移动边缘计算中带有缓存机制的任务卸载策略

郭 煜
(广东国防科技技师学院 广东 广州 510515)

摘 要 为了满足延时敏感型应用执行的需求,实现移动设备的能耗优化,基于移动边缘计算环境提出一种融入缓存机制的任务卸载策略。与仅关注计算卸载决策不同,该策略可将已完成的重复请求任务及相关数据在边缘云上进行缓存,这样可以降低任务的卸载延时。将计算与存储能力受限的边缘云中的任务缓存与卸载优化决策问题分解为两个子优化问题进行求解。证明任务卸载子问题可转换为决策变量的凸最优化问题,而任务缓存子问题可转换为 0 - 1 整数规划问题。分别设计内点法和分支限界法对两个子问题进行求解,进而得到满足截止时间约束时能耗最优的卸载决策解。仿真算例证明了该策略在动态异构的任务执行环境下可以实现更好的能效优化。

关键词 移动边缘计算 任务缓存 任务卸载 能耗 整数规划

中图分类号 TP393 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.06.023

TASKS OFFLOADING STRATEGY WITH CACHING MECHANISM IN MOBILE MARGIN COMPUTING

Guo Yu
(*Guofang Science and Technology Technician Institute*, *Guangzhou 510515*, *Guangdong*, *China*)

Abstract In order to meet the requirements of delayed sensitive application execution and to realize the energy optimization of mobile devices consumption, I proposed a task offloading strategy with caching mechanism based on mobile margin computing environment. Different with offloading decision only focused on computation, the proposed strategy could cache the completed repeated tasks and related data in margin cloud, to reduce the tasks offloading delay. Further, the joint optimization problem of task caching and offloading on margin cloud with constrained computing and storage resource could be divided into two sub-problem. It was proved that the task offloading problem could be transferred to a convex optimization problem of the decision variable, and the task caching problem could be transferred to 0 - 1 integer programming problem. I designed interior point method and the branch bounding method respectively to solve these two sub-problems, which further achieved the offloading decision solution to get optimal energy consumption within certain time constraint. The simulation example proves that the strategy can achieve better energy efficiency optimization under dynamic and heterogeneous tasks execution condition.

Keywords Mobile margin computing Task caching Task offloading Energy consumption Integer programming

0 引 言

当前,随着无线通信技术和物联网技术的迅速发展,越来越多的移动设备(如智能手机、穿戴设备)对于带宽与计算能力有了更多的无线网络访问需求。未来的移动设备将变得更加智能,而部署于移动设备上的应用将需要更多的计算能力和数据访问[1]。然而,这些新型应用和服务的部署受到移动设备有限计算能力和电池容量的限制。虽然将数据饥饿型和计算密集型任务卸载至云端执行可以克服移动设备计算能力受限的弊端,但移动设备与云端间的无线网络连接带来的较长延时会导致延时敏感型应用的执行无法达到用户的满意[2]。

收稿日期:2018 - 12 - 21。国家重点研发计划项目(2017YFC0804301)。**郭煜**,高级讲师,主研领域:计算机网络,云计算。

近年来,移动边缘计算通过将计算节点部署于网络边缘的方式为用户满足延时敏感型任务的需求提供了低延时和高性能计算的保证[3,4]。边缘云具备的优势在于:1)比较移动设备的本地计算[5],移动边缘计算克服了移动设备的计算能力受限的不足;2)比较远程云端计算[6],尽管边缘云部署范围较小,且资源能力受限制,但可避免任务卸载至远程云端的高延时问题。因此,移动边缘计算可以在延时敏感和计算密集型任务的执行上得到更好的平衡。

当前移动边缘计算中的卸载问题主要包括:页面卸载,即边缘缓存,页面提供者将常用页面缓存于边缘云上,以降低用户请求页面时的延时和能耗[7-8]。相关研究中已有考虑页面分布[9]和用户移动性[10]的缓存策略。任务卸载,该问题即是决定何时、何地、多少任务应从移动设备卸载至边缘上执行,以降低计算延时和节省能耗。该类研究中主要集中于考虑多用户环境[11]和多服务器环境[12]下的卸载决策问题。页面卸载主要关心边缘云的存储能力,不同步考虑计算能力。而在任务卸载的相关研究中,是以边缘云具有足够的软硬件资源支持任务计算为常态假设的,这与边缘云资源受限以及无法支持所有类型的任务是相违背的。

本文将设计更加实际且注重能效的移动边缘计算卸载策略。首先引入任务缓存的概念,即将多次请求的任务及相关数据缓存于边缘云的操作。由于任务同时需要存储与计算资源,任务缓存需要同时考虑边缘云的计算和存储能力。在此基础上,设计基于边缘云的任务缓存与卸载联合优化策略,目的是通过最优的任务缓存与卸载决策,实现满足用户延时需求的情况下最小化移动设备总能耗达到最小。

1　系统模型

系统目标是在满足用户延时需求的同时最小化移动设备的总体能耗,系统模型如图 1 所示。

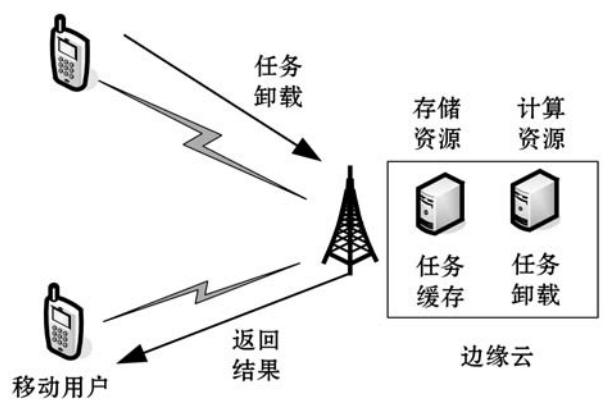


图 1　系统模型

移动设备拥有五个计算任务,每个任务拥有不同的请求次数(即任务分布不同)、数据量以及计算能力需求。以虚拟现实应用为例,其场景渲染任务拥有更多分布,会重复多次执行,而追踪目标任务则拥有更大的数据量,需要更多的数据传输,而目标识别任务则需要更高性能的计算资源。通过任务缓存和卸载可以在满足用户延时需求的同时降低移动设备的能耗。对于在考虑任务异构和边缘云资源受限的情况下实现最优的任务缓存和卸载,重点需要解决:1)哪些任务需要缓存,即决定是否任务需要缓存于边缘云上;2)多少任务需要卸载,即决定多少任务在本地执行,多少任务在边缘云上执行。

1.1　场景描述

移动边缘计算场景由多个移动设备和一个边缘云组成,假设移动设备(用户)数量为 N,计算任务数量为 K,用户 $n=\{1,2,\cdots,N\}$,任务 $k=\{1,2,\cdots,K\}$,且假设用户数量大于任务数量,即 $N\geqslant K$,这是由于某些计算任务具有更高的分布,使得这些任务需要重复请求并多次执行。假设任意用户一次仅能进行一个任务请求,且不同用户可根据其偏好进行同一任务的请求。定义 $u_{n,k}$ 表示用户 n 请求任务 k,移动设备通过无线信道与边缘云连接,边缘云为拥有计算和存储资源的小规模数据中心,计算资源为移动设备提供任务处理资源,存储资源为任务代码处理提供存储资源。

对于异质计算任务,利用三元组定义计算任务模型,将用户 $u_{n,k}$ 定义为 $u_{n,k}=\{w_k,s_k,D_n\}$,w_k 表示任务 $u_{n,k}$ 请求的计算资源数量,以每 bit 的 CPU 周期数表示,即完成该任务需要的 CPU 周期总数,s_k 表示任务 $u_{n,k}$ 的数据量,单位为 bit,D_n 表示用户 n 的任务完成截止时间。由于边缘云计算和存储能力受限,假设边缘云的缓存大小和计算能力分别为 c_e 和 c_s。

1.2　通信模型

令 H_n 表示用户 n 与边缘云间的信道增益,假设用户在将任务卸载至边缘云时不发生移动,则为常量。令 P_n 表示用户 n 的移动设备的发送功率,则用户 n 的上传数据速率可定义为:

$$r_n=B\log_2\left(1+\frac{P_nH_n}{\sigma^2}\right) \tag{1}$$

式中:σ^2 表示噪声功耗,B 表示移动设备与边缘云间的无线信道带宽。本文不考虑边缘云处理后的数据下载延时,由于处理后的数据量通常远小于任务处理前的数据量,且边缘云至移动设备的下载速率是远高于移动设备至边缘云的上传速率的。

1.3 计算模型

1) 本地任务计算的延时和能耗代价。

对于本地任务计算,定义 f_n^{l} 为移动用户 n 的 CPU 计算能力,则任务 $u_{n,k}$ 的本地执行时间为:

$$T_{n,k}^{\mathrm{l}}=\frac{w_k}{f_n^{\mathrm{l}}} \tag{2}$$

单个 CPU 计算周期的能耗可表示为:$\varepsilon=\kappa f^2$,则在本地执行时的能耗代价为:

$$E_{n,k}^{\mathrm{l}}=\kappa(f_n^{\mathrm{l}})^2 w_k \tag{3}$$

式中:κ 表示能量因子,大小取决于 CPU 芯片工艺,本文在实验中将该值设置为 10^{-25}。

2) 边缘云计算的延时和能耗代价。

对于边缘云上的任务计算,定义 f_n^{c} 为分配给用户 n 的 CPU 计算能力,此时任务执行延时包括:(1) 移动用户卸载任务的时间,即任务上传时间 $T_{n,k}^{\mathrm{tra}}$;(2) 边缘云执行任务的时间,即任务处理时间 $T_{n,k}^{\mathrm{pro}}$。因此,任务 $u_{n,k}$ 在边缘云上的处理延时为:

$$T_{n,k}^{\mathrm{c}}=T_{n,k}^{\mathrm{tra}}+T_{n,k}^{\mathrm{pro}}=\frac{s_k}{r_n}+\frac{w_k}{f_n^{\mathrm{c}}} \tag{4}$$

若任务卸载至边缘云,移动设备的能耗仅为任务卸载时间通信能耗,因此,发送 s_k bit 的任务至边缘云时的传输能耗代价为:

$$E_{n,k}^{\mathrm{c}}=P_n T_{n,k}^{\mathrm{tra}}=\frac{P_n s_k}{r_n} \tag{5}$$

1.4 任务缓存模型

任务缓存即将已完成任务及相关数据缓存在边缘云上,任务缓存过程如下:移动设备发出计算任务的卸载请求,若该任务已在边缘云缓存,则边缘云告之移动设备已在云端存储,移动设备无需进行任务卸载。边缘云完成任务处理后,直接将结果传输至移动设备。通过该方式,用户无需对已缓存的任务进行任务卸载,可以降低移动设备的能耗与任务卸载的延时。尽管边缘云的缓存能力和计算能力优于移动设备,但并非可以缓存和支持所有类型的计算任务,且任务缓存需要同时考虑任务分布、数据量以及任务请求资源大小。在任务缓存时,本文假设任务延时简化为任务处理延时 $T_{n,k}^{\mathrm{pro}}$,且主要能耗发生在边缘云上而非移动设备上。而没有任务缓存时,延时定义为 $T_{n,k}^{\mathrm{c}}$,移动设备能耗定义为 $E_{n,k}^{\mathrm{c}}$。

1.5 问题形式化

对于任务缓存问题,定义一个整型缓存决策变量 $x_k\in\{0,1\}$,表示任务 k 是否在边缘云缓存,若是,$x_k=1$;否则,$x_k=0$。因此,任务缓存策略可表示为:$x=(x_1,x_2,\cdots,x_k)$。对于任务卸载问题,定义卸载决策变量 $a_n\in[0,1]$,若 $a_n=1$,则用户 n 任务在本地执行,若 $a_n=0$,则用户 n 任务卸载至边缘云执行,若 $a_n\in(0,1)$,则用户 n 的任务部分 a_n 在本地执行,任务部分 $1-a_n$ 在边缘云执行。因此,任务卸载策略可表示为:$a=(a_1,a_2,\cdots,a_n)$。

基于以上定义,考虑任务缓存、任务的本地和边缘云执行,用户 n 任务 $ku_{n,k}$ 的总执行延时为:

$$T_{n,k}=x_k\frac{w_k}{f_n^{\mathrm{c}}}+(1-x_k)\left[a_n T_{n,k}^{\mathrm{l}}+(1-a_n)T_{n,k}^{\mathrm{c}}\right] \tag{6}$$

移动设备的能耗代价为:

$$E_{n,k}=(1-x_k)\left[a_n E_{n,k}^{\mathrm{l}}+(1-a_n)E_{n,k}^{\mathrm{c}}\right] \tag{7}$$

算法目标是在确保服务质量的同时最小化移动设备的能耗代价,则问题可形式化为:

$$\begin{aligned}
&\min_{x,a}\sum_{n=1}^{N}E_{n,k}\\
&\text{s.t.}\quad \mathrm{C1}:\sum_{k=1}^{K}x_k s_k\leqslant c_e\\
&\qquad\ \mathrm{C2}:\sum_{n=1}^{N}a_n f_n^{\mathrm{c}}\leqslant c_s\\
&\qquad\ \mathrm{C3}:T_{n,k}\leqslant D_n\quad\forall n\in N,k\in K\\
&\qquad\ \mathrm{C4}:x_k\in\{0,1\}\quad\forall k\in K\\
&\qquad\ \mathrm{C5}:a_n\in[0,1]\quad\forall n\in N
\end{aligned} \tag{8}$$

式中:目标函数表明通过任务的缓存与卸载决策对移动设备的总能耗代价最小化,约束 C1 表明缓存任务的数据量不能大于边缘云的缓存能力,约束 C2 表明卸载任务的总的资源请求不能大于边缘云的计算能力,约束 C3 表明用户 n 的任务执行需在截止时间前完成,约束 C4 表明任务缓存决策变量为二进制变量,约束 C5 表明任务是可分割的,可部分在本地执行,部分在边缘云上执行。

2 能效任务缓存与卸载

本节求解以上的任务缓存与卸载决策最优化问题,考虑两个子优化问题:1) 任务卸载问题:当给定任务缓存策略,即 $x=x^0$ 时,初始问题可转变为关于 a 的凸最优化问题,可以通过内点法获得最优解 a^*。2) 任务缓存问题:当 a^* 固定时,该子优化问题可转变为 0-1 整数规划问题,可利用分支限界法获得问题的最优解。以下将分别证明以上的结论。

1) 能效任务卸载。

定理:给定 $x=x^0$,式(8)中关于 a 的初始最优化问题为凸最优化问题。

证明:给定 $x=x^0$,目标函数变为 a 的函数。由于

目标函数中的 $x_k T_n^{\text{pro}}$,k 与 a 无关,因此,可定义关于 a 的目标函数为 $f(a)$,表示为:

$$f(a) = \sum_{n=1}^{N} (1 - x_k^0)[a_n E_{n,k}^{\text{l}} + (1 - a_n) E_{n,k}^{\text{c}}] \quad (9)$$

进一步,式(8)关于 a 的最优化问题可重写为:

$$\begin{aligned} &\min_{a} f(a) \\ &\text{s.t.} \quad \text{C1}: \sum_{n=1}^{N} a_n f_n^{\text{c}} \leqslant c_s \\ &\qquad \text{C2}: T_{n,k} \leqslant D_n \quad \forall n \in N, k \in K \\ &\qquad \text{C3}: a_n \in [0,1] \quad \forall n \in N \end{aligned} \quad (10)$$

其中,目标函数表示给定任务缓存策略时通过制定最优任务卸载决策得到能耗的最小化,约束 C1 表明卸载任务的总的资源请求不能大于边缘云的计算能力,约束 C2 表明用户 n 的任务执行需在截止时间内完成,约束 C3 表明任务是可分割的,可部分在本地执行,部分在边缘云上执行,a_n 本身是任务在本地执行的比例。

由于目标函数是线性的,且约束条件也均是线性的,故该最优化问题为凸最优化问题。证毕。

凸最优化问题即可利用内点法得到最优任务卸载策略 a^*,即通过求解一系列 KKT 条件的修改形式,在等式约束下得到最优解,求解方法可具体参见文献[13]。

2) 能效任务缓存。

根据以上的讨论,可能得到最优任务卸载策略 a^*。当 $a = a^*$ 时,目标函数可转变为 x 的函数,可表示为:

$$g(x) = \sum_{n=1}^{N} (1 - x_k)[a_n^* E_{n,k}^{\text{l}} + (1 - a_n^*) E_{n,k}^{\text{c}}] \quad (11)$$

进一步,式(8)关于 x 的最优化问题可表示为:

$$\begin{aligned} &\min_{x} g(x) \\ &\text{s.t.} \quad \text{C1}: \sum_{n=1}^{N} a_n f_n^{\text{c}} \leqslant c_s \\ &\qquad \text{C2}: T_{n,k} \leqslant D_n \quad \forall n \in N, k \in K \\ &\qquad \text{C3}: x_k \in \{0,1\} \quad \forall k \in K \end{aligned} \quad (12)$$

其中,目标函数表示给定任务缓存卸载策略时通过制定最优任务缓存决策得到能耗的最小化,约束 C1 表明卸载任务的总的资源请求不能大于边缘云的计算能力,约束 C2 表明用户 n 的任务执行需在截止时间内完成,约束 C3 表明任务缓存决策变量为二进制变量,只有任务缓存或不进行缓存两种选择。

因此,目标函数可转变为关于 x 的 0 - 1 线性规划问题,利用分支限界法即可得到其最优解,具体过程如下:

(1) 分支步骤。该步骤中,计算任务集合被转换为一棵树,代表所有任务在本地或需要缓存的可能性组合。该树由空节点的树根开始构建,对于该父节点,每个任务的副本被连接为孩子节点,每个孩子节点代表在本地执行或云端缓存(标记为 local 或云端服务器编号)。类似地,下一任务节点的副本与树结构的每一层上与孩子节点相连。重复以上过程直到所有任务被添加至树结构中,该树的每条分支即代表一个侯选的任务缓存决策解。

为了减少决策时间,将节点添加至树结构中时,需要根据任务的能耗代价降序进行添加,这样得到的树中主分支将拥有更高的代价,这将使得分支步骤中可以提前遍历对代价影响更大的节点,从而尽可能早地发现不可行的高代价分支,有利于修剪分支状态,降低决策空间的搜索时间。

(2) 搜索步骤。该步骤由搜索分支步骤构建的树、计算每条分支的代价和寻找最小代价分支组成。为了加快决策时间,搜索过程利用一个限界函数修剪树结构中的某些非最优的分支。每条分支利用深度优先规则 DFS 计算其代价。同时,若分支的局部代价超过限界函数定义的限制值则修剪该分支。设计的限界函数以完全的本地执行或云端执行得到的能耗代价进行初始化。若一个分支被完整搜索而未被修剪,则将其考虑为当前的最优解,且其代价作为新的限界函数值。当所有分支被搜索后,算法返回最优解。

3　仿真实验

3.1　实验配置

考虑一个由边缘云和移动设备组成的移动边缘计算系统,移动设备拥有计算密集型任务和数据密集型任务执行需求,假设边缘云部署于无线访问点 AP 的附近,移动设备可通过无线信道将任务卸载至边缘云执行。假设无线信道带宽 $B = 20$ MHz,移动设备的发送功率 $P_n = 0.5$ W,对应的噪声功耗 $\sigma^2 = 2 \times 10^{-13}$,无线信道增益 $H_n = 127 + 30 \times \log d$,$d$ 表示用户 n 与边缘云间的距离。对于任务 $u_{n,k}$,假设请求的计算能力 w_k 和数据量 s_k 由概率分布产生,分别服从正态分布和均匀分布。对于任务的分布,假设任务请求次数服从 Zipf 分布,同时,设置边缘云和移动设备的计算能力分别为 25 GHz 和 1 GHz。其他相关参数中,$n = 100$,$c_e = 500$ MB,w 服从均值为 0.8(单个任务的 CPU 周期数)的均匀分布,s 服从均值为 100 MB 的均匀分布。利用 MATLAB 进行实验的验证分析。

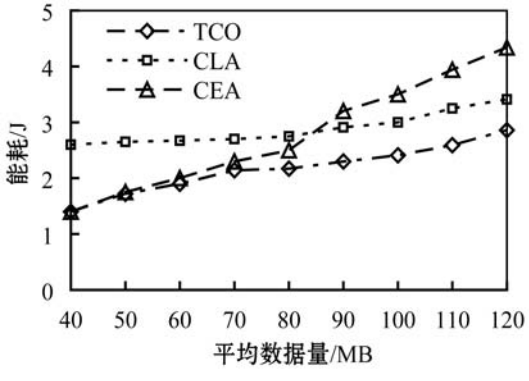
3.2 结果分析

1）任务卸载影响 将本文提出的基于能效的任务缓存与卸载算法命名为 TCO 算法，并与以下算法进行性能比较：

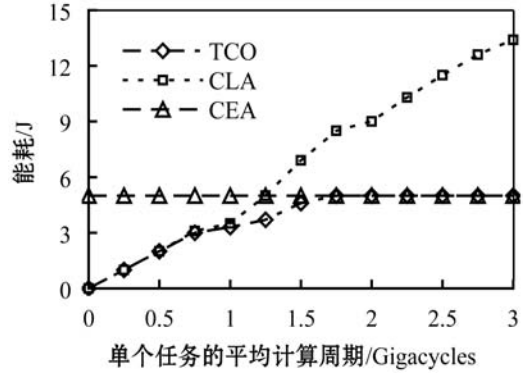
缓存 + 本地执行算法，简称 CLA。该算法根据本文提出的缓存策略对计算任务进行缓存，然后，未被缓存的任务仅在本地设备上执行，不被卸载至边缘云处理。

缓存 + 边缘云执行算法，简称 CEA。与 CLA 不同，该算法中未缓存的任务全部卸载至边缘云处理。

从图 2 看出，TCO 算法得到的能耗最低，说明应用最优的缓存部署和任务卸载可以有效降低移动设备的能耗。图 2(a)中，当任务数据量较小时，TCO 和 CEA 的能耗差异较小，而任务数据量较大时，TCO 和 CLA 的能耗差异较小。可以得出，在相同的任务计算能力请求下，当任务数据量较小时，任务应在边缘云上处理更为有效。比较来说，若任务数据量较大，则任务应在本地执行较好。从图 2(b)可以得出，在相同的任务数据量下，当任务计算能力请求相对较小时，任务应在本地执行较好，而计算能力请求较大时，则在边缘云执行较优。



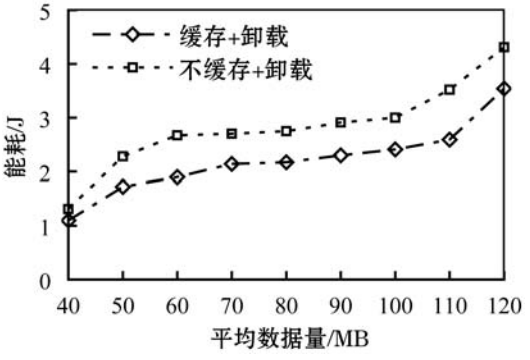
(a)任务数据量变化



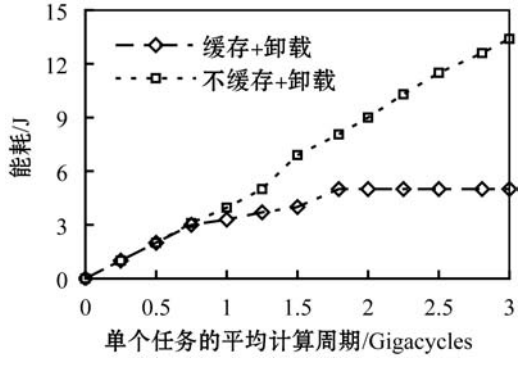
(b) 任务请求计算能力变化

图 2 任务卸载的影响

2）任务缓存影响 观察两种情况下的能耗状况：任务不被缓存和任务缓存，如图 3 所示。当任务被缓存时，移动设备的能耗低于不缓存时的能耗，说明计算任务缓存可以降低能耗。还可以看出，任务数据量越大，计算能力请求越高，能耗将越高。



(a) 任务数据量变化



(b) 任务请求计算能力变化

图 3 任务缓存影响

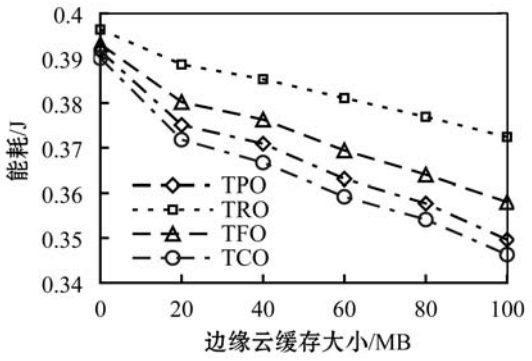
为了评估缓存策略对结果的影响，将 TCO 与以下算法进行比较：

任务尽力缓存与卸载算法，简称 TPO。对于边缘云，它将最大限度地缓存计算任务直到达到边缘云的缓存能力。

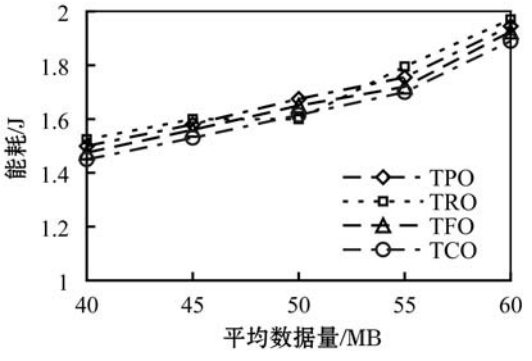
任务随机缓存与卸载算法，简称 TRO。边缘云随机地进行任务缓存，直到达到边缘云的缓存能力。

任务有限缓存与卸载算法，简称 TFO。边缘云的缓存能力初始设置为空，迭代式增加一个任务至缓存以最小化总能耗，直到达到边缘云的缓存能力。

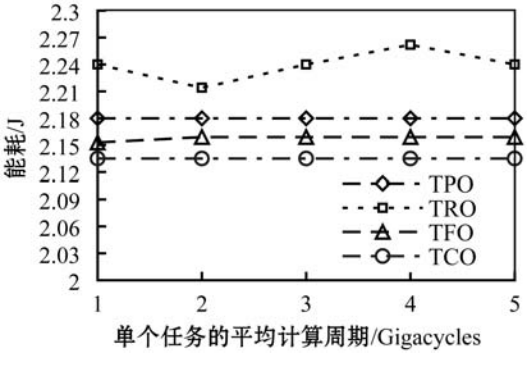
由图 4 可知，TCO 是最优的，TRO 相对较差，这是由于随机缓存策略没有考虑任务请求数量的原因导致，且该算法在进行任务缓存时也无法考虑任务计算量和计算任务的数据量。TPO 仅考虑了任务请求数量，没有考虑任务数据量和任务计算量，TFO 在一定程度上考虑了以上三个量。从图 4(a)还可以看到，边缘云的缓存能力越大，能耗将越小，这是由于缓存能力越大，缓存任务越多，能耗将降低。从图 4(b)和(c)可以看出，任务数据量对算法的影响要弱于计算能力对算法的影响。



(a) 边缘云缓存大小变化



(b) 任务数据量变化



(c) 任务请求计算能力变化

图 4　任务缓存的影响

4　结　语

移动边缘计算环境中,延时敏感型应用任务的执行不仅需要满足用户时延的约束,还需要考虑移动设备端的能耗问题。为了解决这一问题,基于移动边缘计算环境提出了一种融入任务缓存机制的任务卸载策略。不同于常规的边缘计算仅仅关注计算卸载决策问题,该策略设计了一种基于边缘云的任务缓存机制,降低了重复的任务需求执行时的卸载延时。此外,策略将计算与存储能力受限的边缘云中的任务缓存与卸载优化决策问题形式化为混合整数规划问题,并对其进行了求解。实验结果证明,带有缓存机制的任务卸载策略在动态异构的任务执行环境下可以实现更好的能效优化。进一步的研究将集中于面向多个边缘云的任务卸载环境,在任务缓存与卸载的边缘云资源的选择上作出优化。

参　考　文　献

[1] Chen M, Qian Y, Hao Y, et al. Data-Driven Computing and Caching in 5G Networks: Architecture and Delay Analysis [J]. IEEE Wireless Communications, 2018, 25(1):70-75.

[2] 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.

[3] Wang S, Zhang X, Zhang Y, et al. A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications[J]. IEEE Access, 2017, 5:6757-6779.

[4] Zhang K, Mao Y, Leng S, et al. Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks [J]. IEEE Access, 2016, 4:5896-5907.

[5] Chen M, Hao Y, Li Y, et al. On the computation offloading at ad hoc cloudlet: architecture and service modes [J]. IEEE Communications Magazine, 2015, 53(6):18-24.

[6] Barbera M V, Kosta S, Mei A, et al. To offload or not to offload? The bandwidth and energy costs of mobile cloud computing[C]//IEEE INFOCOM. IEEE, 2013:1285-1293.

[7] Wang X, Wang H, Li K, et al. Serendipity of Sharing: Large-Scale Measurement and Analytics for Device-to-Device (D2D) Content Sharing in Mobile Social Networks [C]// IEEE International Conference on Sensing, Communication and NETWORKING. IEEE, 2017:1-9.

[8] Li X, Wang X, Li K, et al. CaaS: Caching as a Service for 5G Networks[J]. IEEE Access, 2017, 5:5982-5993.

[9] Wang X, Zhang Y, Leung V C M, et al. D2D Big Data: Content Deliveries over Wireless Device-to-Device Sharing in Large-Scale Mobile Networks[J]. IEEE Wireless Communications, 2018, 25(1):32-38.

[10] Chen M, Hao Y, Hu L, et al. Green and Mobility-aware Caching in 5G Networks[J]. IEEE Transactions on Wireless Communications, 2017, 16(12):8347-8361.

[11] Chen X, Jiao L, Li W, et al. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing [J]. IEEE/ACM Transactions on Networking, 2015, 24(5):2795-2808.

[12] Min C, Hao Y, Qiu M, et al. Mobility-Aware Caching and Computation Offloading in 5G Ultra-Dense Cellular Networks:[J]. Sensors, 2016, 16(7):974.

[13] Necoara I, Suykens J A K. Interior-Point Lagrangian Decomposition Method for Separable Convex Optimization [J]. Journal of Optimization Theory & Applications, 2009, 143(3):567.