基于移动边缘计算的任务迁移和协作式负载均衡机制

殷 佳 管昕洁 白光伟
(南京工业大学计算机科学与技术学院 南京 211816)

摘 要 由于使用中心云服务会产生相应的延迟和通信成本,更靠近移动用户的移动边缘计算已经成为处理计算密集型和延迟敏感型应用程序的主要技术。位于网络边缘的小型云数据中心被称为微云,其能够为周围邻近的移动设备提供计算能力,减少服务交付的时延。然而,在移动微云组成的边缘网络环境下,负载均衡问题直接影响了任务的响应时间。为了提高用户服务质量,文中提出基于移动边缘计算的任务迁移和协作式负载均衡机制,包括分别针对用户和微云设计的延迟感知目标选择策略 LATS 和协作式负载均衡策略 CLB。LATS 根据微云当前的负载信息为移动用户选择最优的任务迁移对象;CLB 使用 Balls-into-bins 模型,只需要获取局部信息就可以有效地实现移动微云之间的负载均衡。仿真结果表明,所提策略能够有效减小系统延迟和负载差异,同时降低通信和计算成本。

关键词 移动边缘计算,用户任务迁移,负载均衡,Balls-into-bins 理论,计算通信成本
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/jsjkx.181202453

Task Offloading and Cooperative Load Balancing Mechanism Based on Mobile Edge Computing

YIN Jia GUAN Xin-jie BAI Guang-wei
(College of Computer Science and Technology, Nanjing Tech University, Nanjing 211816, China)

Abstract Because of the corresponding delay and communication cost associated with the use of cloud service, mobile edge computing (MEC) which is closer to mobile users has become the main technology for processing computing-intensive and delay-sensitive applications. Small data centers located on the edge of network are called "cloudlets", which can provide computing resource for nearby mobile devices. The service delivery delay can be reduced significantly by using MEC. However, in the edge network environment composed of mobile micro-clouds, load balancing directly affects the response time of tasks. In order to improve the quality of service for users, this paper proposed a task offloading and cooperative load balancing mechanism. The mechanism includes a latency-aware target selection strategy (LATS) for mobile users and a collaborative load-balancing strategy (CLB) for mobile cloudlets. LATS chooses the best task migration object for mobile users according to the current load information of cloudlets. CLB uses balls-in-bins theory and it can balance the task loads with extremely limited information. Simulations and evaluations demonstrate that the proposed mechanism can effectively reduce the system delay and load gaps, as well as the communication and computing cost.

Keywords Mobile edge computing, User task offloading, Workload balancing, Balls-into-bins theory, Computing and communication cost

1 引言

随着物联网和 5G 技术以及各种新型移动应用(如增强现实、在线游戏和视频直播等)的爆炸式发展,移动用户对网络服务质量(Quality of Service, QoS)的要求也越来越高[1]。但由于电池、存储容量和计算能力的限制,移动设备很难满足资源密集型应用和延迟敏感型应用对时延和可靠性的要求。拥有丰富云端资源的移动云计算(Mobile Cloud Computing, MCC)能够为用户更快速、高效地处理应用程序的任务请求,解决移动设备的电池能耗和资源短缺问题[2-3]。然而,将任务迁移到位于核心网络的中心云会消耗带宽资源,还会导致额外的网络延迟,难以保证用户满意的 QoS[4]。

为了应对以上挑战,移动边缘计算(Mobile Edge Computing, MEC)的概念被提出。MEC 将小规模数据中心即微云部署在网络边缘节点(如基站、无线接入点),为邻近区域的移动用户提供云计算服务,能有效减少网络时延,节省通信消耗,并减轻网络中心的拥塞压力[5-6]。此外,在靠近边缘节点的微云上处理迁移的计算任务请求能更好地保护用户隐私[7]。

到稿日期:2018-12-31 返修日期:2019-02-22 本文受国家自然科学基金项目(61802176,61602235,61501224),江苏省自然科学基金项目(BK20161007),江苏省研究生科研与实践创新计划项目(KYCX18_1074)资助。

殷 佳(1995-),女,硕士生,主要研究方向为边缘计算云、网络优化等,E-mail:yvonneit@njtech.edu.cn;**管昕洁**(1984-),女,博士,硕士生导师,主要研究方向为网络优化、边缘计算、软件定义网络等,E-mail:xjguan@njtech.edu.cn(通信作者);**白光伟**(1961-),男,教授,博士生导师,CCF 杰出会员,主要研究方向为计算机网络、云计算、移动计算等。

在传统的部署策略中,一旦微云被搭建部署,其最大可用物理资源量也被固定,移动设备的任务迁移请求通常也由距离最近的微云处理。这种部署策略会因为负载不均问题造成微云上计算资源的利用率不足。因此,移动性增强的动态微云网络开始受到关注,城市区域中密集的车辆被看作具有计算能力的移动微云,由这些微云组成的动态微云网络可以更有效地处理来自移动用户的任务迁移请求[8-9]。

由于车辆的移动方向和速度具有不确定性,间接性连接的移动微云网络可能会导致微云之间无法连续地进行任务迁移而发生传输错误。除此之外,不同区域中移动用户的密度不同,发送的任务迁移请求会不断波动,这直接影响了移动微云之间的负载均衡[10-11]。比如,在用户高密度区域,微云因任务数量过多而发生过载,而在人群松散区域的微云处于低负载甚至闲置状态。

针对上述问题,为了达到充分利用计算资源、减少平均任务响应时间的目的,本文提出一种基于移动边缘计算的任务迁移和协作式负载均衡机制,并分别从用户和微云的角度考虑,设计了延迟感知的目标选择(Latency-Aware Target Selection,LATS)策略和协作式负载均衡(Collaborative Load-Balancing,CLB)策略。

在 LATS 策略中,每个移动微云会定期向外广播负载信息,用户能够根据微云广播的信息计算出最优的任务迁移对象。在由纯分布式移动微云组成的动态网络中,获取全局负载信息的代价成本非常高,因此,基于移动微云网络中分布式任务的特性,本文使用 Balls-into-bins 模型[12]来模拟任务分配场景,通过应用 Two-choice 理论[13],仅选择并比较通信区域内两个随机邻居云的负载信息,选择负载较少的邻居进行任务重分配,从而实现微云负载的整体均衡性。最后,通过仿真实验证明了所提机制的有效性。

本文第 2 节阐述相关工作;第 3 节介绍系统模型以及支撑理论;第 4 节分别设计延迟感知的目标选择策略和协作式负载均衡策略,并提出最优化问题;第 5 节给出 LATS 算法和 CLB 算法的详细描述;第 6 节进行仿真实验和性能分析;最后总结全文。

2　相关工作

近年来,云计算系统作为一种集中式的计算模式,能够为移动用户处理任务、备份数据[14],但其存在一定的通信延迟和带宽消耗。移动边缘计算(MEC)可以看作是云计算的扩展,其能够节省通信成本并且保证 QoS[15]。文献[16]提出了微云的概念,即靠近移动用户的小型数据中心。微云可以为邻近区域内的用户提供低延迟、高带宽的网络连接。Mao 等[17]对 MEC 系统的资源管理、系统部署、移动性管理等问题进行了研究和探讨。

随着 MEC 的不断发展,移动性增强的微云系统开始受到关注。相关学者针对微云系统中的负载平衡问题提出了不同的解决方案。

第一种方案是微云的策略部署。Xu 等[10]提出了在无线城域网中的微云部署策略,其能够最小化移动设备的访问延迟和平均任务响应时间。Jia 等[3]基于静态边缘计算环境下的任务重定向问题,设计了一种负载均衡最优算法,以保证用户 QoS。然而,在本文的研究场景中,网络的间接性连接以及不断改变的用户任务流,导致上述解决方案无法满足移动微云网络的动态环境。

另外一种方案是面向微云的任务重定向和再分配,用以解决移动用户的相关迁移决策(包括是否迁移、目标微云的选择、迁移的计算任务量)等关键问题。文献[18]提出了移动设备的迁移决策模型,为了减小延迟,先对应用程序的计算任务进行划分,再将子任务迁移至附近的移动设备或者微云。文献[19]对这一工作进行了扩展,使用动态机会迁移算法来最小化处理成本,同时考虑了用户设备的移动性。Zhang 等[20]提出了一种考虑用户移动模式和微云接收控制的最优迁移算法。Jia 等[5]进一步结合微云部署与任务分配问题,提出了基于接入点负载最大的优先算法和基于密度的聚类算法来平衡负载。

以上研究中的微云大多被设定为独立且自私的个体,只服务于附近的移动用户,并且未考虑用户选择的迁移对象对 QoS 的影响。本文提出了一种基于边缘计算的任务迁移和协作式负载均衡机制,针对用户设计延迟感知的目标选择策略,在保证用户 QoS 的同时考虑微云之间协作式的负载均衡策略,以最大化移动微云网络的资源利用率。

3　系统模型

3.1　网络模型

在某个城市的市中心区域 G 内,假设有 N 辆具有计算能力的移动微云,用集合 $C=\{c_1,c_2,\cdots,c_i,\cdots,c_N\}$ 表示,其中 c_i 表示第 i 个移动微云。当微云 i 和微云 j 之间的距离 d_{ij} 在通信范围 R 内时,则可以通过无线接入点(APs)互相连接,并为通信区域内的移动用户提供低延迟无线接入和计算任务处理的服务。移动用户选择通信范围内的任意微云处理发送的迁移任务,微云根据自身负载决定是在本地处理用户的任务请求还是将任务转发至同一通信区域内的其他微云[21]。

移动微云网络模型的实际应用场景如图 1 所示,其中圆形虚线表示移动微云的可通信区域,车辆旁边的矩形表示该移动微云的当前负载。

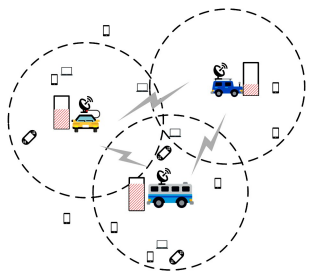


图 1　移动微云网络模型的实际应用场景

Fig. 1　Practical application scenario in mobile cloudlet network

3.2　通信模型

根据文献[22-23],微云 i 和 j 的内部通信时间 t_{ij} 遵从成对速率 α_{ij} 的指数分布,如:$f(t)=\frac{1}{\alpha_{ij}}e^{-\frac{1}{\alpha_{ij}}t}$,$t\geqslant 0$。在任意两个时间间隔 t_a 和 t_b 之内,微云 i 和 j 相遇的可能性为:

$$P_{ij}(t_a,t_b)=e^{-\frac{1}{\alpha_{ij}}t_a}-e^{-\frac{1}{\alpha_{i,j}}t_b} \tag{1}$$

考虑到在通过 Wi-Fi 连接的微云网络中,某些应用程序

(如增强现实、人脸识别)中任务迁移的执行时间大约为 $10^{-4}\sim10^{-2}$ s,本文假设时间间隔(包括执行时间和无线传输的往返时延)足够长,即任务执行结果可以在同一时间间隔内返回给移动用户[24]。

3.3 任务迁移模型

本文考虑通用的任务模型,即"任务"是指应用程序中的一个程序或者微服务,其可以被任意移动微云执行[20],因此来自不同移动用户的迁移任务请求会不断波动。

负载不平衡是指并行系统中任务间计算负载的不均衡分布。这种负载不均会延长任务队列中排队进程执行完成的时间。随着处理器数量的增加,性能损耗将呈线性增长,因此定期平衡任务负载尤为重要。

基于以上考虑,本文假设微云 i 上的任务到达服从泊松分布[3],同时采用文献[25]中的失衡度量和统计偏度来验证任务分配的总体负载均衡情况,具体公式如下:

$$\eta=(\frac{L_{\max}}{\overline{L}})\times100\%,\varphi=\frac{\frac{1}{n}\sum_{i=1}^{n}(L_i-\overline{L})^3}{(\frac{1}{n}\sum_{i=1}^{n}(L_i-\overline{L})^2)^{3/2}} \tag{2}$$

其中,$L_{\max}$ 和 $\overline{L}$ 分别表示最大负载和平均负载。失衡度量可以衡量负载分布的均衡程度,统计偏度则更详细地反映了负载分布结果,例如分布结果是否有过高或者过低的异常值[25]。

4 任务迁移和协作式负载均衡

本节分别针对延迟感知的目标选择策略和协作式负载均衡策略给出了详细的解决方案。针对 LATS 策略,通过量化系统延迟指标为用户提供最优的迁移对象选择;在 CLB 策略中,使用 Balls-into-bins 模型模拟分布式任务分配的场景,从而达到负载均衡的目的。

4.1 延迟感知的目标选择策略

4.1.1 系统延迟指标

假设此刻处于 $x\in R$ 处的移动用户的迁移任务到达速率为 $\lambda(x)$,平均任务大小为 $\omega(x)$,则来自该用户的迁移任务密度为 $\tau(x)=\lambda(x)\omega(x)$。迁移任务请求经历的延迟与各种因素有关,如微云处理任务时所能提供的最大服务速率、基于微云当前负载产生的队列延迟、用户和微云之间的距离导致的通信延迟和其他网络开销[26]。为了简化分析,假设微云 i 为处于 x 处的移动用户提供的服务速率为:

$$s_i(x)=\frac{S_i^{\max}}{1+\beta(dis(x,c_i))^{\alpha}} \tag{3}$$

其中,$S_i^{\max}$ 表示微云 i 提供的最大服务速率;$dis(x,c_i)$ 表示用户和微云 i 之间的欧氏距离;α 和 β 是用于调整服务速率灵活性的参数,以适应各种网络场景。观察式(3)可以发现,微云能够提供给移动设备的服务速率与其最大服务速率成正比,与用户之间的距离成反比。

引入任务分配指标函数 $k(x)$,将微云与移动设备之间的任务分配关系具体化。如果微云 i 向 x 处的用户提供服务,则任务分配指标函数值为 1,否则为 0。微云负载 γ_i 的计算公式为[27]:

$$\gamma_i=\int_R\frac{\tau(x)}{s_i(x)}k_i(x)\mathrm{d}x \tag{4}$$

则所有微云的负载集合 $\gamma=(\gamma_1,\gamma_2,\cdots,\gamma_N)$ 可以定义为:

$$L=\{\gamma|\gamma_i=\int_R\frac{\tau(x)}{s_i(x)}k_i(x)\mathrm{d}x\},0\leqslant\gamma_i\leqslant1-\varepsilon,$$
$$k_i(x)\in\{0,1\},\forall i\in N,\forall x\in R\} \tag{5}$$

其中,ε 表示任意最小正数。

之前假设移动用户的迁移任务到达服从泊松分布,那么微云转发任务的到达也服从泊松分布。根据文献[27],将微云看作 M/G/1-处理器共享队列,则微云 i 的平均任务流量为 $\frac{\gamma_i}{1-\gamma_i}$。根据利特尔法则(Little's law),在一个稳定的系统中,系统延迟与平均流量成正比[28]。因此,可将微云 i 的平均任务流量作为延迟指标 $T_i(\gamma_i)$:

$$\Gamma_i(\gamma_i)=\frac{\gamma_i}{1-\gamma_i} \tag{6}$$

可观察到,当 γ_i 增加并趋于 1 时,延迟将呈指数增长并接近∞。延迟指标没有单位,只是系统延迟的相对指标,通过该延迟指标可以量化系统延迟性能。

4.1.2 LATS 策略最优化问题

考虑到用户需求,为缩短迁移任务处理的响应时间,从而最小化网络延迟,可将最小化网络延迟这一优化问题描述为:

$$\text{Minimize } U(\gamma)=\sum_{i=1}^{N}\Gamma_i(\gamma_i) \tag{7}$$

在每个时间间隔开始时,用户根据微云广播的延迟指标选择最优迁移对象来处理任务请求,以保证自身的 QoS。下一节将针对微云之间的负载均衡问题提出相应的解决方案。

4.2 协作式负载均衡策略

4.2.1 Balls-into-bins 理论

Balls-into-bins 是关于随机分配过程的经典概率模型。假设有 n 个球和 n 个盒子,每个球独立且均匀地随机选择一个盒子,那么最大负载即某个盒子中球的数量最多,该数值近似为 $\frac{\log n}{\log\log n}$。如果每个球随机地在数量 $d\geqslant2$ 的盒子中选择负载最小的那个盒子(称为 d-choice 范例[13]),则在 d-choice 范例中的最大负载近似为 $\frac{\log\log n}{\log d}+\Theta(1)$。

进一步考虑 Balls-into-bins 模型的扩展问题,如果随机将 m 个球放入 n 个盒子,其中 $m\geqslant n\log n$,则最大负载近似为 $\frac{m}{n}+\sqrt{\frac{m\log n}{n}}$。因此,应用 d-choice 范例,且球和盒子的数量关系满足 $m\geqslant n\log n$,那么所有盒子中的最大负载近似为:

$$\frac{m}{n}+\Theta(\sqrt{\frac{m\log n}{n}}) \tag{8}$$

在本文的研究场景中,用户的迁移任务和移动微云分别对应 Balls-into-bins 模型中的球和盒子。Balls-into-bins 模型的理论保证了本文所提出的 CLB 策略可以有效地减小移动微云网络中的最大负载。

4.2.2 CLB 策略的最优化问题

针对微云负载均衡问题,最小化移动微云中任务负载的总体方差,就能实现负载平衡的任务分配。该优化问题可以描述为:

$$\text{Minimize}\sum_{i\in N}\|\gamma_i-E[\gamma]\| \tag{9}$$

除此之外,还可以评估微云当前任务的最大负载和平均负载的差额,最大负载和平均负载由式(2)中的失衡度量和统计偏度计算得出。负载均衡问题如下所示:

$$\text{Minimize} \sum_{i\in N} \| \gamma_i \| - E[\gamma_i] \tag{10}$$

5　算法设计

5.1　延迟感知的目标选择算法

在用户任务迁移阶段,移动微云网络中的所有微云定期更新当前任务负载并根据式(11)评估延迟参数,然后将结果广播至通信区域内的所有移动用户。由用户选择目标微云进行任务迁移,以减小应用程序处理任务的网络时延。为了达到收敛性,本文假设微云同步向外广播延迟参数,且广播时间比任务到达和离开的时间长,以确保用户能够在微云广播下一组延迟参数之前完成任务迁移对象的选择决策。

本文使用上标 h 来表示第 h 个时间间隔的开始时刻,此时微云 i 广播的延迟参数如下所示:

$$\bar{\omega}_i^h = \frac{\partial \Gamma_i^h(\gamma)}{\partial \gamma_i^h} = \frac{1}{(1-\gamma_i^h)^2} \tag{11}$$

基于微云提供的延迟参数和服务速率,位于 x 处的移动用户会选择合适的微云来处理自己的任务迁移请求。具体公式如下:

$$k_i^h(x) = \begin{cases} 1, & \text{if } i = \arg\max\limits_{i\in C} \dfrac{s_i(x)}{\bar{\omega}_i^h} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

根据上一节的描述,$k_i(x)$ 表示微云 i 是否为 x 处的移动用户提供服务,若值为 1,则表示提供,反之则不提供。$s_i(x)$ 则表示微云 i 为 x 处的移动用户提供的服务速率。

在时间间隔 h 将要结束时,微云 i 会再一次评估当前负载 $P_i(\gamma_i^h)$:

$$P_i(\gamma_i^h) = \min\left(\int_R \frac{\tau(x)}{s_i(x)} k_i(x)\mathrm{d}x, 1-\varepsilon \right) \tag{13}$$

负载信息更新完毕之后,广播下一个延迟指标:

$$\gamma_i^{h+1} = \sigma\gamma_i^t + (1-\sigma)P_i(\gamma_i^h) \tag{14}$$

其中,$0<\sigma<1$。

LATS 算法的具体执行过程如算法 1 所示。

算法 1　LATS

1. Input:R, S_i^{max}, λ(x), ω(x)
2. Output:U(γ)
3. for h=[1:T] do
4.　for i =1 to N do
5.　　$s_i(x) \leftarrow \dfrac{S_i^{max}}{1+\beta(dis(x,c_i))^{\alpha}}$
6.　　$\omega_i^h \leftarrow \dfrac{1}{(1-\gamma_i^h)^2}$
7.　　if $i=\arg\max \dfrac{s_i(\chi)}{\bar{\omega}_i^h}$ then
8.　　$k_i^h(x)=1$
9.　　else then
10.　　$k_i^h(x)=0$
11.　　end if
12.　　return U(γ)
13.　end for
14. end for

5.2　协作式负载均衡算法 CLB

首先介绍在 CLB 算法设计中应用的 d-choice 范例。该范例具有两个特性[13],第一个是随机选择的高效用性,即使简单地随机选择两个对象(即 $d=2$)进行比较,也比只选择一个目标能够更有效地实现负载均衡;第二个特性是目标选择的随机性,由于移动微云网络的间接性连接,每个微云的邻居会随着时间间隔的迭代不断更新。

所有微云在初始阶段检查是否有新的微云进入自己的通信范围,并更新邻居列表。如果邻居数量大于 d,则应用 d-choice 范例;否则,减小 d 值,直到其小于当前邻居数量。

第二阶段是任务重分配,每个微云在当前的邻居列表中随机选择 d 个邻居,然后比较邻居云的任务负载并进行排序,最终选择负载最小的邻居执行任务迁移。根据文献[29]所提出的比例算法(Proportional algorithm),计算源微云和目标微云之间的迁移概率,并基于该概率进行任务迁移。最后,计算失衡度量和统计偏度用以衡量每个时间间隔内的负载均衡情况,算法的输出能够直接反映 CLB 策略的优化情况。任务重分配过程结束后,当前时间间隔结束,新的时间间隔开始。CLB 算法的具体描述如算法 2 所示。

算法 2　CLB

1. Input:R, S_i^{max}, λ(x), ω(x)
2. Output:η,φ
3. for h=[1:T] do
4.　for i =1 to N do
5.　　if len(l(i))≫d then
6.　　Select d neighbors //选择 d 个邻居
7.　　else if len(l(i))≪d
8.　　d←d/2 until len(l(i))≫ d
9.　　end if
10.　　Select d neighbors
11.　　Select j in d
12.　　for q = [2:d] do
13.　　if $\gamma_j > \gamma_q$ then j←q
14.　　end if
15.　　end for
16.　　if $\gamma_i > \gamma_j$ then
17.　　Prob ←1− γ_j/γ_i
18.　　$\gamma_j \leftarrow l_j + W(i) * Prob$
19.　　$\gamma_i \leftarrow l_i - W(i) * Prob$
20.　　end if
21.　end for
22. end for
23. return η,φ

6　仿真实验与结果分析

本节通过仿真实验对本文所提出的目标选择策略和负载均衡策略进行性能分析。

首先在城区 G 内随机生成每个移动微云的位置。当时间间隔开始时,根据随机漫步(Random walk)算法更新微云的位置。每个微云在通信区域 R 内随机选择 2($d=2$)个邻居,并选择负载较小的目标微云进行任务迁移,这一过程的计算复杂度为 O(1)。

选择 3 种分配策略作为对比,分别为随机分配策略、比例分配策略[12]和贪婪分配策略[23]。在随机分配策略中,移动微云随机选择通信范围内的邻居云进行任务迁移。与之相反,贪婪分配策略首先查询每一个邻居云的负载信息,然后选

择负载最小的微云进行任务迁移，这一过程的计算复杂度为 $O(n)$。在比例分配策略中，根据选择的目标微云和源微云的负载信息计算出任务迁移概率，按照比例进行任务重分配。

通过 Matlab R2015b 平台设计并实现基于移动边缘计算的任务迁移和协作式负载均衡策略的仿真实验。移动用户的任务迁移请求服从泊松分布，且每个微云的任务请求到达速率服从正态分布，具体参数设置如表 1 所列。

表 1 实验参数

Table 1 Experimental parameters

参数名称	参数值
目标区域	100 m×100 m
移动微云数量	30
通信区域半径/m	20
任务迁移请求到达速率	$N(4,2)>0$
任务请求数据量/kB	20
灵活性参数	1
灵活性参数	10
平均参数	0.95

6.1 延迟性能

图 2 反映了不同任务迁移请求到达速率对网络延迟性能的影响。可以看出，本文提出的 LATS 策略比传统机制的延迟低，并且随着任务到达速率的增加，延迟性能表现更好。因为当任务到达速率比较高时，在传统机制中，用户只选择距离最近的微云进行任务迁移，当该微云周围的用户过于密集时就会出现过载现象，从而导致网络总延迟增加。但是在本文提出的 LATS 策略中，用户在选择目标微云时可以同时考虑微云的当前负载，因此网络延迟性能得到了很大的提升。

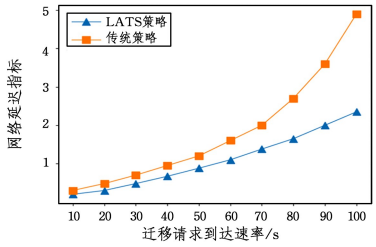
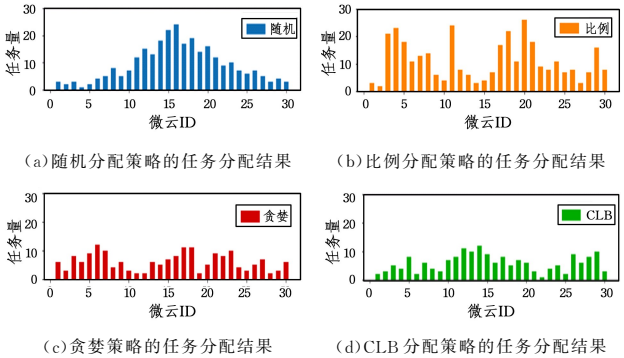


图 2 不同任务迁移请求到达速率下的网络延迟性能

Fig. 2 Latency behavior with different offload request arrival rates

6.2 任务分配结果

CLB 策略和 3 种对比策略在所有微云上的任务分配结果如图 3 所示。



(a)随机分配策略的任务分配结果 (b)比例分配策略的任务分配结果

(c)贪婪策略的任务分配结果 (d)CLB 分配策略的任务分配结果

图 3 4 种分配策略的任务分配结果

Fig. 3 Task allocation results of four schemes

从图 3(a)中可以看到，ID 为 11－22 的移动微云上的任务数量都超过 10，可能导致过载的发生，而靠近区域边缘的其他微云却处于闲置状态。同样地，采用比例分配策略的任务分配结果也不均衡，如图 3(b)所示，相较于随机分配策略，该策略下任务数量超过 10 的微云分布得更加稀疏。但图 3(c)和图 3(d)中，大部分微云上的任务数量都少于 10。因此与随机分配策略和比例分配策略相反，CLB 策略和贪婪分配策略都能够使微云负载相对均衡，而贪婪算法通过比较所有邻居的任务负载，选择最优即负载最小的目标进行任务迁移，计算复杂度较高。

6.3 负载均衡性

本文进一步利用失衡度量和统计偏度来衡量负载均衡结果。其中，失衡度量可以用来衡量负载不均的严重程度，该值越低表示负载均衡结果越好。统计偏度为正，则表示微云当前的任务数量大于平均值，负值反之。

随着时间间隔的不断迭代，图 4 给出了 4 种分配策略的失衡度量的变化情况。可以看出，贪婪分配策略的失衡度量趋近于 0，这说明该策略实现的负载均衡结果表现最优。CLB 策略和比例分配策略的失衡度量分别趋近 0.1 和 0.35。实验结果表明，随机分配策略下的负载最不均衡，其次是比例分配策略，而本文提出的 CLB 策略能够在更低的通信和计算成本下实现接近最优的负载均衡分配结果。

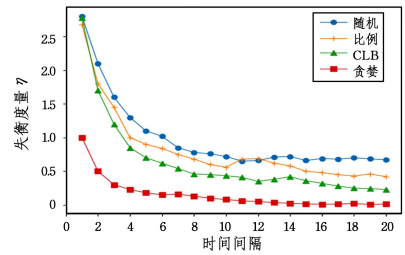


图 4 4 种分配策略的失衡度量值

Fig. 4 Imbalance metrics obtained with four schemes

图 5 描述了 4 种分配策略的统计偏度的变化情况。可以直观地观察到，比例分配策略的统计偏度值大约为 2，而随机分配策略下的统计偏度值为剧烈波动的负值，这说明大部分微云都处于闲置状态。CLB 策略和贪婪分配策略的统计偏度值都逐渐趋近于 0，这说明只有少部分的微云负载处于过载或者闲置状态，从而减小了各个移动微云之间的负载差异。

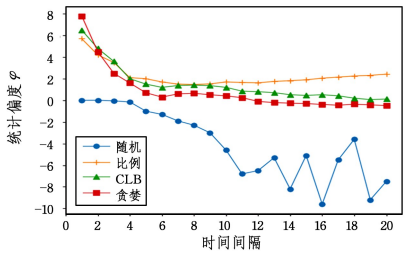


图 5 4 种分配策略的统计偏度

Fig. 5 Statistical skewness metrics obtained with four schemes

结束语 本文提出了基于移动边缘计算的任务迁移和协作式负载均衡策略。为了保证用户的 QoS，减小网络时延，提高计算资源的利用率，本文提出了一种基于边缘计算的任务迁移和协作式负载均衡机制。针对用户的最佳迁移对象选择，设计了延迟感知的目标选择策略 LATS；并综合考虑了微

云之间的负载均衡问题,提出了协作式负载均衡策略 CLB。在本文所提出的机制中,只需要获取局部信息就可以有效地实现移动微云负载的整体均衡。最后,通过仿真实验验证了所提策略的有效性。仿真结果表明,相比于传统的用户任务分配方法,LATS 算法能够有效减小网络延迟;CLB 算法在任务分配中接近依赖于全局任务负载信息的贪婪分配策略,且同时保持了较低的通信成本。

参 考 文 献

[1] BOTTA A,DE DONATO W,et al. Integration of cloud computing and internet of things: A survey[J]. Future Generation Computer Systems,2016,56(2):684-700.

[2] OHU Y C,PATEL M,SABELLA D,et al. Mobile Edge Computing:A Key Technology towards 5G[J]. ETSI White Paper,2015,11(11):1-16.

[3] JIA M K,LIANG W F,XU Z C,et al. Cloudlet load balancing in wireless metropolitan area networks[C]// Proceedings of IEEE INFOCOM. San Francisco,CA,USA,2016:1-9.

[4] KUMAR K,LIU J,LU Y H,et al. A Survey of Computation Offloading for Mobile Systems[J]. Mobile Networks and Applications,2013,18(1):129-140.

[5] JIA M,CAO J,LIANG W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing,2017,5(4):725-737.

[6] TRAN T X,HAJISAMI A,PANDEY P. Collaborative mobile edge computing in 5g networks:New paradigms,scenarios,and challenges[J]. IEEE Communications Magazine,2017,55(4):54-61.

[7] CLINCH S,HARKES J,FRIDAY A,et al. How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users[C]// Proceedings of IEEE Pervasive Computing and Communication. Switzerland,2012:122-127.

[8] CAO H,CAI J. Distributed multiuser computation offloading for cloudlet-based mobile cloud computing:A game-theoretic machine learning approach[J]. IEEE Transactions on Vehicular Technology,2018,67(1):752-764.

[9] JEONG S,SIMEONE O,KANG J. Mobile edge computing via a UAV-Mounted cloudlet:Optimization of bit allocation and path planning[J]. IEEE Transactions on Vehicular Technology,2018,67(3):2049-2063.

[10] XU Z,LIANG W,XU W,et al. Efficient algorithms for capacitated cloudlet placements[J]. IEEE Transactions on Parallel and Distributed Systems,2016,27(10):2866-2880.

[11] ZHAO J,YANG K,WEI X,et al. A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment[J]. IEEE Transactions on Parallel and Distributed Systems,2016,27(2):305-316.

[12] VÖCKING B. How asymmetry helps load balancing[J]. Journal of the ACM (JACM),2003,50(4):568-589.

[13] MITZENMACHER M. The power of two choices in randomized load balancing[J]. IEEE Transactions on Parallel and Distributed Systems,2001,12(10):1094-1104.

[14] FERNANDO N,LOKE S W,RAHAYU W. Mobile cloud computing:A survey[J]. Future Generation Computer Systems,2013,29(1):84-106.

[15] BASTUG E,BENNIS M,DEBBAH M. Living on the Edge:The Role of Proactive Caching in 5G Wireless Networks[J]. IEEE Communications Magazine,2014,52(8):82-89.

[16] SATYANARAYANAN M,BAHL P,CACERES R,et al. The case for VM-based cloudlets in mobile computing[J]. IEEE Pervasive Computing,2009,8(4):14-23.

[17] MAO B Y,YOU C S,ZHANG J. A survey on mobile edge computing:The communication perspective[J]. IEEE Communications Surveys & Tutorials,2017,19(4):2322-2358.

[18] ZHANG Y,NIYATO D,WANG P,et al. Dynamic Offloading Algorithm in Intermittently Connected Mobile Cloudlet Systems[C]// Proceedings of IEEE International Conference on Communications. Sydney,Australia,2014.

[19] HUU T T,THAM C K,NIYATO D. To Offload or to Wait:An Opportunistic Offloading Algorithm for Parallel Tasks in a Mobile Cloud[C]// Proceedings of IEEE 6th International Conference on Cloud Computing Technology and Science. Singapore,2014.

[20] ZHANG Y,NIYATO D,WANG P. Offloading in mobile cloudlet systems with intermittent connectivity[J]. IEEE Transactions on Mobile Computing,2015,14(12):2516-2529.

[21] GUO X J,LIU L Q,CHANG Z,et al. Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds[J]. Wireless Network,2018,24(1):79-88.

[22] TOURNOUX P U,LEGUAY J,BENBADIS F,et al. The accordion phenomenon: Analysis, characterization, and impact on DTN routing[C]// Proceedings of INFOCOM. Brazil, 2009:1116-1124.

[23] LI Q Y,YANG P T,FAN X C,et al. Taming the big to small:Efficient selfish task allocation in mobile crowdsourcing systems[J]. Concurrency and Computation Practice and Experience,2017,29(14):2213-2226.

[24] LIU Y,LEE M J,ZHENG Y. Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system[J]. IEEE Transactions on Mobile Computing,2016,15(10):2398-2410.

[25] PEARCE O,GAMBLIN T,DE SUPINSKI B R,et al. Quantifying the effectiveness of load balance algorithms[C]// Proceedings of the 26th ACM International Conference on Supercomputing. San Servolo Island,Venice,Italy,2012:185-194.

[26] CHEN Z,HU W L,WANG J J,et al. An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance[C]// Proceedings of the Second ACM/IEEE Symposium on Edge Computing Article. San Jose,California,2017.

[27] LIU D,CHEN Y,CHAI K K,et al. Distributed latency-energy aware user association in 3-tier HetNets with hybrid energy sources[C]// Proceedings of IEEE Globecom Workshops. Austin,TX,USA,2014.

[28] KIM S H,WHITT W. Statistical analysis with Little's Law[J]. The Institute for Operations Research and the Management Science,2013,16(4):1030-1045.

[29] BERENBRINK P,FRIEDETZKY T,GOLDBERG L A,et al. Distributed selfish load balancing[C]// Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm. Miami,Florida,2006:354-363.