移动边缘计算场景下针对资源竞争的服务迁移优化方法

王海艳[1,2]，张霖[1]，骆健[1]
（1.南京邮电大学计算机学院，江苏 南京 210023；2.江苏省大数据安全与智能处理重点实验室，江苏 南京 210023）

摘　要：针对移动边缘计算（MEC）场景中边缘服务器资源受限导致服务迁移存在资源竞争的问题，基于Lyapunov技术和博弈论，提出了一种针对资源竞争的服务迁移优化方法OMRC-LG。考虑到系统迁移成本有限且当用户数量过多时难以进行轨迹预测，将服务迁移问题建模为迁移成本约束下的最优化问题，并利用Lyapunov技术将最优化问题转化为不需要预测用户轨迹的在线问题处理。为了缓解资源竞争，提出了一种基于博弈论的分布式方法求解在线问题，通过共享用户服务迁移决策以获取准确的边缘服务器可用资源，并不断更新迁移决策，实现服务迁移优化。仿真结果表明，OMRC-LG方法在满足迁移成本约束的同时，降低了平均服务时延。

关键词：移动边缘计算；服务迁移；服务时延；迁移成本；资源竞争
中图分类号：TP393
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2024143

Service migration optimization method for resource competition in mobile edge computing scenarios

WANG Haiyan[1,2], ZHANG Lin[1], LUO Jian[1]
1. School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
2. Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing 210023, China

Abstract: To tackle the problem of resource competition among service migrations caused by limited edge server resources in mobile edge computing (MEC) scenarios, a service migration optimization method for resource competition based on Lyapunov and game theory (OMRC-LG) was proposed. Considering the system's limited migration costs and the difficulty of predicting trajectories when the number of users was large, the service migration was modeled as an optimization problem with migration cost constraints and used the Lyapunov technique to transform it into an online problem without user trajectory prediction. To alleviate resource competition among users, a distributed method based on game theory was proposed. By sharing user service migration decisions, the method obtained accurate information on available edge server resources and would continuously update these decisions to optimize service migration. Simulation results show that the OMRC-LG method can reduce the average service delay while satisfying the migration cost constraints.

Keywords: mobile edge computing, service migration, service delay, migration cost, resource competition

0　引言

近年来，随着人工智能和大数据技术的迅速发展，新型移动应用如增强现实、虚拟现实和交互式游戏不断涌现。移动设备由于存储、计算及缓存能力有限，已经难以满足这些新兴应用对计算资源和低时延的高需求[1]。移动边缘计算（MEC, mobile edge computing）将边缘服务器与基站融合形成微服务中心，部署在靠近用户的网络边缘，为资源受

收稿日期：2024–03–29；修回日期：2024–07–16
通信作者：王海艳，wanghy@njupt.edu.cn
基金项目：国家自然科学基金资助项目（No.62272243）
Foundation Item：The National Natural Science Foundation of China (No.62272243)

限的移动设备提供了新的解决方案。在这种架构下，移动设备不需要承担所有计算任务，可以将一些复杂任务卸载至边缘服务器上执行，从而能够运行具有严格质量要求的复杂应用程序[2]。然而，这种架构也带来了新的挑战。在MEC场景中，用户位置的动态变化可能导致移动用户与边缘服务器上的服务之间的通信需要经过多次跳转，这会显著增加服务时延。因此，服务需要在边缘服务器之间动态迁移，以适应用户位置的变化，避免服务时延恶化。

现有的服务迁移策略主要根据用户移动轨迹、迁移成本以及服务器可用资源等信息制定服务迁移决策[3]。然而，大多数迁移方案从单用户角度制定决策，忽略了用户之间的资源竞争（如计算、存储和I/O资源[4]），容易导致用户间盲目争夺有限资源，影响服务迁移效果。单用户角度的迁移决策未考虑其他用户的影响，可能导致多个服务同时迁移到同一个边缘服务器，超出其资源限制或者过多服务共享有限的I/O资源，最终影响服务响应。尽管可以将多用户的服务迁移问题建模为整体优化问题，但这种集中式方法在实际应用中仍存在一定的局限性。随着用户数量不断增加，集中式方法的问题求解空间可能呈指数增长，难以在有限时间内寻找出最优解，可扩展性受到限制。

针对MEC场景中制定服务迁移决策时面临的资源竞争问题，本文提出了一种基于Lyapunov技术和博弈论的服务迁移优化方法OMRC-LG，主要工作如下。

1) 由于系统迁移成本有限以及用户数量较多时难以进行全面的轨迹预测，基于用户历史迁移数据进行迁移成本的分配，将服务迁移问题建模为带有迁移成本约束的优化问题，并采用Lyapunov技术将其转化为在线问题处理，避免预测用户未来的移动轨迹。

2) 提出了一种基于博弈论的分布式算法求解在线问题，通过共享用户服务迁移决策来获取准确的服务器可用资源，并不断更新迁移决策，缓解了服务迁移之间的资源竞争。相比集中式方法，所提方法具有更强的可扩展性。

3) 通过仿真实验对所提方法进行评估，结果表明，所提方法在满足迁移成本约束的同时，降低了平均服务时延。

1 相关工作

在MEC场景中，服务迁移作为解决用户位置动态变化引发服务响应时延过高问题的关键技术，吸引了研究者的广泛关注。现有工作根据是否考虑用户之间的资源竞争可以分为未考虑资源竞争和考虑资源竞争2类。

1) 未考虑资源竞争

大多数工作未考虑到用户间的资源竞争，这些工作通常以迁移成本和服务时延的组合为优化目标制定服务迁移决策。文献[5-6]分别采用了基于马尔可夫决策过程（MDP, Markov decision process）的策略迭代和值迭代方法求解服务迁移决策。由于实际场景中的复杂随机性使得难以对MDP进行准确建模，一些工作转向使用无模型的强化学习方法。文献[7]提出了一种强化学习方法Q-learning求解迁移决策。文献[8]进一步提出使用深度强化学习方法深度Q网络（DQN, deep Q network）来制定服务迁移决策。这些方法通常要求问题的求解空间不能过大，所以大多采用了单用户服务迁移问题的建模。然而，单用户服务迁移方案忽略了资源竞争，当用户数量较多时容易导致用户间盲目争夺有限资源，从而增加服务时延，影响迁移效果。文献[9]研究了长期成本预算约束下的移动边缘服务放置问题，提出了基于Lyapunov技术的预测服务放置算法。但该算法仅考虑了单用户的情况，当用户数量较多时仍然存在资源竞争的问题，并且难以为所有用户进行轨迹预测。因此，本文将服务迁移问题建模为不需要预测用户轨迹的在线问题处理，提出了针对用户之间资源竞争的服务迁移解决方案，通过缓解用户之间的资源竞争，实现服务迁移决策的优化。

2) 考虑资源竞争

仅少数工作考虑了资源竞争，这些工作大部分采用集中式解决方法。文献[10]研究了服务迁移和任务重路由的联合优化问题，提出了一种基于改进的Lyapunov技术的在线算法，结合随机舍入和拉格朗日对偶技术的迭代算法，以优化长期平均服务时延。文献[11]研究了多任务迁移中最小化迁移成本的问题，考虑了边缘服务器上的I/O干扰，提出了一种基于松弛和舍入的有效求解方法。文献[12]研究了异构密集蜂窝网络中的服务迁移问题，通过基于Lyapunov技术和粒子群优化的高效节能在线

算法来最小化平均能耗。尽管集中式方法避免了单用户角度制定决策所引起的资源竞争问题，但随着用户数量的增加，这些方法的问题求解空间可能呈指数增长，限制了其可扩展性。分布式方法在可扩展性方面具有优势，已经有一些工作将其应用于资源竞争问题。文献[13]采用启发式思想设计了一种分布式迁移策略，以解决资源竞争造成的边缘服务器过载问题。该方法使用了一种停止-等待协议的算法，用户在制定服务迁移决策时，首先发送预迁移请求，随后在获得候选服务器的许可后进行服务迁移。然而，这种方法忽略了服务迁移决策对服务器可用资源的影响，未能充分利用边缘服务器的有限资源降低服务时延。本文提出了一种基于博弈论的分布式方法解决服务迁移问题。相比于集中式方法，所提方法在可扩展性方面更具优势。同时，该方法通过共享用户服务迁移决策，能够获取准确的服务器可用资源，并利用这些信息不断更新服务迁移决策，降低服务时延。

2　系统模型和问题构建

2.1　场景描述和服务迁移建模

如图 1 所示，考虑 MEC 场景中有 M 个边缘服务器和 U 个移动用户，$\mathcal{M}=\{1,2,\cdots,M\}$ 表示边缘服务器集合，$\mathcal{U}=\{1,\cdots,U\}$ 表示移动用户集合。每个边缘服务器与一个基站集成，移动用户可以通过无线接入网等方式访问服务器。移动用户的服务以虚拟机（VM, virtual machine）的形式在边缘服务器上运行，一个虚拟机是用户服务环境的软件克隆，它包含用户的配置文件和应用程序，用于处理用户的服务请求，VM_u 对应于用户 u 的服务。当移动用户在 MEC 场景中动态移动时，边缘服务器可以将服务器信息和用户信息传输至集中式控制器，由控制器制定迁移决策，动态迁移服务以避免服务时延恶化。在 MEC 场景中，集中式控制器制定服务迁移决策时所面临的资源竞争情况：用户 1、用户 3 和用户 4 由服务器 1 提供服务，各个用户的服务会共享有限的 I/O 资源，这会导致服务器性能下降，从而导致服务时延的增加。当用户 1 和用户 2 移动到边缘服务器 4 附近时，2 个用户的服务迁移存在着对服务器 4 的资源竞争，如果服务器 4 的资源不足，就可能导致迁移失败。

本文假设每个用户同一时隙有且仅有一个计算服务，用户服务 S_u 的相关配置信息表示为 $S_u(\lambda_u,\gamma_u,\theta_u)$，其中，$\lambda_u$ 为服务 S_u 的请求数据大小，γ_u 为服务 S_u 的计算强度（单位为 cycle/bit），θ_u 表示服务 S_u 部署在边缘服务器上的虚拟机数据大小。本文系统以一个时隙的方式运行 $t\in\mathcal{T}=\{1,2,\cdots,T\}$，一个时隙长度为 τ。为了更好地描述用户的移动，

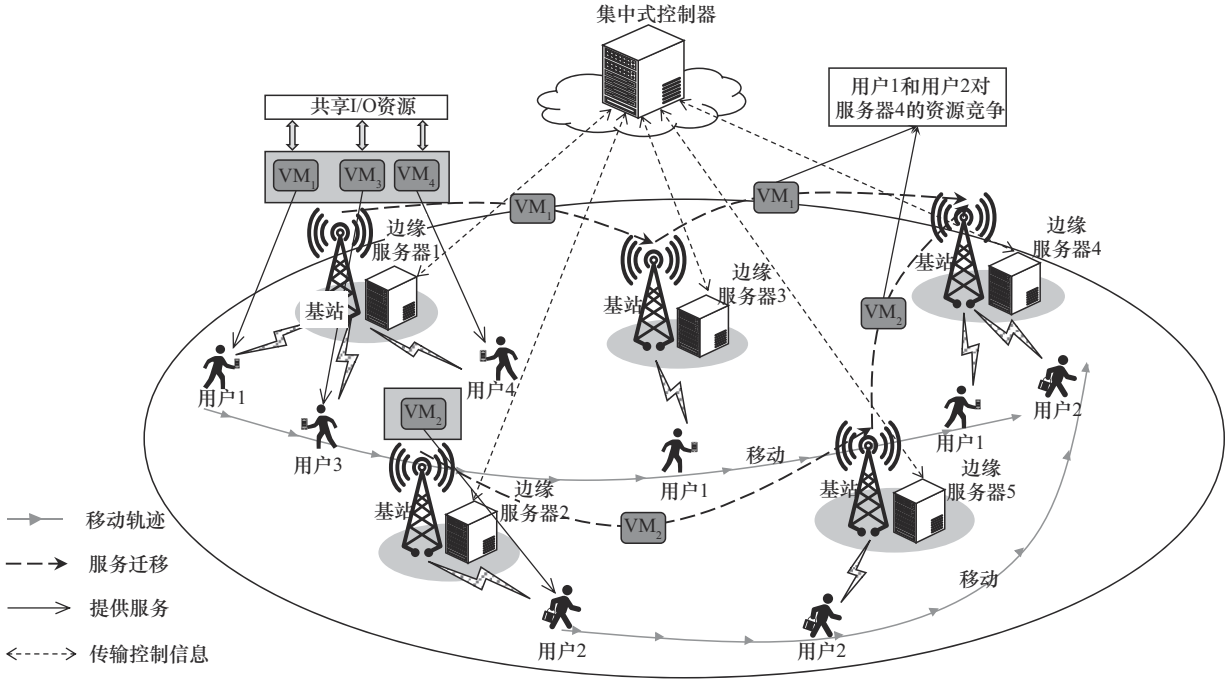


图1　服务迁移场景

假设用户的位置在时隙内不变，在时隙间变化。假设向量$\boldsymbol{H}^t=\{h_1^t,h_2^t,\cdots,h_u^t\}$表示在$t$时刻用户的服务所在的边缘服务器，其中$h_u^t\in\{1,2,\cdots,m\}$，每一个服务由一台边缘服务器提供；向量$\boldsymbol{A}^t=\{a_1^t,a_2^t,\cdots,a_u^t\}$表示在$t$时刻所有服务的迁移决策，其中$a_u^t\in\{1,2,\cdots,m\}$，每一个服务只能迁移到其中一台边缘服务器。因为服务迁移的时间相比于时隙长度来说很短，所以有$\boldsymbol{H}^{t+1}=\boldsymbol{A}^t$。

边缘服务器资源有限，只能处理有限的用户服务请求，假设$C_m(F_m,R_m)$表示边缘服务器m的资源容量，F_m表示服务器m的最大计算速度，R_m表示服务器m的存储大小，则服务迁移决策需要满足以下约束条件

$$\sum_{u=1}^{U}\theta_u I\{a_u^t=m\}\leqslant R_m,\forall t\in\mathcal{T},m\in\mathcal{M} \tag{1}$$

$$\sum_{u=1}^{U}f_{u,m}I\{a_u^t=m\}\leqslant F_m,\forall t\in\mathcal{T},m\in\mathcal{M} \tag{2}$$

其中，$I\{x\}$是一个布尔函数，当x为真时，$I\{x\}=1$，否则为0；$f_{u,m}$表示边缘服务器m分配给用户u的计算速度。式(1)和式(2)分别表示服务迁移需要满足边缘服务器的存储和最大计算速度约束。接下来，介绍服务迁移的优化目标：服务时延和迁移成本。

2.2 服务时延和迁移成本模型

服务时延由通信时延、计算时延和迁移时延3个部分组成。通信时延包括用户u发送服务请求到边缘服务器m和边缘服务器m将结果返回用户u的时间，通常情况下，输出数据相比输入数据而言很小，所以可以忽略输出数据的传输时间。本文使用$r(u,m)$表示用户u与边缘服务器m之间的上行数据传输速率，即

$$r(u,m)=W\mathrm{lb}\left(1+\frac{P_u\boldsymbol{h}d(u,m)^{-3}}{\sigma_u^2}\right) \tag{3}$$

其中，P_u、$\boldsymbol{h}$和σ_u^2分别表示用户u的发射功率、复衰落向量和噪声功率[14]；$d(u,m)$表示用户u与边缘服务器m之间的参考距离，该距离通过用户与边缘服务器的经纬度计算得出，单位为米（m），定义为

$$d(u,m)=2R\arcsin\left(\sqrt{\sin^2\frac{\mathrm{lat}_u-\mathrm{lat}_m}{2}+\cos(\mathrm{lat}_u)\cos(\mathrm{lat}_m)+\sin^2\frac{\mathrm{lng}_u-\mathrm{lng}_m}{2}}\right) \tag{4}$$

其中，lng_u和lat_u表示用户u的经纬度所对应的弧度，lng_m和lat_m表示服务器m的经纬度所对应的弧度，R表示平均地球半径。因此，通信时延可以表示为

$$L_t^{\mathrm{cm}}(u,m)=\frac{\lambda_u}{r(u,m)} \tag{5}$$

计算时延是指用户将服务请求发送给服务器之后，服务器处理请求的时间。考虑到多个虚拟机共享服务器有限的I/O资源会导致计算速度下降，本文引入了$d_m>0$作为服务器m的性能退化因子[11]。因此，用户u的服务实际计算速率为

$$F_{u,m}=(1+d_m)^{1-\sum_{u=1}^{U}I\{a_u^t=m\}}f_{u,m} \tag{6}$$

式(6)表示每个用户的计算速率随着服务器上虚拟机数量的增加而减小。因此，服务请求的计算时延可以表示为

$$L_t^{\mathrm{cp}}(u,m)=\frac{\lambda_u\gamma_u}{F_{u,m}} \tag{7}$$

迁移时延是指服务迁移过程中的服务停机时间，通常服务以虚拟机的形式实时迁移到目标服务器。实时迁移采用的是预复制内存迁移技术[15]，它在不中断服务的情况下，直接将整个虚拟机传输到目标服务器，然后迭代地传输服务迁移过程中源服务器内存中改变的脏页，直到满足预先指定的标准。最后，停止源服务器上的服务，将剩余数据迁移到目标服务器。最后阶段的数据迁移会产生一定的服务停机时间，本文使用c_u来近似表示这部分数据的迁移时间，定义为

$$c_u=L_{\mathrm{left}}^{\mathrm{mig}}+(L_{\mathrm{right}}^{\mathrm{mig}}-L_{\mathrm{left}}^{\mathrm{mig}})\,\mathrm{beta}(\theta_u,d(h_u^t,a_u^t)) \tag{8}$$

其中，$L_{\mathrm{left}}^{\mathrm{mig}}$和$L_{\mathrm{right}}^{\mathrm{mig}}$分别表示迁移时延的左右边界；beta表示符合beta分布的随机数生成器，其会根据虚拟机数据大小θ_u和边缘服务器之间服务迁移的距离$d(h_u^t,a_u^t)$生成0到1范围的随机值。所以，迁移时延可以表示为

$$L_t^{\mathrm{mig}}(u)=c_u I\{h_u^t\neq a_u^t\} \tag{9}$$

因此，服务时延$L_t^{\mathrm{Qos}}(u)$为

$$L_t^{\mathrm{Qos}}(u)=L_t^{\mathrm{cm}}(u,m)+L_t^{\mathrm{cp}}(u,m)+L_t^{\mathrm{mig}}(u)=\frac{\lambda_u}{r(u,a_u^t)}+\frac{\lambda_u\gamma_u}{F_{u,a_u^t}}+c_u I\{h_u^t\neq a_u^t\} \tag{10}$$

服务迁移还会产生特定的迁移成本，本文将迁移成本建模为服务迁移的能源消耗。假设E表示每

迁移 1 bit 数据的能耗，单位为焦耳（J），则用户 u 在 t 时刻的迁移成本 $E_t(u)$ 可以表示为

$$E_t(u)=E\left(\theta_u+\sum_{i=1}^{P}M_{\mathrm{d}}+M_{\mathrm{handoff}}\right)I\left\{h_u^t\neq a_u^t\right\} \tag{11}$$

其中，P 为传输的迭代次数，M_{d} 为脏内存的迭代传输设置的阈值，M_{handoff} 为迭代结束剩余的数据。为了便于计算，本文使用 ε_u 表示使用预复制内存迁移技术增加的数据量，定义为

$$\varepsilon_u=\varepsilon_{\mathrm{left}}+(\varepsilon_{\mathrm{right}}-\varepsilon_{\mathrm{left}})\,\mathrm{beta}\,(\theta_u,d\,(h_u^t,a_u^t)) \tag{12}$$

其中，beta 与式(8)中的定义一致，$\varepsilon_{\mathrm{left}}$ 和 $\varepsilon_{\mathrm{right}}$ 表示增加数据量的左右边界。因此，迁移成本定义为

$$E_t(u)=\varepsilon_u\theta_u EI\left\{h_u^t\neq a_u^t\right\} \tag{13}$$

2.3　在线服务迁移问题构建

考虑到系统有限的迁移成本，系统中所有用户的实际迁移成本 $E_t(u)$ 应保持在预期迁移成本范围之内。假设 E_{budget} 表示系统的平均迁移成本预算，E_u^{avg} 表示用户 u 的平均迁移成本预算，则所有用户的平均迁移成本预算之和应该满足系统的平均迁移成本预算约束，即

$$\sum_{u=1}^{U}E_u^{\mathrm{avg}}\leqslant E_{\mathrm{budget}} \tag{14}$$

由于用户的移动具有不确定性，难以在初始阶段准确确定分配给每个用户的平均迁移成本预算。受文献[16]的启发，本文将分配给用户 u 的平均迁移成本预算 E_u^{avg} 定义为

$$E_u^{\mathrm{avg}}=\frac{\sum_{t=(D-1)T}^{DT}\theta_u L_t^{\mathrm{Qos}}(u)}{\sum_{v=1}^{U}\sum_{t=(D-1)T}^{DT}\theta_v L_t^{\mathrm{Qos}}(v)}E_{\mathrm{budget}} \tag{15}$$

其中，D 代表分配预期迁移成本的次数。系统的迁移成本预算最初被平均分配给每个用户。随后，系统根据 T 个时隙的平均服务时延以及服务对应的虚拟机数据大小 θ_u，自适应地调整各个用户所分配的预期迁移成本，以适应用户移动模式的变化。因此，每个用户的服务迁移问题可以表述为

$$\begin{aligned}&\mathrm{P1}:\min_{a_u^t}\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}L_t^{\mathrm{Qos}}(u)\\&\text{s.t. 式(1)~式(2)}\\&\quad a_u^t\in\{1,2,\cdots,m\},\forall t\in\mathcal{T}\qquad(16)\\&\quad\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E_t(u)\leqslant E_u^{\mathrm{avg}}\qquad(17)\end{aligned}$$

其中，约束式(17)保证实际的平均迁移成本小于平均迁移成本预算。解决这个问题的最优解需要未来 T 个时隙的用户位置信息，这在现实场景中是非常难以获取的。本文利用 Lyapunov 技术[17]将原问题转化为在线问题处理，即只使用用户当前的位置信息求解原问题。该方法通过构造一个成本队列来遵循长期的成本约束式(17)。假设初始队列 $Q_u(0)=0$，队列长度根据实际迁移成本和预期迁移成本 E_u^{avg} 演变为

$$Q_u(t+1)=\max\left[Q_u(t)+E_t(u)-E_u^{\mathrm{avg}},0\right] \tag{18}$$

其中，$Q_u(t)$ 是 t 时段中的队列积压，表示当前迁移成本与长期成本约束的偏差，$Q_u(t)$ 值较大意味着实际迁移成本已经超过了预期成本 E_u^{avg}。

本文将 Lyapunov 函数定义为 $L\left(Q_u(t)\right)=\frac{1}{2}Q_u^2(t)$，表示对超出迁移成本程度的度量，如果 $L\left(Q_u(t)\right)$ 的值很小，则意味着队列积压很小，成本队列拥有很强的稳定性。假设 $\Delta\left(Q_u(t)\right)=L\left(Q_u(t+1)\right)-L\left(Q_u(t)\right)$ 表示单槽 Lyapunov 漂移，可以得到单槽 Lyapunov 漂移的上确界为

$$\begin{aligned}&\Delta\left(Q_u(t)\right)=\frac{1}{2}Q_u^2(t+1)-\frac{1}{2}Q_u^2(t)\leqslant\\&\frac{1}{2}\left(Q_u(t)+E_t(u)-E_u^{\mathrm{avg}}\right)^2-\frac{1}{2}Q_u^2(t)=\\&\frac{1}{2}\left(E_t(u)\right)^2+\frac{1}{2}\left(E_u^{\mathrm{avg}}\right)^2-\\&E_t(u)E_u^{\mathrm{avg}}+Q_u(t)\left(E_t(u)-E_u^{\mathrm{avg}}\right)\leqslant\\&B+Q_u(t)\left(E_t(u)-E_u^{\mathrm{avg}}\right)\end{aligned} \tag{19}$$

其中，$B=\frac{1}{2}\left(E_u^{\max}\right)^2+\frac{1}{2}\left(E_u^{\mathrm{avg}}\right)^2$ 是一个常数，$E_u^{\max}=\max_{t\in\mathcal{T}}E_t(u)$ 表示迁移过程中的最大迁移成本。然后，基于队列稳定性的思想，可以将问题 P1 的目标函数作为一个惩罚函数添加到 Lyapunov 漂移 $\Delta\left(Q_u(t)\right)$ 中，通过最小化每个时隙的 Lyapunov 漂移加惩罚函数的上确界，将问题 P1 转化为在线问题 P1′处理，即

$$\begin{aligned}&\mathrm{P1}':\min_{a_u^t}Q_u(t)E_t(u)+VL_t^{\mathrm{Qos}}(u)\\&\text{s.t. 式(1)~式(2),式(16)}\end{aligned} \tag{20}$$

其中，$V\geqslant 0$ 是一个非负控制参数，表示在每个时隙中强调服务时延的重要性。Lyapunov 技术通过动态地更新迁移成本与服务时延之间的权重（即 $Q_u(t)$），实现在满足长期迁移成本约束的同时，最

小化用户的服务时延。直接求解问题P1′所得到的服务迁移决策忽略了用户之间的资源竞争，因此本文提出了一种针对资源竞争的服务迁移优化方法，以缓解用户之间的资源竞争。

3 针对资源竞争的服务迁移优化方法

为了缓解用户之间的资源竞争，基于博弈论，本节提出了一种OMRC-LG方法。首先，将在线问题P1′建模为服务迁移博弈，并证明了该博弈存在纳什均衡解。然后，提出了一种基于博弈论的分布式方法找到该博弈中的纳什均衡解，实现服务迁移优化。

3.1 基于博弈论的服务迁移

博弈论是研究分布式机制的强大工具，使用博弈论可以使系统中的所有用户都能实现相互满意的解决方案[18]。在博弈中，通常假设每个博弈玩家都是理性的参与者，他们的目的是通过采取合理的策略，以最大化自身的收益。在本文的服务迁移问题中，在给定其他用户的服务迁移决策 $\boldsymbol{a}_{-u}=\left(a_1,\cdots,a_{u-1},a_{u+1},\cdots,a_U\right)$ 情况下，用户 u 期望通过选择一个最优的迁移决策，以最小化自己的决策成本，即 $\min\limits_{a_u\in\{1,\cdots,M\}} Q_u(t)E_t(u)+VL_t^{\mathrm{Qos}}(u)$。为了便于表示，使用 $C_{\boldsymbol{a}_{-u}}\left(a_u\right)$ 来表示决策成本，即

$$C_{\boldsymbol{a}_{-u}}\left(a_u\right)=Q_u(t)E_t(u)+VL_t^{\mathrm{Qos}}(u) \tag{21}$$

因此，服务迁移问题可以表示为一个服务迁移博弈 $\Gamma=\left(\mathcal{U},\left\{Y_u\right\}_{u\in\mathcal{U}},\left\{C_{\boldsymbol{a}_{-u}}\left(a_u\right)\right\}_{u\in\mathcal{U}}\right)$，其中，$\mathcal{U}$ 是移动用户的集合，$Y_u=\{1,2,\cdots,M\}$ 是用户 u 进行服务迁移决策时可选的目标服务器集合。在博弈论中，最重要的概念是纳什均衡[19]，如定义1所示。

定义1 纳什均衡。对于一个服务迁移博弈 Γ，当给定其他用户的服务迁移决策 $\boldsymbol{a}_{-u}=\left(a_1,\cdots,a_{u-1},a_{u+1},\cdots,a_U\right)$ 时，如果没有用户可以通过单方面改变其迁移决策以降低其决策成本，则该博弈达到纳什均衡。在纳什均衡下，对于任意用户 $u\in\mathcal{U}$ 及其可能的迁移决策 $a_u\in Y_u$，有

$$C_{\boldsymbol{a}_{-u}^*}\left(a_u^*\right)\leqslant C_{\boldsymbol{a}_{-u}^*}\left(a_u\right),\forall a_u\in Y_u,u\in\mathcal{U} \tag{22}$$

其中，a_u^* 代表博弈达到纳什均衡时用户 u 的服务迁移决策，$\boldsymbol{a}_{-u}^*$ 表示博弈达到纳什均衡时除了用户 u 之外的剩余用户的服务迁移决策。纳什均衡是博弈论中的一种稳定状态，其中每个参与者都能够获得一个相互满意的解。本文的服务迁移博弈 Γ 是一个拥有完美信息的有限策略博弈，即用户决策信息共享且每个用户的服务迁移决策集合有限。如果该博弈存在纯策略纳什均衡解，则博弈会在有限迭代次数后结束[20]；如果不存在纯策略纳什均衡解，则意味着服务迁移博弈可能无法收敛。可以利用反证法证明服务迁移博弈 Γ 存在纯策略纳什均衡解，证明过程如下。

证明 如果服务迁移博弈 Γ 不存在纯策略纳什均衡，则存在 N（$1\leqslant N\leqslant U$）个用户总是满足不等式(22)，可以不断地更新迁移决策以最小化决策成本。

然而，当 $N=1$ 时，上述论述显然不成立。当其他用户的决策确定时，一个用户的决策成本不能一直下降，否则将与不等式(22)矛盾。

当 $N=2$ 时，假设2个可以不断更新迁移决策的用户为 u 和 v，其他用户不再更新迁移决策。不失一般性，假设在第 $k-1$ 轮迭代时，用户 u 和 v 的迁移决策分别为 $a_u^{k-1}=1$ 和 $a_v^{k-1}=2$。在第 k 轮迭代时，更新的是用户 u 的迁移决策 $a_u^k=3$。根据不等式(22)，则有 $C_{\boldsymbol{a}_{-u}^k}\left(a_u^k=3\right)<C_{\boldsymbol{a}_{-u}^{k-1}}\left(a_u^{k-1}=1\right)$。在第 $k+1$ 轮迭代时，将会执行用户 v 的迁移决策 a_v^{k+1}，这里有3种情况：$a_v^{k+1}=a_u^k$；$a_v^{k+1}=a_u^{k-1}$；$a_v^{k+1}\neq a_u^k$ 且 $a_v^{k+1}\neq a_u^{k-1}$。

对于情况1（$a_v^{k+1}=a_u^k$），在第 k 轮迭代后，用户 u 将决定从服务器2迁移到服务器3，导致服务器3的用户数量增加，根据式(6)，服务器3的计算性能将会下降。用户 v 仍然决定迁移到服务器3，这表明 $a_v^{k+1}=a_u^k=3$ 是用户 v 最终最优的服务迁移决策，不再需要更新迁移决策，博弈结束。

对于情况2（$a_v^{k+1}=a_u^{k-1}$），如果执行 $a_v^{k+1}=a_u^{k-1}$ 决策，用户 v 将决定从服务器2迁移到服务器1，这会导致服务器2的用户数量减少，从而服务器2的计算性能将会提升。因此，在第 $k+2$ 轮迭代时，用户 u 唯一可能更新的决策是 $a_u^{k+2}=2$，否则博弈将结束迭代。类似地，如果执行 $a_u^{k+2}=2$ 决策，则在第 $k+3$ 轮迭代时，用户 v 唯一可能更新的决策是 $a_v^{k+3}=3$。这会导致服务器1的计算性能提升。因此，在第 $k+4$ 轮迭代时，用户 u 可能更新的决策为 $a_u^{k+4}=1$。根据不等式(22)，对于用户 u，有

$$\begin{aligned}&C_{\boldsymbol{a}_{-u}^{k-1}}\left(a_u^{k-1}=1\right)>C_{\boldsymbol{a}_{-u}^k}\left(a_u^k=3\right)>\\&C_{\boldsymbol{a}_{-u}^{k+2}}\left(a_u^{k+2}=2\right)>C_{\boldsymbol{a}_{-u}^{k+4}}\left(a_u^{k+4}=1\right)\end{aligned} \tag{23}$$

这将与不等式(22)矛盾，因此博弈最多会在第k+3轮结束。

对于情况3（$a_v^{k+1} \neq a_u^k$且$a_v^{k+1} \neq a_u^{k-1}$），在执行第k+1轮迭代的a_v^{k+1}决策后，服务器2的用户数量将会减少，导致服务器2的计算性能提升。因此，在第k+2轮迭代时，用户u唯一可能更新的决策是$a_u^{k+2}=2$。在这种情况下，后续的博弈遵循情况2的分析，即情况3是情况2的前一轮迭代开始的情况。

因此，当N=2时，本文分析了在2种情况下最多只能按照情况2进行k+3轮博弈之后结束，不存在2个用户可以不断地更新迁移决策。

当N=3时，假设有3个可以不断更新迁移决策的用户u、v和x，而其他用户不再更新迁移决策。假设在第k−1轮迭代时，用户u、v和x的迁移决策分别为$a_u^{k-1}=1$、$a_v^{k-1}=2$和$a_x^{k-1}=3$。不失一般性，假设在第k轮迭代时，更新的是用户u的迁移决策$a_u^k=4$。类似N=2的分析，在接下来的博弈中，对于用户u，有

$$\begin{aligned}&C_{\boldsymbol{a}_{-u}^{k-1}}\left(a_u^{k-1}=1\right)>C_{\boldsymbol{a}_{-u}^{k}}\left(a_u^{k}=4\right)>\\&C_{\boldsymbol{a}_{-u}^{k+3}}\left(a_u^{k+3}=3\right)>C_{\boldsymbol{a}_{-u}^{k+6}}\left(a_u^{k+6}=2\right)>\\&C_{\boldsymbol{a}_{-u}^{k+9}}\left(a_u^{k+9}=1\right)\end{aligned}\tag{24}$$

这与不等式(22)矛盾，因此博弈会在有限轮数内结束。因此，不存在3个用户可以不断地更新迁移决策。由此可以推广到N=U的情况，即不存在N（1≤N≤U）个用户总是满足不等式(22)可以不断地更新迁移决策。

所以，本文的服务迁移博弈Γ存在一个纯策略纳什均衡解。根据有限改进定理[21]，一个拥有纯策略纳什均衡解的有限策略博弈会在有限迭代过程中达到纳什均衡。基于此，本文设计了一种分布式方法找到纳什均衡解。证毕。

3.2　基于博弈论的分布式方法

根据博弈的时间序列，博弈可以分为动态博弈和静态博弈。在动态博弈中，每个用户在制定策略时存在顺序约束，后续用户可以观察到前面用户的决策信息并据此制定迁移决策。然而，并非每个时隙中所有用户都需要进行服务的迁移，直接采用动态博弈可能导致不必要的等待时间。静态博弈是指所有用户同时制定迁移决策，之后由集中式控制器选择其中一个用户来确认其迁移决策，然后所有用户再次制定新的服务迁移决策，直到所有用户的迁移决策不再更新。由于静态博弈一次只更新一个用户的服务迁移决策，迁移决策的信息可能需要在边缘服务器和集中式控制器之间频繁地传输，从而耗费较长时间。因此，本文将博弈论中的动态博弈和静态博弈结合起来设计了一种分布式方法，其执行过程如图2所示，伪代码如算法1所示。

算法1　针对资源竞争的服务迁移优化方法

输入 $\{C_m\}_{m\in\mathcal{M}}$、$\{S_u\}_{u\in\mathcal{U}}$、$\{Q_u(t)\}_{u\in\mathcal{U}}$、$\{E_u^{\text{avg}}\}_{u\in\mathcal{U}}$、$\boldsymbol{H}^t$、$E_{\text{budget}}$、$V$

输出 $\boldsymbol{A}^*=\{a_1^*,a_2^*,\cdots,a_U^*\}$

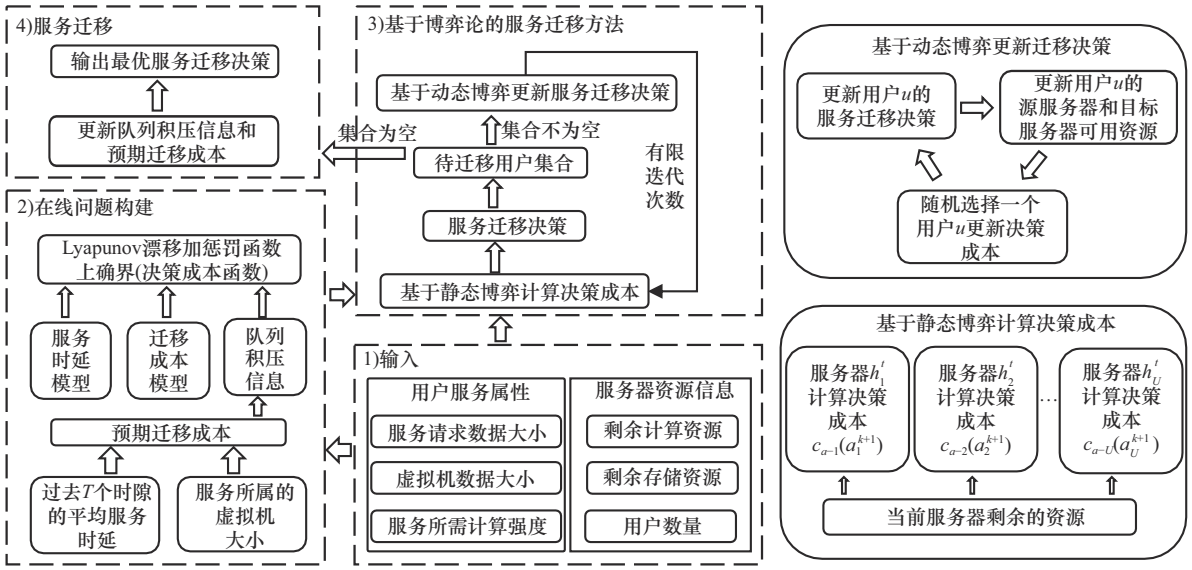


图2　针对资源竞争的服务迁移优化方法的执行过程

1) for 每个时隙 t do：
2) 初始化迭代轮数 k=0，服务迁移决策 $\boldsymbol{A}^k=\left\{h_1^t,h_2^t,\cdots,h_U^t\right\}$
3) 根据式(25)和式(26)计算每个服务器的可用资源 F_m^{remain} 和 R_m^{remain}
4) end for
5) do：
6) ToMigrate = [ ]
7) for each $u\in\mathcal{U}$ do：// 边缘服务器制定服务迁移决策
8) 根据式(27)计算迁移到不同服务器的决策成本 $C\left(a_u^{k+1}\right)$
9) 找到决策成本最小的服务器作为迁移决策 a_u^{k+1}
10) if $a_u^{k+1}\neq a_u^k$ then：
11) ToMigrate ← ToMigrate $\cup\{u\}$
12) end if
13) end for
14) for 集合 ToMigrate 随机选择一个用户 u do：// 集中式控制器协调迁移决策
15) 根据式(27)更新用户 u 的决策成本 $C\left(a_u^{k+1}\right)$
16) 选择决策成本最小的服务器更新迁移决策 a_u^{k+1}
17) 根据式(25)和式(26)更新决策的源服务器和目标服务器的可用资源 F_m^{remain} 和 R_m^{remain}
18) end for
19) $\boldsymbol{A}^{k+1}=\left\{a_1^{k+1},a_2^{k+1},\cdots,a_U^{k+1}\right\}$
20) k←k+1
21) while ToMigrate == NULL
22) end while
23) $\boldsymbol{A}^*=\boldsymbol{A}^k$
24) $\boldsymbol{H}^{t+1}=\boldsymbol{A}^k$
25) 根据式(18)更新每个用户的成本队列 $Q_u(t+1)$
26) 根据式(15)更新每个用户给的预期迁移成本 E_u^{avg}
27) Return $\boldsymbol{A}^*$

在每个时隙开始时，首先将每个用户的服务迁移决策初始化为“不迁移”决策（第 2 行），然后根据式(25)和式(26)来计算各个服务器的可用计算资源 F_m^{remain} 和存储资源 R_m^{remain}（第 3 行）。每个边缘服务器计算相应的资源信息之后，可以通过集中式控制器协调信息同步更新到每个服务器。

$$F_m^{\text{remain}}=F_m-\sum_{u=1}^{U}f_{u,m}I\left\{a_u=m\right\},\forall m\in\mathcal{M} \quad (25)$$

$$R_m^{\text{remain}}=R_m-\sum_{u=1}^{U}\theta_u I\left\{a_u=m\right\},\forall m\in\mathcal{M} \quad (26)$$

在进行一些初始化操作后，算法 1 进入静态博弈阶段，边缘服务器独立地制定用户的服务迁移决策。每个边缘服务器计算当前运行的服务迁移到不同边缘服务器的决策成本（第 7~8 行）。根据式(10)、式(13)、式(18)和式(20)，在满足资源限制要求下，决策成本的计算式为

$$C\left(a_u^{k+1}\right)=\begin{cases}Q_u(t)\varepsilon_u\theta_u E+V\left(L_t^{\text{cm}}\left(u,a_u^{k+1}\right)+L_t^{\text{cp}}\left(u,a_u^{k+1}\right)+c_u\right) & ,a_u^{k+1}=a_u^k,a_u^{k+1}\neq h_u^t\\ V\left(L_t^{\text{cm}}\left(u,a_u^{k+1}\right)+L_t^{\text{cp}}\left(u,a_u^{k+1}\right)\right) & ,a_u^{k+1}=a_u^k,a_u^{k+1}=h_u^t\\ Q_u(t)\varepsilon_u\theta_u E+V\left(L_t^{\text{cm}}\left(u,a_u^{k+1}\right)+\dfrac{L_t^{\text{cp}}\left(u,a_u^{k+1}\right)}{1+d_{a_u^{k+1}}}+c_u\right) & ,a_u^{k+1}\neq a_u^k,a_u^{k+1}\neq h_u^t\\ V\left(L_t^{\text{cm}}\left(u,a_u^{k+1}\right)+\dfrac{L_t^{\text{cp}}\left(u,a_u^{k+1}\right)}{1+d_{a_u^{k+1}}}\right) & ,a_u^{k+1}\neq a_u^k,a_u^{k+1}=h_u^t\end{cases} \quad (27)$$

如果不满足资源限制要求，则决策成本设置为无穷大。用户会选择决策成本最小的服务器作为当前迁移决策 a_u^{k+1}（第 9 行）。如果 $a_u^{k+1}\neq a_u^k$，边缘服务器会将用户 u 的服务迁移决策 a_u^{k+1} 发送到集中式控制器以获取决策更新机会，用户 u 加入待迁移集合 ToMigrate 中（第 10~12 行）。

然后，算法进入动态博弈阶段，集中式控制器按照随机顺序依次更新待迁移集合 ToMigrate 中用户的服务迁移决策（第 14~18 行），在选择某个用户更新其服务迁移决策后，会更新该决策相关的服务器可用资源信息。因为一个用户的决策更新只会影响对应的源服务器和目标服务器的资源更新，所以动态博弈中的每一次决策更新可以快速地进行。当待迁移集合 ToMigrate 的用户全部更新了迁移决策

后，算法进行下一轮静态博弈（第19~20行）。

根据有限改进性质，经过有限轮博弈之后，所有的用户将不再更新其迁移决策，即待迁移集合为空（第21行），算法结束。此时，将$\boldsymbol{A}^k$作为最优的服务迁移决策$\boldsymbol{A}^*$（第23行），并更新下一时刻的服务提供信息$\boldsymbol{H}^{t+1}$（第24行）以及每个用户的队列积压$Q_u(t+1)$（第25行）和预期迁移成本E_u^{avg}（第26行）。最后，输出最优的服务迁移决策$\boldsymbol{A}^*$（第27行）。

在初始化阶段，算法1主要的计算开销是计算服务器的剩余资源信息，时间复杂度为$O(M)$。当算法进入静态博弈阶段后，边缘服务器计算正在运行的服务迁移到不同服务器的决策成本，时间复杂度为$O(M)$。随后，算法进入动态博弈的阶段，由控制器集中协调用户的服务迁移决策。假设$\left|\text{ToMigrate}\right|_{\max}$表示迭代过程中最大的待迁移用户集合大小，每次更新一个用户的服务迁移决策时，会更新与决策相关的源服务器和目标服务器的资源信息，因此，动态博弈阶段的复杂度上限为$O\left(2\left|\text{ToMigrate}\right|_{\max}\right)$。假设$K$表示经历的总迭代次数，则总的时间复杂度为$O\left(M+KM+2K\left|\text{ToMigrate}\right|_{\max}\right)$。

4　实验

为了评估本文方法的可行性和有效性，本节将本文方法与多个基准方法进行了比较。以下将介绍仿真实验的数据集、实验设置和基准方法，最后对实验结果进行分析。

4.1　数据集

为了模拟用户数量众多的MEC场景，本文需要足够多的用户数据，这些数据应当包含用户在长时间内按固定时间间隔记录的位置信息。因此，本文在实验中采用了2个数据集：上海电信公司的基站数据集[7]和上海出租车跟踪数据集。上海电信公司的基站数据集包含3 233个基站的确切位置信息；上海出租车跟踪数据集记录了2007年2月20日在上海行驶的4 328辆出租车的轨迹，每条出租车轨迹都包含具体的位置信息，每0.5~1 min记录一次。在数据处理过程中，本文从上海出租车跟踪数据集中剔除了数据量不足的记录，最终保留了3 634辆出租车的完整轨迹。接下来，本文使用K-Means算法选择用户轨迹最密集的区域，并将距该区域中心20 km×20 km的范围作为本文实验的MEC场景。为确保基站的分布相对均匀，本文同样使用K-Means算法从上海电信公司的基站数据集中选取了位于该区域内的1 712个基站中的100个位置作为本文实验的边缘服务器位置。最后，本文计算了该区域内不同时间段的出租车数量，选择了上午7点到9点之间的早高峰时期的数据作为本文实验的背景，并选取该时间段中的1 275辆出租车作为本文实验的移动用户集，用于模拟MEC场景中的移动用户。每个用户以1 min为时隙长度，记录了120个时隙的移动轨迹。

4.2　实验设置和基准方法

本文采用Java IDE IntelliJ IDEA作为实验的仿真软件，版本为IntelliJ IDEA 2021.1.3。实验在一台华硕笔记本计算机上完成，该计算机的硬件配置为：CPU型号为AMD Ryzen 7 5800H with Radeon Graphics3.20 GHz，内存为16 GB，系统为Windows11。实验中所涉及的参数[12,16,22]如表1所示，其中，MIPS（million instructions per second）表示计算机每秒执行的百万指令数。

表1　　参数设置

参数	值
边缘服务器数目M	100
移动用户的数目U	[200,1 200]
边缘服务器的计算资源F_m/MIPS	[30 000,60 000]
边缘服务器的存储资源R_m/GB	[8,12]
边缘服务器的性能退化因子d_m	[0.1,0.4]
信道带宽W/MHz	20
噪声功率σ_u^2/W	2×10^{-13}
传输功率P_u/W	0.5
复衰落向量$\boldsymbol{h}$	圆对称复高斯随机变量
服务的虚拟机数据大小θ_u/MB	[400,600]
服务请求的数据大小λ_u/MB	[0.1,0.3]
服务请求所需的计算资源$f_{u,m}$/MIPS	[20 000,40 000]
服务请求的计算强度γ_u/(cycle·bit^{-1})	[800,1 600]
迁移时延c_u/ms	[20,50]
单位数据迁移能耗E/J	1.2×10^{-7}
实时迁移增加的数据量ε_u	[1.0,1.5]
控制参数V	10^4
预期迁移成本E_u^{avg}/J	[40,240]
初始队列积压$Q_u(0)$	0
时隙的长度τ/s	60

本文实验所采用的评价指标介绍如下。

平均服务时延：资源竞争会导致服务器计算性能下降，从而增加服务器上所有服务的服务时延。通过对比不同方法下的平均服务时延，可以清晰地看出各个方法的优劣。

平均迁移成本：指服务迁移策略所需的能源消耗。本文的重点是确保服务迁移决策在满足预期迁移成本约束的同时，最小化平均服务时延，因此关注实际迁移成本是否低于预期迁移成本。

决策求解所需迭代次数：指集中式控制器与边缘服务器在求解决策时所需的交互次数。迭代次数越少，则所需的交互次数就越少，在求解速度上更快。

本节将OMRC-LG方法与5个基准方法以及OMRC-LG方法的2个变种方法进行了对比实验。

不迁移方法（简称NM方法）。这种方法不迁移服务，因此，它可以最大限度地降低迁移成本。

一直迁移方法（简称AM方法）。这种方法不关心迁移成本，总是将服务迁移到距离用户最近的服务器。

基于MDP的分布式方法（简称MDP方法）[5]。该方法以用户的服务时延和迁移成本组合为优化目标，使用MDP建模以求解服务迁移决策。在模拟实验中，每个用户独立应用该方法制定迁移决策，当迁移决策出现资源冲突时，则选择附近的边缘服务器迁移。

基于Lyapunov技术和粒子群优化的集中式方法（简称PSO方法）[12]。该方法的优化目标是最小化服务时延和迁移成本的组合。为了确保在限定时间内制定出服务迁移决策，在模拟实验中，初始粒子群中有10%的粒子被初始化为不迁移决策，并将运行时间限制在1 min之内。

采用启发式方法解决资源竞争的分布式方法（简称SMRC方法）[13]。用户在制定服务迁移决策时，首先发送预迁移请求，随后在获得候选服务器的许可后才能进行服务迁移。候选服务器根据服务所需的资源量决定是否接受迁移请求。该方法旨在解决资源竞争造成的边缘服务器过载问题。

仅Lyapunov方法（简称OL方法）。只使用Lyapunov技术求解每个用户的服务迁移决策，当遇到资源超载时，则选择附近的边缘服务器迁移。

只使用静态博弈方法（简称OMRC-LGS方法）。使用Lyapunov技术求解每个用户的服务迁移决策，然后使用静态博弈协调每个用户的服务迁移决策。

为了验证本文方法在缓解用户之间的资源竞争方面的有效性，本文实验从不同用户数量、不同性能退化因子和不同预期迁移成本3个维度，分析了上述8种方法在平均服务时延上的表现，并评估了这些方法是否能确保实际迁移成本低于预期迁移成本。此外，本文实验还比较了OMRC-LG与OMRC-LGS方法在求解服务迁移决策时所需迭代次数的差异，以突出OMRC-LG方法在求解速度上的优势。

4.3 实验结果分析

4.3.1 用户数量对平均服务时延和平均迁移成本的影响

本节实验从不同用户数量探究了8种方法在平均服务时延和平均迁移成本上的表现，用户数量从200个逐步增加到1 200个，每次增加200个用户。实验中设置预期迁移成本为80 J，性能退化因子为0.2。图3和图4分别展示了用户数量对平均服务时延和平均迁移成本的影响。

从图3可以看到，在平均服务时延方面，当用户数量为200个时，除了NM方法之外，其他方法的表现接近，均在0.22 s左右。AM方法实现了最低的平均服务时延，为0.187 s，这是由于当用户数量较少时，用户之间的资源竞争相对较少，服务迁移到用户附近可以获得更低的服务时延。当用户数量相同时，NM方法的平均服务时延最高，这是因为NM方法依赖于初始的服务放置位置，如果服务放置不当，将导致部分服务器资源高度利用，计算速率下降，进而服务时延增加。随着用户数量不断增加，所有方法的平均服务时延都有不同程度的上升。除AM和NM方法之外，PSO方法的平均服务时延增加最为明显，与用户数量为200个时相比，最终增加了3.29倍。这是因为集中式方法在用户数量增加时，求解空间不断扩大，难以在有限时间内找到最优解。OL和MDP方法由于忽略了用户之间的资源竞争，导致其平均服务时延超过了本文方法。在考虑了资源竞争的方法中，OMRC-LG、OMRC-LGS和SMRC方法的平均服务时延增加幅度较小。具体而言，OMRC-LG和OMRC-LGS方法的平均服务时延相对于用户数量为200个时最终增加了约2.25倍，而SMRC方法增加了2.73倍。这表明，本文方法在面对大量用户时，能够更有效地缓解资源竞争，减缓服务时延的增加。

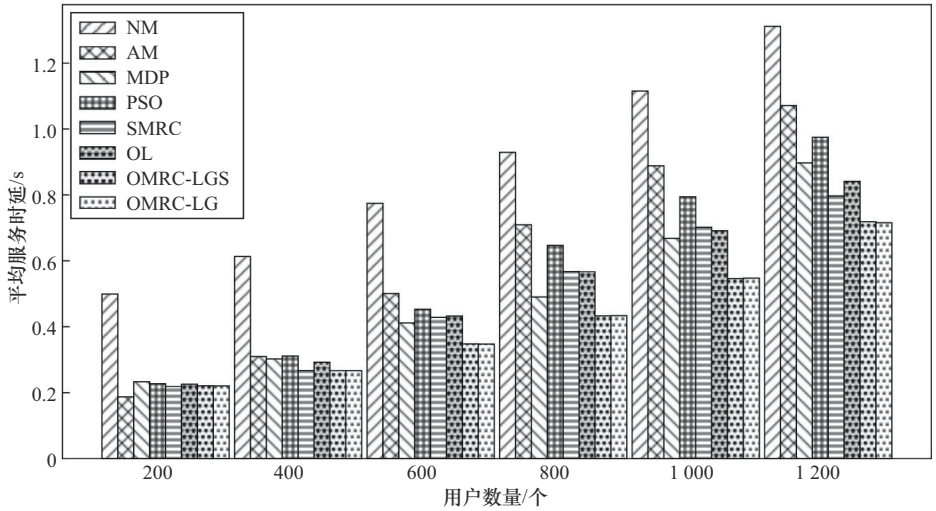


图3　用户数量对平均服务时延的影响

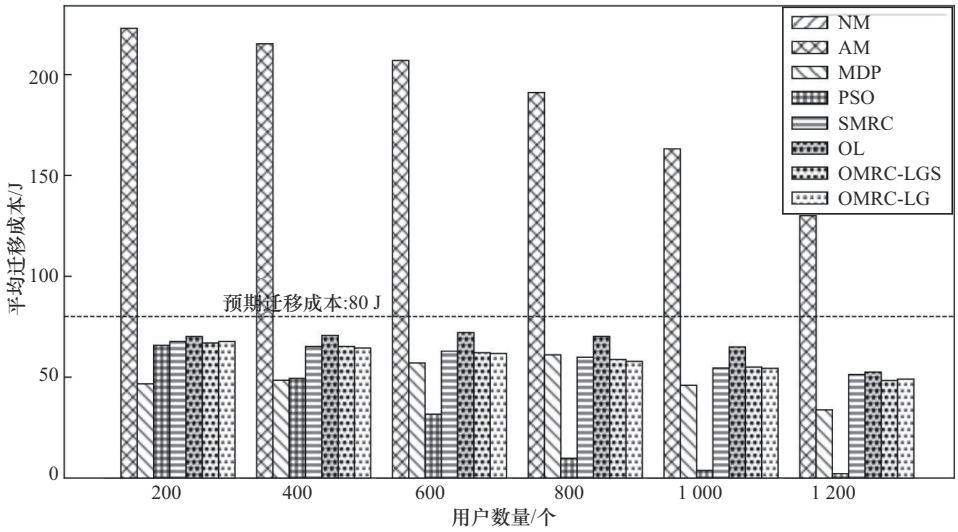


图4　用户数量对平均迁移成本的影响

在平均迁移成本方面的实验结果显示，NM方法的平均迁移成本为0，而AM方法的平均迁移成本最高。其他方法（包括OMRC-LG、OMRC-LGS、OL、PSO、SMRC和MDP方法）的平均迁移成本均小于预期迁移成本80 J。然而，随着用户数量增加，各种方法的平均迁移成本均有所降低。这种现象可能由以下几个原因引起：首先，当用户数量增加时，计算时延成为主导因素，而通信时延的影响相对较小，因此服务迁移的频率可能会降低；其次，由于服务器资源有限，当用户数量较多时，服务在边缘服务器之间频繁迁移的情况可能会受到限制。PSO方法的平均迁移成本降低最为显著，这可能是随着用户数量的不断增加，问题的求解空间不断扩大，导致在限定时间内难以找到最优解。因此，PSO方法更倾向于选择初始粒子的不迁移决策，从而导致平均迁移成本不断降低。

图5展示了OMRC-LG和OMRC-LGS方法在求解服务迁移决策时所需的平均迭代次数，其中百分数表示OMRC-LG方法所需的迭代次数占OMRC-LGS方法所需迭代次数的百分比。这2种方法在平均服务时延和平均迁移成本上的表现非常接近，但在平均迭代次数方面，OMRC-LG方法的平均迭代次数仅为OMRC-LGS方法的10%左右。这意味着，OMRC-LG方法在求解服务迁移决策时，与集中式控制器和边缘服务器之间的交互次数更少，求解速度更快。因此，OMRC-LG方法在效率方面具有更大的优势。

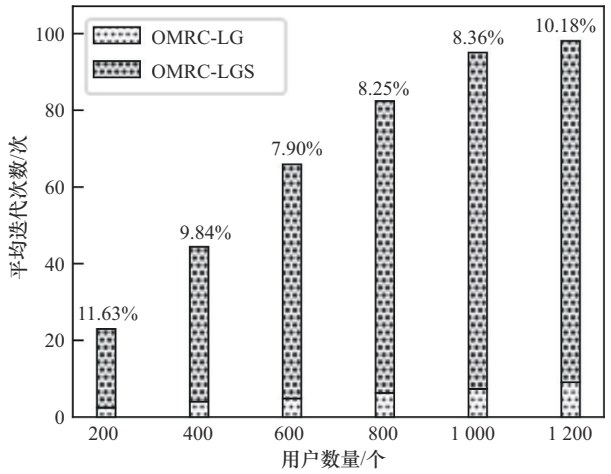


图5　OMRC-LG和OMRC-LGS方法在平均迭代次数上的对比

4.3.2　性能退化因子对平均服务时延和平均迁移成本的影响

性能退化因子表示用户共享I/O资源时计算速率下降的程度。该值增加意味着服务器同时运行多个服务所需的计算时间更长。本节实验研究了在不同性能退化因子下，8种方法在平均服务时延和平均迁移成本上的表现，性能退化因子从0.1逐渐增加到0.4，每个用户的预期迁移成本设为80 J，用户数量为600个。图6和图7分别展示了性能退化因子对平均服务时延和平均迁移成本的影响。

从图6中可以看出，在平均服务时延方面，当性能退化因子较小时，除NM方法的平均服务时延相对较高，达到了0.53 s外，其他7种方法的平均服务时延较为接近，均在0.3 s左右。随着性能退化因子的增加，各种方法的平均服务时延都有所增加。AM方法的平均服务时延增加最为明显，与性能退化因子为0.1时相比，最终增加了约12.88倍，甚至超过了NM方法的平均服务时延。这表明，当用户之间资源竞争较为严重时，盲目地将服务迁移到用户附近可能会导致更高的平均服务时延。OL和MDP方法由于忽略了用户之间

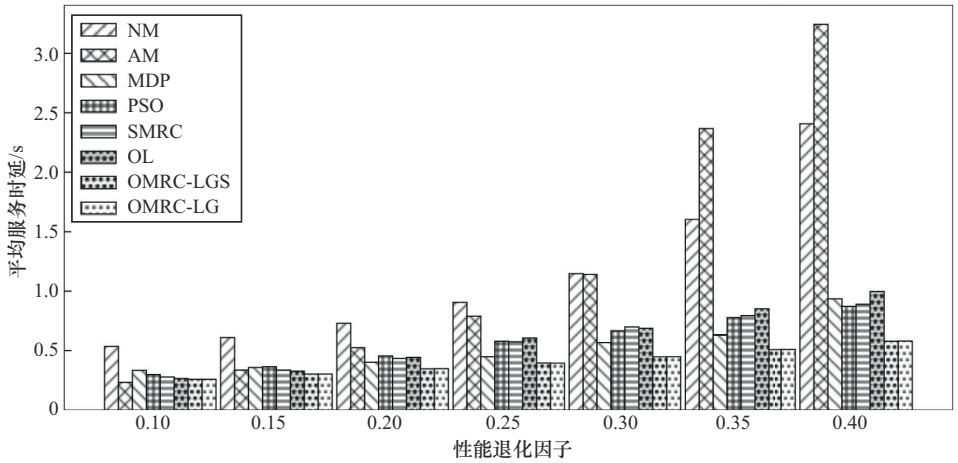


图6　性能退化因子对平均服务时延的影响

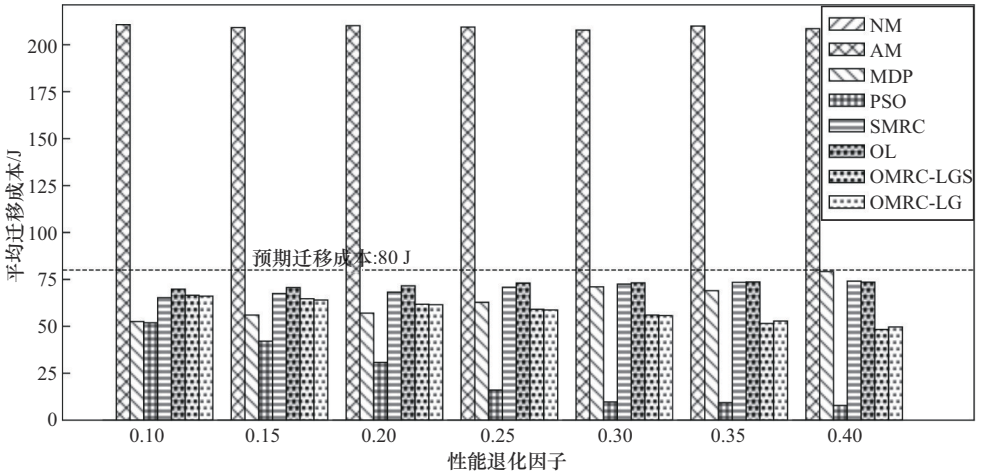


图7　性能退化因子对平均迁移成本的影响

的资源竞争，平均服务时延的增加幅度超过了本文方法。PSO 方法的平均服务时延也超过了本文方法，这是因为随着性能退化因子的不断增加，搜索最优解的方向减少，导致在限定时间内搜索最优解的速率变慢。在考虑了资源竞争的方法中，SMRC 方法的平均服务时延增加了 2.19 倍，而 OMRC-LG 与 OMRC-LGS 方法增加了 1.24 倍。这表明通过引入博弈论的方法来求解用户的服务迁移决策，可以有效缓解用户之间的资源竞争，降低平均服务时延。

在平均迁移成本方面，除了 AM 方法外，其余方法的迁移成本均低于预期迁移成本 80 J。NM 方法由于不迁移服务，迁移成本为 0。随着性能退化因子增加，OL 和 SMRC 方法的平均迁移成本变化不大。PSO、OMRC-LG 和 OMRC-LGS 方法的平均迁移成本出现了一定程度的下降。这是因为当用户之间资源竞争加剧时，服务迁移受通信时延的影响比例减少，服务倾向于迁移到服务数量较少的边缘服务器上，导致服务迁移频率的下降。对于 PSO 方法，随着性能退化因子的增加，搜索最优解的方向变得更少，导致在限定时间内找到最优解的速率变慢，结果倾向于保持初始粒子的不迁移决策，从而降低了平均迁移成本。MDP 方法的平均迁移成本出现了一定程度的增加，这可能是因为这种策略忽略了用户之间的资源竞争，当性能退化因子增加时，难以找到实际的最优解，从而导致服务进行了多次迁移。

4.3.3　预期迁移成本对平均服务时延的影响

本节实验分析了不同预期迁移成本对平均服务时延的影响，预期迁移成本从 40 J 开始，每次增加 40 J，直到 240 J，用户数量为 600 个，性能退化因子为 0.2。图 8 展示了预期迁移成本对平均服务时延的影响。

从图 8 中可以看出，随着预期迁移成本的增加，AM 和 NM 方法由于遵循固定的规则，平均服务时延并没有明显的变化。PSO 方法也没有出现明显的变化，可能是因为增加迁移成本预算没有对求解空间起到缩小的作用。OL 方法甚至出现了一定程度的增加，这表明如果在忽略了用户资源竞争的情况下盲目迁移服务，可能会导致更高的服务时延。除此之外，SMRC、MDP、OMRC-LG 和 OMRC-LGS 方法的平均服务时延都出现了下降。相对而言，OMRC-LG 和 OMRC-LGS 方法的平均服务时延处于最低，这表明了本文方法能够更有效地利用迁移成本预算进行迁移，降低服务时延。

5　结束语

本文提出了一种针对资源竞争的服务迁移优化方法。该方法将服务迁移问题建模为迁移成本约束下的最优化问题，并利用 Lyapunov 技术将其转化

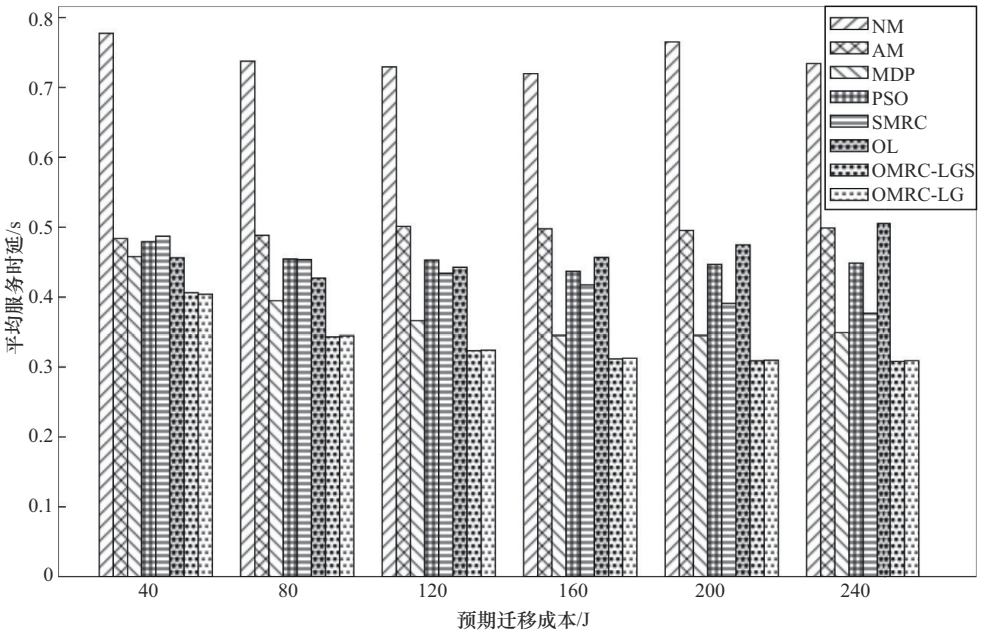


图 8　预期迁移成本对平均服务时延的影响

为在线问题处理。在此基础上，本文提出了一种基于博弈论的分布式方法求解在线问题，通过共享用户服务迁移决策，及时获取准确的服务器可用资源信息，尽可能地缓解了用户之间的资源竞争。目前的工作主要围绕多用户资源竞争的服务迁移优化，在未来的工作中将考虑如何促进用户之间的协作进行服务迁移，提高边缘服务器的资源利用率。

参考文献：

[1] YOUSEFPOUR A, FUNG C, NGUYEN T, et al. All one needs to know about fog computing and related edge computing paradigms: a complete survey[J]. Journal of Systems Architecture, 2019, 98: 289-330.

[2] LI C L, ZHANG Y, LUO Y L. Flexible heterogeneous data fusion strategy for object positioning applications in edge computing environment[J]. Computer Networks, 2022, 212: 109083.

[3] 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, 2020, 52(5): 1-33.

[4] LEE K, KIM J, KWON I H, et al. Impact of secure container runtimes on file I/O performance in edge computing[J]. Applied Sciences, 2023, 13(24): 13329.

[5] WANG S Q, URGAONKAR R, ZAFER M, et al. Dynamic service migration in mobile edge computing based on Markov decision process[J]. IEEE/ACM Transactions on Networking, 2019, 27(3): 1272-1288.

[6] SHAMSADINI A, ENTEZARI-MALEKI R. Time-aware MDP-based service migration in 5G mobile edge computing[C]//Proceedings of the 2022 27th International Computer Conference, Computer Society of Iran (CSICC). Piscataway: IEEE Press, 2022: 1-5.

[7] WANG S G, GUO Y, ZHANG N, et al. Delay-aware microservice coordination in mobile edge computing: a reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2021, 20(3): 939-951.

[8] LI C L, ZHANG Y, GAO X, et al. Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing[J]. Journal of Parallel and Distributed Computing, 2022, 166: 15-31.

[9] MA H R, ZHOU Z, CHEN X. Leveraging the power of prediction: predictive service placement for latency-sensitive mobile edge computing[J]. IEEE Transactions on Wireless Communications, 2020, 19(10): 6454-6468.

[10] SHI Y, YI C Y, WANG R, et al. Service migration or task rerouting: a two-timescale online resource optimization for MEC[J]. IEEE Transactions on Wireless Communications, 2024, 23(2): 1503-1519.

[11] LIANG Z Z, LIU Y, LOK T M, et al. Multi-cell mobile edge computing: joint service migration and resource allocation[J]. IEEE Transactions on Wireless Communications, 2021, 20(9): 5898-5912.

[12] ZHOU X B, GE S X, QIU T, et al. Energy-efficient service migration for multi-user heterogeneous dense cellular networks[J]. IEEE Transactions on Mobile Computing, 2023, 22(2): 890-905.

[13] DUAN J R, REN K, ZHOU W, et al. A service migration method for resource competition in mobile edge computing[C]//Proceedings of the 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC). Piscataway: IEEE Press, 2021: 1-8.

[14] ZHANG Q, GUI L, HOU F, et al. Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN[J]. IEEE Internet of Things Journal, 2020, 7(4): 3282-3299.

[15] SATPATHY A, SAHOO M N, MISHRA A, et al. A service sustainable live migration strategy for multiple virtual machines in cloud data centers[J]. Big Data Research, 2021, 25: 100213.

[16] OUYANG T, RUI L, XU C, et al. Adaptive user-managed service placement for mobile edge computing: an online learning approach[C]//Proceedings of the IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 1468-1476.

[17] NEELY M J. Stochastic network optimization with application to communication and queueing systems[M]. Berlin: Springer, 2010.

[18] BARRON E N. Game theory: an introduction[M]. New Jersey: John Wiley & Sons, 2024.

[19] OSBORNE M J, RUBINSTEIN A. A course in game theory[M]. Cambridge: MIT Press, 1994.

[20] KUHN H W. Extensive games and the problem of information[J]. Contributions to the Theory of Games, 1953, 2(28): 193-216.

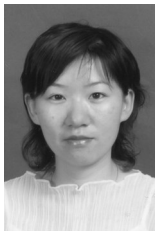
[21] HU G S, JIA Y J, CHEN Z C. Multi-user computation offloading with D2D for mobile edge computing[C]//Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM). Piscataway: IEEE Press, 2018: 1-6.

[22] XU M X, ZHOU Q H, WU H M, et al. PDMA: probabilistic service migration approach for delay-aware and mobility-aware mobile edge computing[J]. Software: Practice and Experience, 2022, 52(2): 394-414.

[作者简介]

王海艳（1974– ），女，江苏东台人，博士，南京邮电大学教授，主要研究方向为服务计算、可信计算、大数据应用与云计算技术、隐私保护技术等。

张霖（1998– ），男，湖南怀化人，南京邮电大学硕士生，主要研究方向为边缘计算。



骆健（1976– ），女，江西赣州人，南京邮电大学副教授，主要研究方向为服务计算、可信计算、服务推荐等。