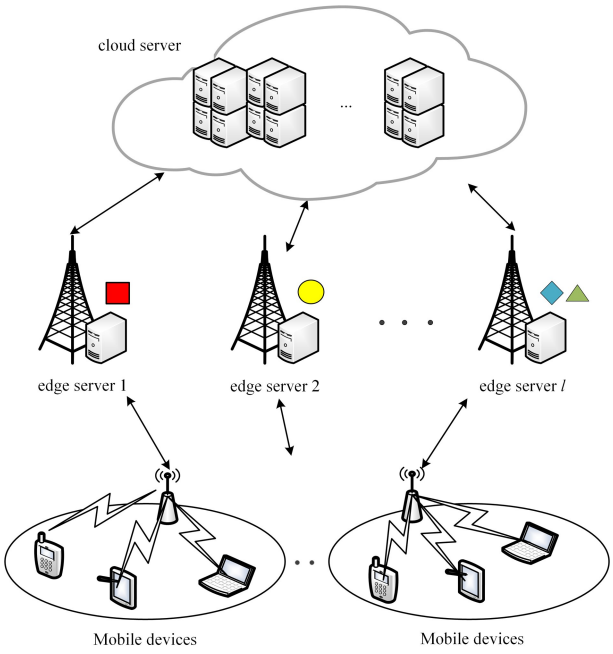


图 1 边缘计算网络模型图

Fig. 1 Edge computing network model

云服务器用 C 表示，其不仅具有强大的处理能力，且可以缓存所有所需服务。但它与用户设备距离较远，意味着用户设备与云服务器之间会产生较大的通信延迟。此外，用户设备需借助距离其最近的边缘服务器才能将任务卸载至云服务器。

本文假定每个用户设备（如智能手机、机器人和无人机等）有 V 个独立实时任务。根据每个任务的特征，及边缘服务器的工作负载、响应时间、缓存服务、延迟和能量消耗等特定参数[12]，用户设备需为每个任务在网络中找到一个最佳的位置来执行，且每个任务只有在一个计算资源上执行。

图 2 为任务集合在云边端协同模型下考虑边缘服务器服务缓存的卸载实例图。云服务器包含任务请求的所有服务。每个任务根据自身特点和所需要的服务，只能被卸载到特定的边缘服务器或云服务器上执行[13]。例如，在边缘计算网络中，只有边缘服务器 1 缓存有可执行任务 1 的服务（“红色方块”），此时任务 1 只能被卸载到边缘服务器 1 上执行。而所有的边缘服务器均没有缓存满足任务 2 所需要的服务（“紫色四角形”），此时任务 2 只能选择卸载到云服务器或在本地设备上执行。此外，虽然边缘服务器 3 缓存有任务 4 所需要的服务（“蓝色菱形”），但任务 4 在该场景下选择在本地设备执行。

由图 2 所示的任务卸载实例可知，不同任务执行时需要不同类型的服务，为了确定任务实际能够卸载的位置，将缓存有任务 $v\in V$ 所需服务的边缘服务器集合用 M_v 表示，即任务 v 只有被卸载到 M_v 中的一个边缘服务器上才能被执行。当集合 M_v 为空时，任务只能被卸载到云服务器或在本地设备执行。本文用二进制变量 $\alpha_v,\beta_v^m,\gamma_v$ 代表任务的卸载决策，$\alpha_v,\beta_v,\gamma_v\in\{0,1\}$。$\alpha_v=1$ 时表示任务 v 选择本地执行，否则为 0。$\beta_v^m=1$ 表示任务 v 卸载至边缘服务器 m 执行，否则为 0。$\gamma_v=1$ 表示任务 v 选择在云服务器执行，否则为 0。此外，每个边缘服务器的计算和存储资源也是有限的，本文假设每个边缘服务器 m 具有的处理资源（CPU 周期数）为 $C(m)$。同时，当任务 v 被卸载到边缘服务器 m 上执行时，每单位时间消耗 r_v^m 的 CPU 周期。

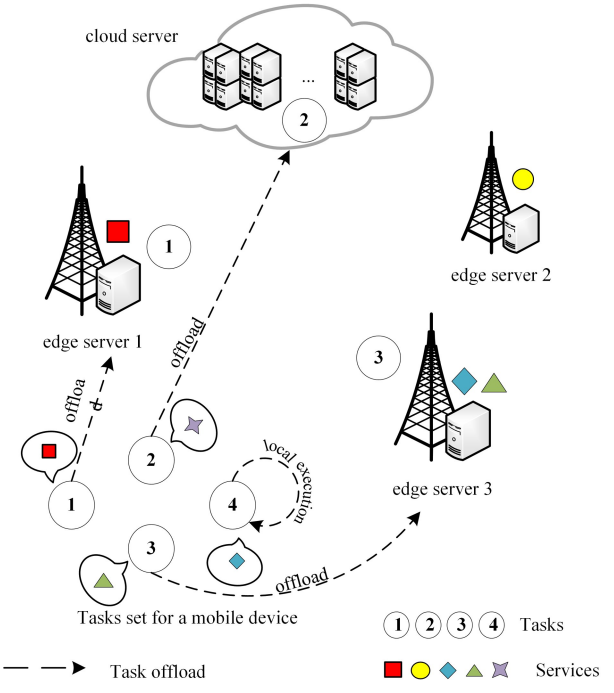


图 2 任务在云边端协同下的卸载实例图（电子版为彩图）

Fig. 2 Example of tasks offloading under cloud-edge-device collaboration

3.2 本地设备时延和能耗模型

本节分析任务选择在本地执行、卸载至边缘服务器或云服务器执行时，本地设备会产生的时延和能耗。

1）本地设备处理

当卸载决策决定在本地设备处理任务 v 时（即 $\alpha_v=1$），任务 v 在本地设备会产生执行时延和执行能耗，分别用 $T_{n,v}^l$ 和 $E_{n,v}^l$ 表示。那么有：

$$T_{n,v}^l=\frac{L_v}{f_l},E_{n,v}^l=kf_l^aT_{n,v}^l=k\frac{(L_v)^a}{(T_{n,v}^l)^{a-1}} \tag{1}$$

其中，L_v 代表处理任务 v 所需的计算量；f_l 代表本地设备 CPU 频率，受最大频率约束 $f_l\leqslant f_{\max}$；$k>0$ 表示计算能耗的效率参数，取决于芯片架构的有效开关电容；$a\geqslant 2$ 代表指数参数[14]。

2）边缘服务器处理

当卸载决策将任务 v 卸载到边缘服务器 m 执行时（即 $\beta_v^m=1$），在本地设备会产生时延，包括任务卸载至边缘服务器的传输时延、任务在边缘服务器上的执行时延和将任务执行结果传回本地设备的传输时延；在本地设备会产生能耗，包括将任务卸载至边缘服务器时的卸载能耗，接收任务执行结果时的接收能耗。由于任务的执行结果一般较小，因此本文不考虑将其传回本地设备的时延，及在接收执行结果时产生的接收能耗。

假定本地设备的发送功率用 p_n 表示，本地设备和边缘服务器间的信道增益用 h_{nm} 表示，带宽用 B^e 表示，则本地设备到

边缘服务器的传输速率可表示为：

$$R_n^e = B^e \log_2\left(1+\frac{p_n h_{nm}}{\sigma^2}\right) \tag{2}$$

其中，σ^2 是数据传输时产生的信噪功率。

因此，本地设备将任务 v 卸载到边缘服务器所消耗的时间 $T_{n,v}^{te}$ 和能耗 $E_{n,v}^e$ 为：

$$T_{n,v}^{te} = \frac{S_v}{R_n^e}, E_{n,v}^e = p_n T_{n,v}^{te} \tag{3}$$

其中，S_v 表示任务 v 的上传数据大小。

边缘服务器处理任务 v 所消耗的时间为：

$$T_{n,v}^u = \frac{L_v}{f_e} \tag{4}$$

其中，f_e 表示边缘服务器 CPU 的频率。本文假设 $f_e > f_{\max}$，即边缘服务器的计算能力大于本地设备。

因此，边缘服务器处理任务 v 所消耗的总时延为：

$$T_{n,v}^e = T_{n,v}^{te} + T_{n,v}^u \tag{5}$$

3）云服务器处理时间

当卸载决策决定将任务 v 卸载到云服务器执行时（即 $\gamma_v=1$），本地设备产生的时延包括传输时延和任务在云服务器的执行时延。传输时延包含任务从本地设备传输到边缘服务器的传输时延，及从边缘服务器传输到云的传输时延。本地设备产生的能耗与任务在边缘服务器上执行分析所产生的能耗相同，因此只计算将任务卸载至边缘服务器时的卸载能耗 $E_{n,v}^e$。

假定边缘服务的传输功率为 p_m，则边缘服务器和云之间的数据传输速率可表示为：

$$R_e^c = B^c \log_2\left(1+\frac{p_m h_{mc}}{\sigma^2}\right) \tag{6}$$

其中，B^c 表示边缘服务器和云之间的带宽，h_{mc} 表示信道增益。那么，任务从本地设备卸载到云服务器所消耗的时间为：

$$T_{n,v}^{tc} = T_{n,v}^e + \frac{S_v}{R_e^c} \tag{7}$$

云服务器执行任务 v 的时间为：

$$T_{n,v}^s = \frac{L_v}{f_c} \tag{8}$$

其中，f_c 表示云服务器 CPU 的频率。因此，云服务器处理任务 v 所消耗的总时延为：

$$T_{n,v}^c = T_{n,v}^{tc} + T_{n,v}^s \tag{9}$$

综上所述，在本文所提的云边端协同卸载模型下，用户设备的总时延和总能耗为：

$$T = \sum_{n=1}^{N}\sum_{v=1}^{V}(\alpha_v T_{n,v}^l + \sum_{m\in M}\beta_v^m T_{n,v}^e + \gamma_v T_{n,v}^c) \tag{10}$$

$$E = \sum_{n=1}^{N}\sum_{v=1}^{V}\alpha_v E_{n,v}^l + (1-\alpha_v)E_{n,v}^e \tag{11}$$

3.3 问题形式化

本文的目标是通过优化任务的卸载决策，最小化用户设备的总时延和总能耗，也就是执行成本 $TEC=\theta T+(1-\theta)E$，其中 $\theta\in[0,1]$ 为用户偏好权重。

每个用户设备都包含多个任务，且每个任务只能在一个计算资源（云服务器、边缘服务器或本地设备）上执行。故可表示为：

$$\alpha_v + \sum_{m\in M}\beta_v^m + \gamma_v = 1, \forall v\in V \tag{12}$$

当任务 v 被卸载到边缘服务器上执行时，还要满足服务约束（$m\in M_v$，本地设备和云服务器不考虑服务约束问题），于是式(12)转变为：

$$\alpha_v + \sum_{m\in M_v}\beta_v^m + \gamma_v = 1, \forall v\in V \tag{13}$$

同时，当任务卸载到边缘服务器上执行时，每个任务都应满足边缘服务器上处理资源的约束，即：

$$\sum_{v\in V}\beta_v^m T_{n,v}^u r_v^m \leqslant C(m), \forall m\in M \tag{14}$$

因此，本文所研究的问题可形式化为如下优化问题：

$$\begin{aligned}&\underset{\alpha,\beta,\gamma}{\text{minimize}}\ \theta T+(1-\theta)E\\ &\text{s.t. } (12),(13),(14),\\ &\quad 0\leqslant f_l\leqslant f_{\max},\\ &\quad T_{n,v}^l, T_{n,v}^e, T_{n,v}^c\geqslant 0, \forall v,\\ &\quad \alpha_v,\beta_v^m,\gamma_v\in\{0,1\}, \forall v,m\end{aligned} \tag{15}$$

4 算法设计与描述

为了确定用户设备需卸载的任务集合 V 中每个任务最优的卸载位置（可为本地设备、边缘服务器或云服务器），本章提出基于贪心策略的任务卸载算法。由于用户设备周围部署有多个边缘服务器，任务在选择卸载到边缘服务器时，需要考虑边缘服务器的处理资源以及是否缓存有所需的服务，从而挑选出满足任务执行的最佳边缘服务器。因此，本章在给出任务卸载算法前，先分析边缘服务器的选择策略。

4.1 边缘服务器选择策略

假定边缘服务器 m 的处理资源为 $C(m)$，$R(m)$ 表示边缘服务器 m 剩余的处理资源，$M_v(R)$ 表示同时满足处理资源约束和服务约束的边缘服务器的集合，即：

$$R(m) = R(m) - T_{n,v}^u r_v^m \tag{16}$$

$$M_v(R) = M_v \cap \{m \mid R(m)\geqslant T_{n,v}^u r_v^m, m\in M\} \tag{17}$$

本文提出了基于贪婪策略的边缘服务器选择算法（Edge Server Selection algorithm，ESS），其具体实现过程如算法 1 所示。当任务选择在边缘服务器上执行时，ESS 首先挑选出能够满足任务服务请求的边缘服务器 M_v，然后保留具有执行任务 v 所需处理资源的边缘服务器 $M_v(R)$（步骤 2－步骤 7）。最后从 $M_v(R)$ 中选择剩余处理资源最多的边缘服务器作为任务将要卸载的节点（步骤 8）。

算法 1 基于贪婪策略的边缘服务器选择算法（ESS）

输入：边缘服务器集合 M，任务 v

输出：所要挑选的边缘服务器 q

1. 为每个边缘服务器编号，$M=\{m_1, m_2, \cdots, m_m\}$
2. FOR 每个边缘服务器 m in M do
3. IF m 能够满足任务 v 的服务请求且执行任务 v 的所需处理资源 $T_{n,v}^u r_v^m \leqslant R(m)$ THEN
4. 将 m 添加到 $M_v(R)$ 的集合中；
5. 将 m 对应的剩余处理资源保存到集合 res_e；
6. END IF
7. END FOR
8. $q \leftarrow$ $maxres_e$ 对应的边缘服务器编号
9. Return q

4.2 任务卸载策略

本文的目标是在所建网络模型中寻找任务最优的卸载

决策方案，从而最小化用户设备的执行成本。因此，本文提出了基于贪心策略的任务卸载算法，用于寻找任务执行成本最小的计算资源位置，解决任务卸载决策问题。算法的具体实现过程如算法 2 所示。

算法 2 任务卸载算法

输入：用户设备 N，任务集合 V，边缘服务器集合 M

输出：最优卸载决策 α_v，β_v^m，γ_v，执行成本 TEC

1. 将卸载决策 α_v，β_v^m，γ_v 和执行成本 TEC 初始化为 0
2. FOR m=1 to M
3. 初始化变量 R(m)为 C(m)；
4. END FOR
5. FOR n=1 to N
6. FOR v=1 to V
7. 初始化变量 $CT_{l,v}$，$CT_{e,v}$，$CT_{c,v}$，值均为 0；
8. 根据式(18)、式(20)，分别计算任务在本地和卸载到云服务器上的执行成本 $CT_{l,v}$和 $CT_{c,v}$；
9. 根据 ESS 算法挑选出最优的边缘服务器，计算任务在该边缘服务器上的执行成本 $CT_{e,v}$；
10. IF $CT_{l,v}=\min\{CT_{l,v},CT_{e,v},CT_{c,v}\}$ THEN
11. $\alpha_v \leftarrow 1$；
12. 保存任务 v 的执行成本 $cost_{n,v} \leftarrow CT_{l,v}$；
13. ELSE IF $CT_{e,v}=\min\{CT_{l,v},CT_{e,v},CT_{c,v}\}$THEN
14. $\beta_v^m \leftarrow 1$；
15. 保存任务 v 的执行成本 $cost_{n,v} \leftarrow CT_{e,v}$；
16. 通过式(16)更新该边缘服务器的剩余处理资源 R(m)；
17. ELSE
18. $\gamma_v \leftarrow 1$；
19. 保存任务 v 的执行成本 $cost_{n,v} \leftarrow CT_{c,v}$；
20. END IF
21. 执行成本 TEC=TEC+$cost_{n,v}$；
22. END FOR
23. END FOR
24. 输出每个任务的卸载决策 α_v，β_v^m，γ_v
25. 输出用户设备的总执行成本 TEC

首先，对于每个任务 $v \in V$，如果被卸载到云服务器或边缘服务器时，分别用 CT_c 和 CT_e 表示其执行成本，用 CT_l 表示它在本地设备执行的成本。因此，当卸载任务 $v \in V$ 时，它在本地设备、边缘服务器和云服务器上的执行成本可分别表示为：

$$CT_{l,v}=\theta T_{n,v}^l+(1-\theta)E_{n,v}^l \tag{18}$$

$$CT_{e,v}=\theta T_{n,v}^e+(1-\theta)E_{n,v}^e \tag{19}$$

$$CT_{c,v}=\theta T_{n,v}^c+(1-\theta)E_{n,v}^e \tag{20}$$

然后，通过步骤 10－步骤 20，比较卸载任务 v 在不同计算资源上的执行成本，选择执行成本最小的计算资源作为任务 v 最优的卸载位置。重复上述过程，直到所有的任务均执行完毕。

在任务卸载算法中，需要遍历每个任务节点，我们假设目标网络中任务的总量为 n，其时间复杂度为 $O(n)$。并且，在确定任务卸载位置时，需要利用 ESS 算法挑选出最优的边缘服务器，该过程需要遍历所有边缘服务器，时间复杂度为 $O(m)$。因此，本文任务卸载算法的时间复杂度为 $O(n\times m)$。

5 实验和性能评估

为了验证本文方法的有效性，将所提出的算法与基于贪婪策略的启发式算法(HAGP)[15]、基于背包问题的预分配算法(PA)[8]以及随机卸载算法进行比较。HAGP 是基于本地设备和边缘服务器协同下成本最小化的卸载策略。PA 是当多个边缘服务器满足任务卸载条件时，选择处理资源少的边缘服务器作为卸载位置。随机卸载指用户设备中所有任务的随机卸载。

假设用户设备数量为 100，每个用户设备随机生成多个任务，任务的类型(请求服务的种类)分为 10 种。边缘服务器的数量为 5，且随机缓存有多种服务。具体的参数设置如表 1 所列。除非另有说明，所有仿真结果都是 100 次独立模拟的平均性能。

表 1 符号说明

Table 1 Notations description

$B^e=B^c=10^6$ Hz	$a=3$
$L_v \in [50,200]\times 10^6$ Cycles	$k=10^{-26}$
$f_{\max}=0.5$ GHz	$f_e=10$ GHz
$\sigma^2=10^{-10}$ Watt	$p_n=0.1$ Watt
$S_v=[2,5]$ Mb	$f_c=100$ GHz

实验配置为：华硕 GL552VW，Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz，12GB of RAM，Windows 10 64 位，实验环境为 python 3.7.8。

本文首先通过改变权重参数 θ 来考察目标函数中两个竞争目标之间的最优性能权衡，即计算延迟 T 和能耗 E。如图 3 所示，E 首先随着 θ 的降低快速下降，当 $\theta \leqslant 0.25$ 时，E 逐渐成为恒量；T 随着 θ 的降低逐渐增加。该曲线可用于设定适当的 θ 值，例如，当用户设备要求总能耗小于 15 J 时，可选择设定 $\theta=0.25$。

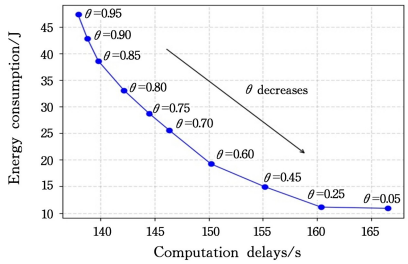


图 3 不同权重 θ 下联合优化的能耗和时延的权衡

Fig. 3 Energy consumption and latency tradeoffs for joint optimization with different weights θ

图 4 和图 5 给出了 4 种算法在不同权重参数 θ 下用户设备执行成本和任务卸载比例(任务卸载到边缘服务器和云服务器的数量占比)之间的关系。θ 的取值从 0.25 变化到 0.75，θ 越大(小)，表示用户更偏好时延(能耗)。TEC 随着 θ 的增加而变大，这是因为任务卸载到云服务器的传输时间会远多于任务在本地或边缘服务器的计算时间，且边缘服务器有限的服务种类和处理资源导致部分任务只能选择在本地执行，任务卸载比例降低(如图 5 所示，本文所提算法和 PA 的卸载比例均随着 θ 的增大而减小)，从而在本地设备产生大量的执行能耗，造成 TEC 的上升。而由于本文算法能够根据

边缘服务器的剩余处理资源均衡地分配任务卸载位置，更多的任务能够卸载至边缘服务器执行，因此相比对比算法，本文算法总能维持较高的卸载比例(见图 5)。因此，用户设备的执行成本在本文算法下总能保持成本最小。总体上看，在 θ 变化时，本文算法相比对比算法总能保持最优的性能。

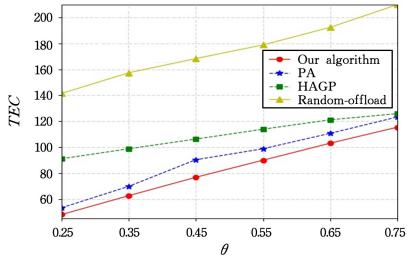


图 4 不同权重 θ 下的算法性能比较

Fig. 4 Algorithm performance comparison with different weights θ

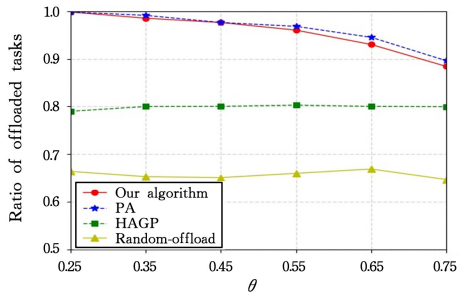
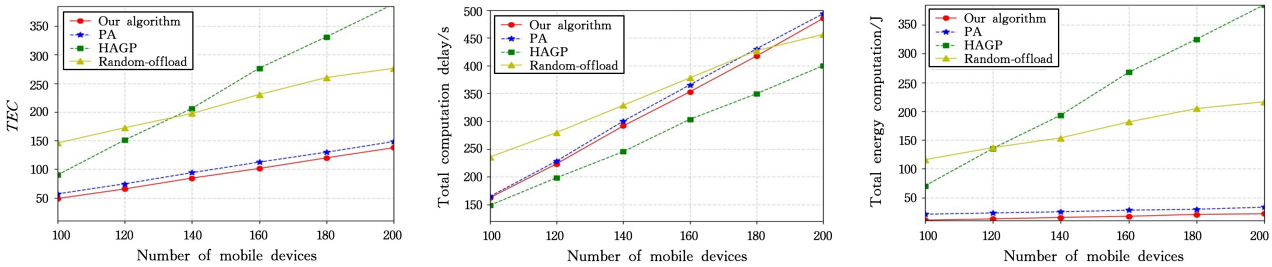


图 5 权重参数 θ 与任务卸载比例的关系

Fig. 5 Relationship between weight parameter θ and task offloading ratio

图 6 给出了不同用户设备数量下成本、时延和能耗的比较。在该实验中，我们假定用户设备要求的总能耗要小于 15J，故设定 θ 的取值为 0.25。用户设备数量范围为 100～300，每个用户设备包含 10 个任务。



(a)TEC (b)Computation delay (c)Energy consumption

图 6 移动设备数量增加对成本、时延和能耗的影响

Fig. 6 Impact of increased number of mobile devices on cost, latency and energy consumption

图 6(a)中，在所有卸载策略下，总成本都随着移动设备数量的增加而增加，但与其他 3 种策略相比，本文提出的策略成本最小，可以降低 8%以上。图 6(b)和图 6(c)描述了在不同算法下最优卸载策略对应的时延和能耗的比较。虽然本文算法具有最小的成本，但是计算时延相对较大，如图 6(b)所示。这是因为当 $\theta=0.25$ 时更强调最小化能量消耗，任务更偏向于卸载至边缘服务器和云服务器上执行。而本文算法能够更好地利用边缘服务器的处理资源，卸载至边缘服务器的任务数量更多，从而产生更大的传输时延，但降低了任务在本地设备上的执行总能耗。图 6(c)中，本文算法具有最小的能耗。

图 7 给出了边缘服务器不同处理资源大小对于不同算法下成本的比较。在该实验中，移动设备数量为 100，θ 为 0.25，边缘服务器处理资源大小从 5 增加到 25。边缘服务器具有更多的处理资源，意味着更多的任务能够被卸载至边缘服务器，从而减少任务在本地设备上执行以及卸载到远程云的数量。因此，针对 4 种算法，TEC 均会降低。而本文算法能够更好地利用边缘服务器的处理资源，卸载至边缘服务器的任务数量更多，故其成本均低于其他算法。HAGP 具有更高的下降率，原因是该算法只考虑边端协同，当处理资源较少时，任务大部分在本地设备处理，成本较大。当处理资源增加到 25 时，大部分任务均能卸载到边缘服务器上执行，因此成本下降较快。同时能够看出，本文算法在不同计算资源大小情况下均具有最低的计算成本。

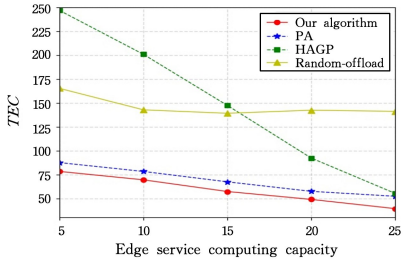


图 7 边缘服务器处理能力与成本的关系

Fig. 7 Relationship between edge server computing capacity and TEC

因此，在服务缓存约束下，面对不同用户设备数量以及边缘服务器不同的处理资源大小，本文提出的适用于云边端协同的任务卸载策略均优于其他卸载方案，有效降低了用户设备的执行成本。

结束语 本文研究了边缘服务器有限服务缓存约束下的任务卸载问题。通过综合考虑用户对时延和能耗的需求，以及不同任务类型所请求服务种类的不同，提出了一种云边端协同的任务卸载模型。为了优化用户设备的执行成本，获取最优的任务卸载方案，提出了基于贪心策略的任务卸载算法。实验结果表明，该方案有效降低了用户设备的执行成本，且在处理资源较少或用户设备数量较多时，依然具有很好的性能。

本文提出的基于云边端协同的任务卸载方案虽然具有

较好的效果,但仍存在局限性。1)本文只对边缘服务器缓存服务是静态的情形进行了研究,没有考虑服务缓存的动态性。2)虽然云边端协同的卸载模型能满足任务在高计算能力、高存储能力和低时延服务等方面的需求,但本文没有考虑用户设备所产生的任务之间可能存在依赖关系。3)本文算法在面对边缘服务器和用户设备数量过多时,算法时间复杂度过高。因此,下一步的工作重点将尝试在任务卸载过程中,根据任务之间存在的依赖关系以及对服务请求的多样性,制定更优的任务卸载策略并动态更新边缘服务器缓存的服务,优化用户设备的计算时延和能耗成本。

参 考 文 献

[1] ALI S,ZHAO H P,KIM H. Mobile edge computing:A promising paradigm for future communication systems[C]// Proceedings of the 2018 IEEE Region 10 Conference. Jeju, Korea, 2018:1183-1187.

[2] ZHANG L,CHAI R,YANG T,et al. Min-max worst-case design for computation offloading in multi-user MEC system[C]// IEEE Conference on Computer Communications Workshops(INFOCOM 2020). Beijing,China,2020:1075-1080.

[3] ZHANG Y L,LIANG Y Z,YIN M J,et al. Survey on the Methods of Computation Offloading in Mobile Edge Computing[J]. Chinese Journal of Computers,2021,44(12):2406-2430.

[4] ABBAS N,ZHANG Y,TACHERKORDI A,et al. Mobile edge computing: A survey[J]. IEEE Internet of Things Journal, 2017,5(1):450-465.

[5] ZHANG H B,LI H,CHEN S X,et al. Computing Offloading and Resource Optimization in Ultra-dense Networks with Mobile Edge Computation[J]. Journal of Electronics & Information Technology,2019,41(5):1194-1201.

[6] ZHANG N,GUO S,DONG Y,et al. Joint task offloading and data caching in mobile edge computing networks[J]. Computer Networks,2020,182:107446.

[7] REN J,GAO L,YU J L,et al. Energy-Efficient Deep Learning Task Scheduling Strategy for Edge Device[J]. Chinese Journal of Computers,2020,43(3):440-452.

[8] WANG H,LIN Z,LV T. Energy and Delay Minimization of Partial Computing Offloading for D2D-Assisted MEC Systems [C]// Proceedings of the 2021 IEEE Wireless Communications and Networking Conference. Nanjing,China,2021:1-6.

[9] CHEN L,WU J,ZHANG J,et al. Dependency-Aware Computation Offloading for Mobile Edge Computing with Edge-Cloud Cooperation[J]. IEEE Transactions on Cloud Computing,2020, 10(4):2451-2468.

[10] NING Z,DONG P,KONG X,et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things[J]. IEEE Internet of Things Journal,2018, 6(3):4804-4814.

[11] ALFAKIH T,HASSAN M M,GUMAEI A,et al. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA[J]. IEEE Access, 2020,8:54074-54084.

[12] MAO Y,YOU C,ZHANG J,et al. Mobile edge computing:Survey and research outlook[J]. arXiv:1701.01090,2017.

[13] ZHANG X,LI Z,LAI C,et al. Joint Edge Server Placement and Service Placement in Mobile Edge Computing[J]. IEEE Internet of Things Journal,2021,9(13):11261-11274.

[14] BI S,HUANG L,ZHANG Y J A. Joint optimization of service caching placement and computation offloading in mobile edge computing systems[J]. IEEE Transactions on Wireless Communications,2020,19(7):4947-4963.

[15] GUO M,HUANG X,WANG W,et al. HAGP:A Heuristic Algorithm Based on Greedy Policy for Task Offloading with Reliability of MDs in MEC of the Industrial Internet[J]. Sensors, 2021,21(10):35

ZHANG Junna, born in 1979, Ph.D, associate professor, is a member of China Computer Federation. Her main research interests include edge computing and service computing.

CHEN Jiawei, born in 1998, postgraduate, is a member of China Computer Federation. His main research interests include edge computing and service computing.

(责任编辑:喻藜)