多层次算力网络集中式不可分割任务调度算法

Centralized Unsplittable Task Scheduling Algorithm for Multi-Tier Computing Power Network

巩宸宇/GONG Chenyu[1]，舒洪峰/SHU Hongfeng[2]，张昕/ZHANG Xin[2]
(1. 上海科技大学，中国 上海 200120；
2. 深圳市智慧城市科技发展集团有限公司，中国 深圳 518046)
(1. ShanghaiTech University, Shanghai 200120, China;
2. Smart Cities Group, Shenzhen 518046, China)

摘要：根据算力网络不同层次的特性和各种应用的不同需求，提出一种多层次算力网络模型和计算卸载系统，并定义一个由时延、能耗组成的加权代价函数以建模一个任务调度问题。为解决这一问题，提出一个基于交叉熵的集中式不可分割任务调度(CUTS)算法。数值仿真结果表明，与其他基线算法相比，该算法在系统平均代价方面拥有较好的性能。

关键词：多层次算力网络；交叉熵；集中式；任务调度；不可分割

Abstract: According to the characteristics of different layers of computing power network and different requirements of various applications, a multi-tier computing power network model and computation offloading system are proposed. Specifically, a cost function consisting of latency and energy consumption to model a task scheduling problem is defined. To solve the problem, a centralized unsplittable task scheduling (CUTS) algorithm based on cross-entropy is introduced. Simulation results show that the algorithm provides superior performance in terms of the average system cost compared with other baseline solutions.

Keywords: multi-tier computing power network; cross-entropy; centralized; task scheduling; unsplittable

DOI：10.12142/ZTETJ.202103008
网络出版地址：https://kns.cnki.net/kcms/detail/34.1228.TN.20210617.1144.010.html

网络出版日期：2021-06-17
收稿日期：2021-05-13

近年来，随着深度学习的不断发展，人工智能服务和应用大量涌现，比如人脸识别、自然语言处理、虚拟现实、增强现实等。这些应用通常都是计算密集型任务，将消耗大量的终端资源(如算力和能耗)。然而，由于计算能力和能量供应有限，终端设备(例如手机)可能无法提供良好的服务质量。为此，研究者们提出云计算的概念。

云计算[1-2]是由分布式计算、并行处理、网格计算发展而来的新型计算模型。通过虚拟化技术建立强大的资源池，云计算使各种应用和服务能够按需获取算力、存储资源及各种软件资源。云计算为海量数据的处理提供了可能，同时也为计算密集型的人工智能应用提供了强大的算力。然而，端与云之间的传输时延使得云计算无法满足时延敏感型应用的需求。因此，雾计算和边缘计算[3-4]的概念被提出，以解决云计算传播时延大的问题。

边缘计算是指，在靠近物或者数据源头的一侧部署设备，提供计算、存储等软件服务，并通过算力和通信资源的联合分配，满足应用的时延需求。经典的边缘计算网络由雾节点和本地用户共同组成。其中，本地用户通过任务拆分和任务卸载决策，来达到全局时延和能耗最小的最优效果。此前，学者们的研究主要集中在单用户多节点[5-6]和多用户单节点[7-8]。文献[9]研究了多用户多节点这一应用场景。研究表明，边缘计算可以降低传输时延。但是对于一些对算力和时延都有较高要求的应用来说，边缘计算网络将不再适用，比如自动驾驶、虚拟现实等。因此，算力网络[10]

的概念被提出。

算力网络涉及云计算、雾计算、边缘计算等。算力网络是由云边端等设备构成的多层次资源网络，它能够将云边端进行统一调配，但是如何实现系统的最优性能仍是一个难题。原因主要有两点：(1)云边端各有其特性。云距离端较远但算力强，多用于处理全局任务；边距离端较近但算力弱，多处理本地实时任务。(2)用户任务的需求不同。计算密集型任务可能更多地需要云的参与，时延敏感型任务可能更多地需要边的参与，对算力和时延同时有较高要求的任务则需要联合进行调度。基于以上原因，文献[11]研究了多层次算力网络，并提出一种分布式调度算法。但是该模型是边端混合的两层算力网络，并未考虑云的作用。

试想存在如下场景：一座办公大楼内有多个楼层，每层都有多间办公室，且每间办公室都有多个用户和不同性质的任务。由于职能划分不同，不同部门通常所需要的算力不尽相同。这就容易造成算力资源的不合理利用，甚至造成任务中断。如果我们按照办公室和楼层的位置，将其构造成一个多层次算力网络，进行任务的调度和算力分配，那么就能够更好地满足计算密集型和时延敏感型应用的需求。

1 计算卸载系统建模

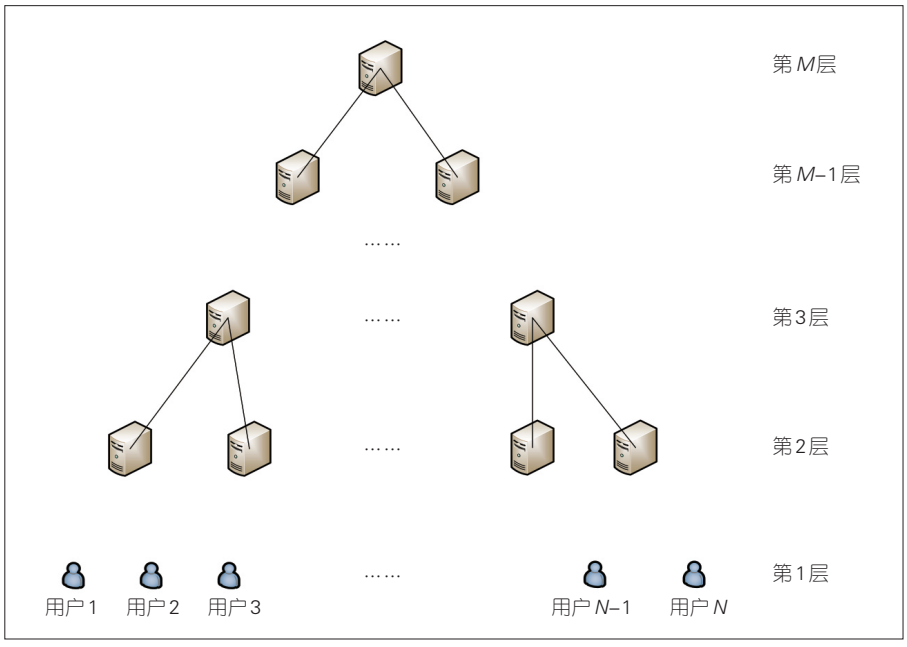
1.1 系统概述

本节将详细介绍一个多层次算力网络和计算卸载系统，定义一个由时延、能耗组成的加权代价函数，并建模一个任务调度问题。

算力网络一共有多层。第1层为用户节点，其他层为雾节点。雾节点的算力随层数的增加而上升。通常，距离用户较远的高层雾节点算力比较强大，但是往返时延较长；距离用户较近的低层雾节点往返时延较短，但是算力有限。在考虑时延和能耗的基础上，用户可以将不可拆分的任务卸载到某层的某个雾节点，也可以选择将任务在本地执行。因此，如何根据时延和能耗帮助用户做出卸载决策，是解决任务调度问题并获取全局最优解的核心。

如图1所示，算力网络一共有M层，第1层包含N个用户，每个用户都有一个任务。用户集合$\mathcal{N}=\{1,2,\cdots,N\}$也可以看作任务集合。此处，我们假设每个任务都不可拆分，即如果用户选择把任务卸载到雾节点，那么只能卸载到某一个雾节点上。我们将用户$n\in\mathcal{N}$的计算任务表示为(z_n,γ_n)，其中，z_n是任务的大小(位)，γ_n是任务的处理密度(中央处理器转数/位，cycle/b)，即处理单位比特数据所需要的(CPU)中央处理器转数。第2层至第M层是雾节点。其中，低层雾节点和高层雾节点均为汇聚式连接方式，即第$m-1$层的某几个雾节点向上汇聚并连接第m层的某一个雾节点。如果用户选择将任务卸载到第m层($m>2$)，因为存在汇聚式的连接方式，当第2层中转雾节点确定后，用户任务卸载到第m层的节点是确定的。因此，用户在3层以上的决策空间为层数。如果第2层的雾节点有K_2个，那么第2层的雾节点可以表示为$\mathcal{K}_2=\{1,2,\cdots,K_2\}$，第$m$层的雾节点表示为$L_m(m>2)$，所有雾节点的集合表示为$\mathcal{K}=\mathcal{K}_2\cup L$，且$L=\{L_3,L_4,\cdots,L_M\}$。用户$n$的卸载策略$a_n\in\{0\}\cup\mathcal{K}$，可以表示为：若$a_n=0$，则用户选择在本地运行任务；若$a_n\in\mathcal{K}$，则用户选择将任务卸载到雾节点上。对于所有用户的算法调度策略，我们用$A=\{a_1,a_2,\cdots,a_n\}$来表示。所有选择雾节点$k\in\mathcal{K}$的用户数$n_k(A)=\sum_{n=1}^{N}I_{\{a_n=k\}}$，其中$I_{\{x\}}$是指示函数。如果$x$为真，则$I_{\{x\}}=1$，否则$I_{\{x\}}=0$。

用户与第2层雾节点之间为全连接式结构。多层次算力网络不考虑



▲图1 多层次算力网络模型

隔层直接通信。

1.2 通信模型

每个用户都有两种卸载决策。当用户决定将任务在本地执行时，此时并没有通信产生的时延；而当用户决定将任务卸载到雾节点时，就会存在通信时延。

假设雾节点k只有一个用户n接入，则从用户n到雾节点k的最大传输速率表示为$R_{n,k}$[12]。但是一个雾节点往往会有多个用户接入，此时各个用户需要竞争来获得通信带宽。根据共享模型，我们将带宽进行均分。当有多个用户接入节点k时，用户n的传输速率表示为公式(1)：

$$R_{n,k}(A)=\frac{R_{n,k}}{n_k(A)}。\tag{1}$$

与大多数工作一样，与卸载到雾节点的任务大小相比，由于从雾节点返回用户的结果太小，并且下行速率要比上行速率大得多，因此，结果返回产生的通信时延忽略不计。用户n将任务卸载到第3层雾节点k上的通信时延如公式(2)所示：

$$T_{\text{trans},n,(2,k)}=\frac{z_n}{R_{n,k}(A)}=\frac{z_n n_k(A)}{R_{n,k}}。\tag{2}$$

如果用户将任务卸载到第m层($m>2$)雾节点，首先用户需要将任务卸载到第2层雾节点，然后通过有线网将任务向上传输。假设任务在上传时经过的第2层雾节点为q^*，则

$$q^*=\arg\min_q\left\{T_{\text{trans},n,(2,q)};q=1,2,\cdots,K_2\right\}。\tag{3}$$

我们假设相邻层雾节点之间的往返传输时间为常数，并用t_c表示。那么，用户将任务卸载到第m层的通信时延可用公式(4)表示：

$$T_{\text{trans},n,(m,k)}=T_{\text{trans},n,(2,q^*)}+(m-2)t_c。\tag{4}$$

综上所述，用户n卸载到雾节点的通信时延如公式(5)所示：

$$T_{\text{trans},n,(m,k)}=T_{\text{trans},n,(2,k)}I_{\left\{a_n\in\mathcal{K}_2\right\}}+\sum_{m=3}^{M}T_{\text{trans},n,(m,k)}I_{\left\{a_n=L_m\right\}}。\tag{5}$$

因为用户的通信能耗只存在与第2层雾节点的通信过程中，所以用户n将任务卸载到雾节点k上的能耗可用公式(6)表示：

$$E_{\text{trans},n,(m,k)}=P_{\text{fog},n}T_{\text{trans},n,(2,q^*)}=P_{\text{fog},n}\frac{z_n n_k(A)}{R_{n,q^*}},\tag{6}$$

这里，$P_{\text{fog},n}$表示用户n向雾节点k发送任务时的发送功率。我们假设这一功率是恒定的。

1.3 计算模型

1.3.1 本地计算模型

当用户决定在本地执行任务时，则存在计算时延和计算能耗。我们将用户n的计算能力表示为f_n，即CPU的时钟频率(CPU转数/s)，那么本地计算时延可以用公式(7)来表示：

$$T_{\text{comp},n}=\frac{z_n\gamma_n}{f_n}。\tag{7}$$

相应地，本地计算能耗可以用公式(8)表示：

$$E_{\text{comp},n}=\kappa_n z_n\gamma_n f_n^2,\tag{8}$$

其中，κ_n是与硬件结构相关的常数。

1.3.2 雾节点计算模型

假设同一层雾节点的算力相等，并且算力随着层数的增加而增加，第$m+1$层的算力是第m层算力的两倍。如果第2层雾节点k所能提供的算力用f_2表示，那么第m层雾节点k的算力为$f_m=2^{m-2}f_2$。当用户决定把任务卸载到第m层雾节点时，如果雾节点k只有一个用户n接入，那么雾节点k的算力将完全由用户n使用。当有多个用户接入雾节点k时，多个用户需要竞争雾节点有限的算力资源。我们简单地将雾节点的算力均分给每个用户，则用户n此时的计算时延如公式(9)所示：

$$T_{\text{comp},n,(m,k)}=\frac{z_n\gamma_n}{f_m}=\frac{z_n\gamma_n n_{k(A)}}{2^{m-2}f_2}。\tag{9}$$

因为任务是在雾节点上运行的，所以对于用户n来说并没有计算能耗。

1.4 问题建模

我们将用户的任务卸载代价建模成时延和能耗的线性组合，即定义一个加权代价函数。我们用α、$(1-\alpha)$分别表示时延和能耗的权重，并且规定$0\leqslant\alpha\leqslant1$。对于不同用户，权重取值不同。时延较敏感用户的$\alpha$取值应该更大，而能耗敏感用户的$\alpha$取值则应该更小。

如果用户选择在本地处理任务，那么代价可以用公式(10)表示：

$$C_n=\alpha T_{\text{comp},n}+(1-\alpha)E_{\text{comp},n}=\alpha\frac{z_n\gamma_n}{f_n}+(1-\alpha)\kappa_n z_n\gamma_n f_n^2。\tag{10}$$

如果用户选择将任务卸载到雾节点k上，代价可以用公式(11)来表示：

$$C_{n,(m,k)}=\alpha\left(T_{\text{trans},n,(m,k)}+T_{\text{comp},n,(m,k)}\right)+(1-\alpha)E_{\text{trans},n,(m,k)}。\tag{11}$$

给定策略组合$A=\left\{a_1,a_2,\cdots,a_n\right\}$，则用户$n$在该策略组合下的代价函数

如公式(12)所示：

$$C_n(A)=C_nI_{\{a_n=0\}}+\sum_{m=2}^{M}C_{n,(m,k)}I_{\{a_n\in\mathcal{K}\}}。\tag{12}$$

每个用户需要做出决策来最小化自己的代价函数。这里，我们将问题建模为最小化所有用户的平均代价函数，如公式(13)所示：

$$\min_{A}\frac{1}{N}\sum_{n\in\mathcal{N}}C_n(A)。\tag{13}$$

我们将公式(13)称为任务调度问题，并提出集中式不可分割任务调度算法来解决该问题。

2 基于交叉熵的集中式不可分割任务调度算法

交叉熵方法最初是用来解决稀有事件估计问题的，其基本思想是：通过不断迭代来修正稀有事件的发生概率，使得修正后的概率不断增大，直到此概率达到收敛。收敛概率即为最优概率。根据最优概率就一定会获得最优解。交叉熵方法后来逐渐发展成为解决优化问题(尤其是组合优化问题)的一种方法。这里，我们将使用交叉熵方法来分析任务调度问题。

我们定义所有用户的平均代价函数，如公式(14)所示：

$$Q(A)=\frac{1}{N}\sum_{n\in\mathcal{N}}C_n(A)。\tag{14}$$

假设$\mathcal{A}$是A的取值空间，γ^*是$Q(A)$的最小值，此时，公式(14)所示的问题就可以转化为对公式(15)的求解。

$$\gamma^*=\min_{A\in\mathcal{A}}Q(A)。\tag{15}$$

假设所有层的雾节点一共有T个，我们将所有用户的概率矩阵表示为P，维度表示为$n\times(T+1)$。元素$p_{i,j}$表示用户i选择将任务卸载到设备j的概率，$j\in\{0\}\cup\{1,2,\cdots,T\}$。其中，$j\in\{0\}$和$j\in\{1,2,\cdots,T\}$分别对应将任务在本地执行和卸载到各层的雾节点$\{\mathcal{K}_2,L_3,\cdots,L_m\}$上执行。我们可以看出，$\sum_j p_{i,j}=1$。概率矩阵$P$的取值空间为$\mathcal{P}$。我们定义一个关于卸载策略$A$的概率密度函数$\{f(\cdot;P\in\mathcal{P})\}$。由此我们构建一个估计问题，如公式(16)所示：

$$l(\gamma)=\mathbb{P}_V(Q(A)\leqslant\gamma)=\mathbb{E}_VI_{\{Q(A)\leqslant\gamma\}},\tag{16}$$

其中，A是符合概率密度函数$\{f(\cdot;V\in\mathcal{P})\}$的一个随机决策向量，$\gamma$是一个常数。

对于这个估计问题，$\{Q(A)\leqslant\gamma\}$是个小概率事件。我们要根据l的取值来使γ逐渐接近最优解γ^*。我们通过交叉熵方法来不断迭代概率密度函数，从而使概率矩阵也会发生变化，以此来获得最优解。理论上，迭代过程中产生的$f(\cdot;V),f(\cdot;P^1),f(\cdot;P^2),\cdots,f(\cdot;P^T)$都向着最优概率密度方向更新，从而使最优解可以获得。

2.1 更新γ^t

对于固定的P^{t-1}，我们使γ^t为在概率矩阵P^{t-1}下$Q(A)$的$(1-\rho)$分位数，如公式(17)所示：

$$\begin{aligned}&\mathbb{P}_{P^{t-1}}(Q(A)\leqslant\gamma^t)\geqslant\rho\\&\mathbb{P}_{P^{t-1}}(Q(A)\geqslant\gamma^t)\geqslant1-\rho,\end{aligned}\tag{17}$$

其中$A\sim f(\cdot;P^{t-1})$。

根据概率密度函数$f(\cdot;P^{t-1})$，我们随机抽取N_s个样本$\{a^1,a^2,a^3,\cdots,a^{N_s}\}$，然后计算出$\{Q(a^1),Q(a^2),Q(a^3),\cdots,Q(a^{N_s})\}$的值，并将其按降序排序，那么$\gamma^t$的估计值$\hat{\gamma}^t$就可以用公式(18)表示。

$$\hat{\gamma}^t=S_{\{\lceil(1-\rho)N_s\rceil\}}。\tag{18}$$

2.2 更新P^t

对于固定的γ^t和P^{t-1}，我们通过最小化交叉熵来得到P^t，相应的求解如公式(19)所示：

$$\max_P D(P)=\max_P\mathbb{E}_{P^{t-1}}I_{\{Q(A)\leqslant\gamma^t\}}\ln f(A;P)。\tag{19}$$

$f(\cdot;P)$的对数表示如公式(20)所示：

$$\ln f(A;P)=\sum_{i=1}^{N}\sum_{j=0}^{T}I_{\{a_i=j\}}\ln p_{ij}。\tag{20}$$

利用公式(20)，我们可以得到公式(19)的拉格朗日函数，如公式(21)所示：

$$\mathcal{L}(P,\lambda)=\mathbb{E}_{P^{t-1}}I_{\{Q(A)\leqslant\gamma^t\}}\sum_{i=1}^{N}\sum_{j=0}^{T}I_{\{a_i=j\}}\ln p_{ij}+\sum_{i=1}^{N}\lambda_i\left(\sum_{j=0}^{T}p_{ij}-1\right),\tag{21}$$

其中λ_i为拉格朗日乘子，$i=1,2,\cdots,N$。应用KKT(Karush－Kuhn－Tucker)条件，并通过$\partial\mathcal{L}(P,\lambda)/\partial p_{ij}=0$，我们可以得到公式(19)的最优解，如公式(22)所示：

$$p_{ij}=\frac{\mathbb{E}_{P^{t-1}}I_{\{Q(A)\leqslant\gamma^t\}}I_{\{a_i=j\}}}{\mathbb{E}_{P^{t-1}}I_{\{Q(A)\leqslant\gamma^t\}}}。\tag{22}$$

公式(23)可以被用来估计$\hat{p}_{ij}$：

$$\hat{p}_{ij}=\frac{\sum_{k=1}^{N_s}I_{\{Q(a^k)\leqslant\gamma^t\}}I_{\{a_i^k=j\}}}{\sum_{k=1}^{N_s}I_{\{Q(a^k)\leqslant\gamma^t\}}}。\tag{23}$$

但是通常来说，我们并不通过公

式(23)来直接优化P^t,而是通过公式(24)来进行优化:

$$\hat{p}_{i,j}^{t}=\beta\hat{p}_{i,j}^{t}+(1-\beta)\hat{p}_{i,j}^{t-1}, \quad (24)$$

其中,$\hat{p}_{i,j}^{t}$可从公式(23)中获得,β是平滑参数,且$\beta\in(0,1]$。

2.3 算法

下面我们将给出具体算法。

Algorithm 1 基于交叉熵方法的集中式不可分割任务调度(CUTS)算法

1: 初始化:
2: $p_{i,j}^{0}=1/(T+1)$, $\forall i\in\mathcal{N},j\in\{0\}\cup\{1,2,\cdots,T\}$
3: $t=1,N_s,\rho,d$
4: 初始化结束
5: 重复执行以下步骤
6: 根据$f(\cdot;P^{t-1})$随机抽取N_s个样本$\{a^1,a^2,a^3,\cdots,a^{N_s}\}$,然后计算出$\{Q(a^1),Q(a^2),Q(a^3),\cdots,Q(a^{N_s})\}$的值并按降序排序
7: 根据公式(18)更新γ^t
8: 根据公式(24)更新P^t
9: $t=t+1$
10: 如果$\gamma^t=\gamma^{t-1}=\cdots=\gamma^{t-d}$就结束循环

3 实验与结果

3.1 仿真设置

我们假设存在这样一个多层次算力网络(参数设置如表1所示)。该网络为3层算力网络:第1层有多个用户,第2层有10个雾节点,第3层有1个雾节点。用户的任务不可拆分。用户先将任务卸载到第2层雾节点,其他层的雾节点之间通过有线进行连接。假设雾节点的初始状态都为无其他任务在运行。

与CUTS算法相对比的几种基准方法为:

(1)本地计算:每个用户都在本地运行任务;

(2)云计算:每个用户都将任务卸载到云端;

(3)随机卸载:每个用户做出的卸载决策是随机的。

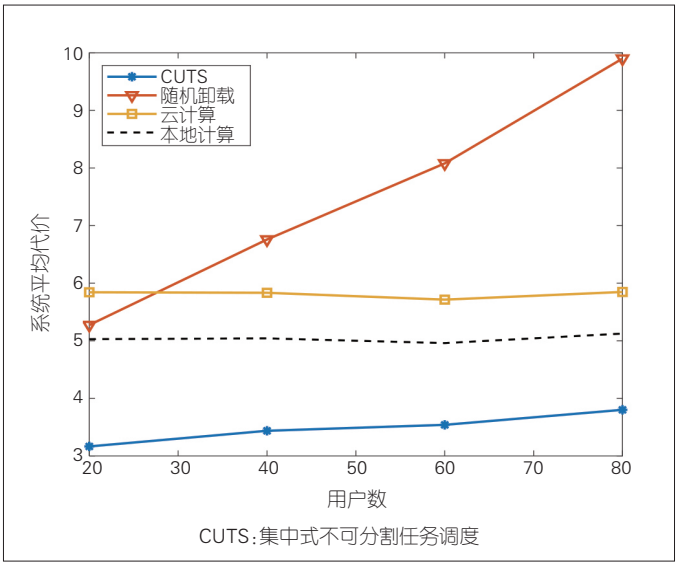
本文以下仿真结果均为400次仿真结果的平均值。

3.2 系统平均代价

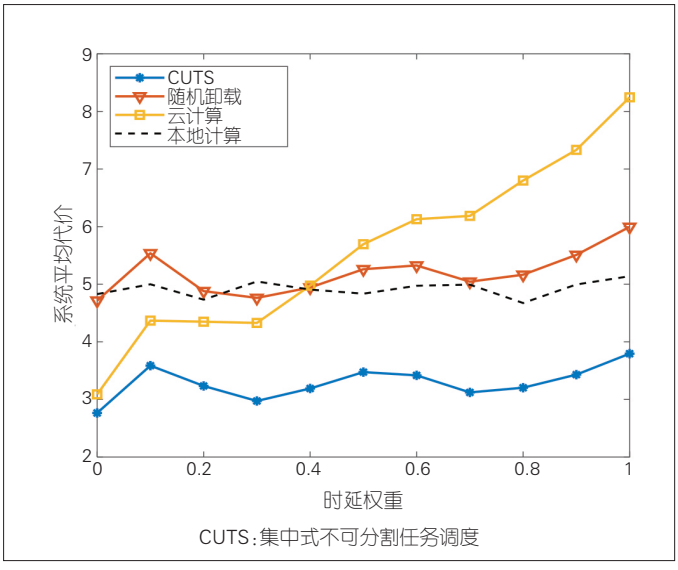
如图2所示,随着用户数的变化,CUTS算法总是能够取得最优的系统平均代价。本地计算通信时延较小,然而总代价却高于多层算力网络,这说明引入算力网络有效解决了本地计算算力较小的问题。图3展示了当

▼表1 仿真参数设置

参数	取值
网络各层节点数	$\{N,10,1\}$
任务大小z_n/kB	[500,5 000]
处理密度γ_n/(cycle·b^{-1})	[500,3 000]
本地设备的计算能力f_n/GHz	{0.8,0.9,1.0,1.1,1.2}
能耗常数κ_n	10^{-27}
二层雾节点的计算能力f_2/GHz	20
用户n到雾节点k的传输速率$R_{n,k}$/(Mbit·s^{-1})	[2.01,4.01]
发送功率$P_{\text{fog},n}$/(mJ·s^{-1})	1 224.78
时延权重α	(0,1)



▲图2 系统平均代价与用户数的关系

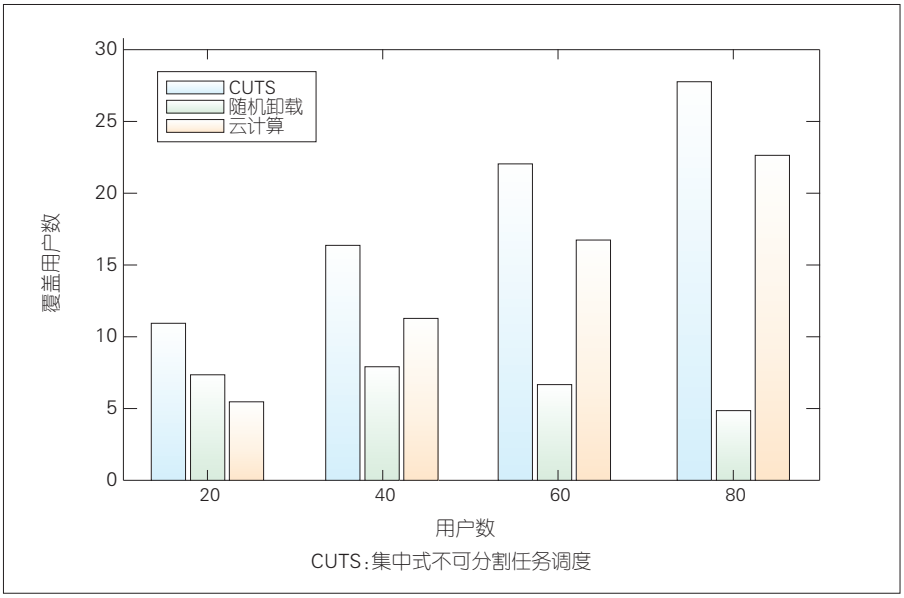


▲图3 系统平均代价与时延权重的关系

用户的任务性质不同时，不同算法的效果。当α值较大时，任务性质偏向时延敏感型。因为云端距离用户较远，通常具有比较大的时延，从图3中我们可以看出，引入多层算力网络可以有效解决云计算网络存在的延迟大的问题。

3.3 受益用户数

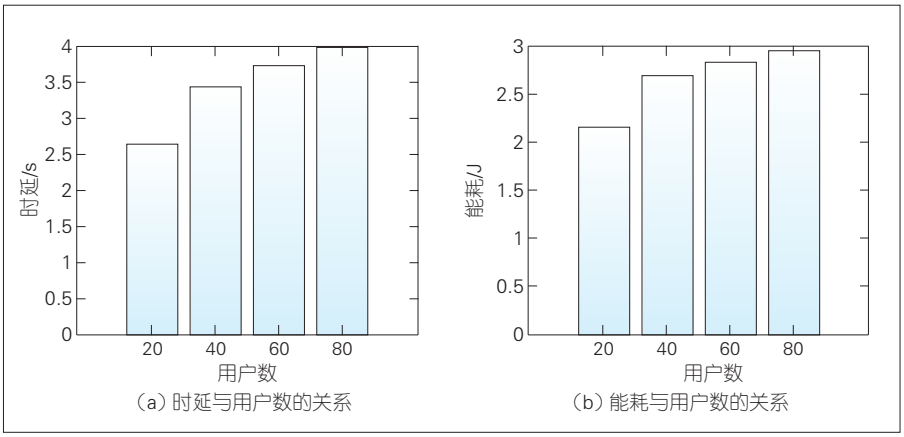
图4展示了在不同算法下的受益用户数。受益用户是指，在当前卸载策略下降低自身处理任务代价的用户。除随即卸载算法外，其他算法的受益用户数都与总用户数呈正相关。由图4可知，CUTS算法依然表现出最优性能。



▲图4 受益用户数与用户数的关系

3.4 时延及能耗成本分布

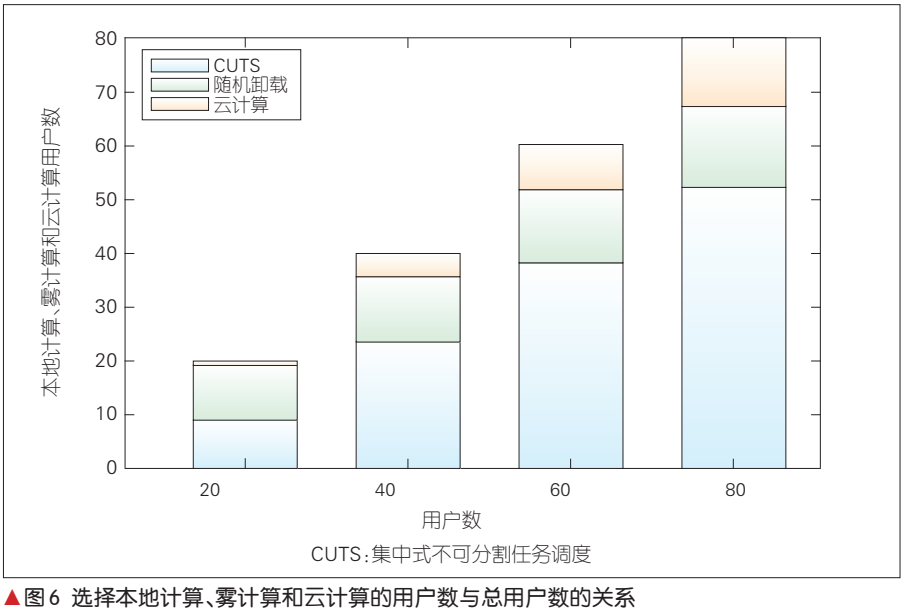
图5展示了随着用户数目的增加，时延和能耗的对比情况。可以看出，随着总用户数的增加，总的代价也在增加，但是增加幅度在减缓。此外，时延产生的代价要略高于能耗产生的代价。



▲图5 时延和能耗与用户数的关系

3.5 本地计算、雾计算和云计算用户分布

图6展示了随着用户数增加，各用户的卸载决策分布。可以看出，选择本地用户和云计算的用户数目逐渐增多，而选择雾节点的用户数却几乎不变。这是因为雾节点的算力资源接近饱和。



▲图6 选择本地计算、雾计算和云计算的用户数与总用户数的关系

4 结束语

本文中，我们提出一种多层次算力网络模型和计算卸载系统，定义一个由时延、能耗组成的加权代价函数，并建模一个任务调度问题。为解决这一问题，我们提出CUTS算法，即将一个确定性问题转化成了一个估计问题，通过重要性采样和交叉熵的方法来求解问题的最优解。数值仿

真结果表明，CUTS算法能够在系统平均代价和受益用户数方面提供最优性能。算力网络可以有效解决单层网络带来的算力小或时延大的问题。

致谢

本研究得到上海科技大学杨旸老师、吴连涛老师的帮助，谨致谢意！

参考文献

[1] REN J K, HE Y H, YU G D, et al. Joint communication and computation resource allocation for cloud–edge collaborative system [C]// 2019 IEEE Wireless Communications and Networking Conference (WCNC). Marrakesh, Morocco: IEEE, 2019: 1–6. DOI: 10.1109/WCNC.2019.8885877
[2] MOURADIAN C, NABOULSI D, YANGUI S M, et al. A comprehensive survey on fog computing: state–of–the–art and research challenges [J]. IEEE communications surveys & tutorials, 2018, 20(1): 416–464. DOI: 10.1109/COMST.2017.2771153
[3] LIU Z N, YANG Y, CHEN Y, et al. A multi–tier cost model for effective user scheduling in fog computing networks [C]//IEEE INFOCOM 2019 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Paris, France: IEEE, 2019: 1–6. DOI: 10.1109/INFCOMW.2019.8845252
[4] MAO Y Y, YOU C S, ZHANG J, et al. A survey on mobile edge computing: the communication perspective [J]. IEEE communications surveys & tutorials, 2017, 19(4): 2322–2358. DOI: 10.1109/COMST.2017.2745201
[5] YANG Y, WANG K L, ZHANG G W, et al. MEETS: maximal energy efficient task scheduling in homogeneous fog networks [J]. IEEE Internet of Things journal, 2018, 5(5): 4076–4087. DOI: 10.1109/JIOT.2018.2846644
[6] DINH T Q, TANG J H, LA Q D, et al. Offloading in mobile edge computing: task allocation and computational frequency scaling [J]. IEEE transactions on communications, 2017, 65(8): 3571–3584. DOI: 10.1109/TCOMM.2017.2699660
[7] CHEN X, JIAO L, LI W Z, et al. Efficient multi–user computation offloading for mobile–edge cloud computing [J]. IEEE/ACM transactions on networking, 2016, 24(5): 2795–2808. DOI: 10.1109/TNET.2015.2487344
[8] NOWAK D, MAHN T, AL–SHATRI H, et al. A generalized Nash game for mobile edge computation offloading [C]//2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). Bamberg, Germany: IEEE, 2018: 95–102. DOI: 10.1109/MobileCloud.2018.00022
[9] YANG Y, LIU Z N, YANG X M, et al. POMT: paired offloading of multiple tasks in heterogeneous fog networks [J]. IEEE Internet of Things journal, 2019, 6(5): 8658–8669. DOI: 10.1109/JIOT.2019.2922324
[10] YANG Y. Multi–tier computing networks for intelligent IoT [J]. Nature electronics, 2019, 2(1): 4–5. DOI: 10.1038/s41928–018–0195–9
[11] LIU Z N, YANG Y, ZHOU M T, et al. A unified cross–entropy based task scheduling algorithm for heterogeneous fog networks [C]//Proceedings of the 1st ACM International Workshop on Smart Cities and Fog Computing. New York, NY, USA: ACM, 2018: 1–6. DOI: 10.1145/3277893.3277896
[12] SHAH–MANSOURI H, WONG V W S. Hierarchical fog–cloud computing for IoT systems: a computation offloading game [J]. IEEE Internet of Things journal, 2018, 5(4): 3246–3257. DOI: 10.1109/JIOT.2018.2838022

作 者 简 介

巩宸宇，上海科技大学信息与技术学院在读硕士研究生；研究领域主要包括物联网与无线通信、雾计算等。

舒洪峰，深圳市智慧城市科技发展集团有限公司副总经理，曾担任深圳市盐田港集团有限公司办公室副主任，深圳市特区建设发展集团有限公司办公室主任、董事会秘书；主要研究领域包括大数据与云计算、5G及城域物联专网、数字经济等。

张昕，教授级高级工程师，深圳市智慧城市科技发展集团有限公司解决方案部部长，深圳市智能交通标准化技术委员会委员、深圳市政府采购中心资深专家、深圳市后备级领军人才；从事智慧城市、智能交通等政府信息化工作，主持并完成综合交通运行指挥中心、智慧宝安总体规划、路边停车系统、侨香路智慧道路、智慧国资管理展示中心及智慧国资大数据中心等项目；获华夏建设科学技术奖一等奖、中国智能交通协会一等奖、深圳市科技创新奖等省部级奖励（7项）；发表论文与专著20余篇，申请发明专利3项，参与编制深圳市地方标准9项。