一种面向边缘计算环境的去中心化服务请求分发方法

吴洪越[1)] 陈志伟[1)] 石博文[1)] 邓水光[2)] 陈世展[1)] 薛 霄[1)] 冯志勇[1)]

[1)](天津大学智能与计算学部 天津 300350)

[2)](浙江大学计算机科学与技术学院 杭州 310027)

摘 要 边缘计算模式的出现使得软件服务可以部署在距离用户较近的边缘服务器上,从而减小服务调用过程中产生的数据传输开销,并提升服务质量.然而,由于计算和存储等资源的限制,边缘服务器通常只能部署有限数量的服务实例,无法满足复杂多样的服务请求,加之边缘计算环境下服务请求分布不均导致了边缘服务器之间负载的不均衡性.因此,边缘计算环境下需要对服务请求进行再分配来满足不同的服务请求并优化服务供应.如何合理地将服务请求分发到合适的边缘服务器以优化系统的负载均衡并提升服务质量成为亟待解决的关键问题.现有的研究方法通常采用集中式的方法来解决该问题,存在单点故障等严重缺陷.因此,本文基于博弈理论设计了一种面向边缘计算环境的去中心化服务请求分发方法.该方法将边缘服务器间的服务请求分发问题建模为分布式的非合作博弈模型,通过多轮次的博弈和竞争达到系统的Nash均衡状态,以此获取服务请求的分发策略.实验结果表明,本文所提出的方法可以有效优化边缘服务器之间的负载均衡,降低服务的响应时间,并且随着系统规模的扩大表现出良好的可扩展性.

关键词 服务计算;边缘计算;请求分发;博弈;去中心化

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2023.00987

Decentralized Service Request Dispatching for Edge Computing Systems

WU Hong-Yue[1)] CHEN Zhi-Wei[1)] SHI Bo-Wen[1)] DENG Shui-Guang[2)]

CHEN Shi-Zhan[1)] XUE Xiao[1)] FENG Zhi-Yong[1)]

[1)](*College of Intelligence and Computing, Tianjin University, Tianjin* 300350)

[2)](*College of Computer Science and Technology, Zhejiang University, Hangzhou* 310027)

Abstract Edge computing paradigm enables software services to be deployed on edge servers that are close to users, thereby reducing the data transmission costs produced during service invocation processes and improving the quality of services. However, due to the limited computing and storage resources, individual edge servers can typically be deployed with only a limited number of service instances and thus cannot satisfy diverse service requests. Moreover, service requests are distributed unevenly in edge computing systems, which leads to an imbalance in the load of edge servers. To solve these problems, a collaboration mechanism has been proposed to bring neighboring edge servers together as a cohort to provide services for their users collaboratively. In this mechanism, on receiving service requests, edge servers re-dispatch them to appropriate servers with the objective to satisfy these diverse service requests and to optimize their quality. Thus, how to reasonably dispatch service requests to appropriate edge servers to optimize the

收稿日期:2022-03-31;在线发布日期:2022-10-08.本课题得到国家自然科学基金重点基金(61832014,62032016)、国家自然科学基金(61972276,62102281,U20A20173,62125206)、国家重点研发计划(2021YFF0900800)、山东省智能建筑技术重点实验室基金(SDIBT202001)、浙江省重点研发项目(2022C01145)资助.**吴洪越**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为服务计算、边缘计算、移动云计算.E-mail:hongyue.wu@tju.edu.cn.**陈志伟**,硕士研究生,主要研究方向为服务计算、边缘计算.**石博文**,硕士研究生,主要研究方向为服务计算、边缘计算.**邓水光**,博士,教授,主要研究领域为服务计算、边缘计算、流程管理、大数据.**陈世展**(通信作者),博士,副教授,主要研究方向为服务计算、软件生态系统挖掘与分析.E-mail:shizhan@tju.edu.cn.**薛 霄**,博士,教授,主要研究方向为服务计算、群体智能、计算实验.**冯志勇**,博士,教授,主要研究领域为知识工程、服务计算、计算机认知.

load balancing among edge computing systems as well as to improve the quality of services has become a key problem to be solved. Existing research methods usually adopt centralized methods to solve this problem, which assume that there is a central node in each edge computing system that can get the global information of the system such as the available resources of all edge servers, service execution status, arriving service requests and network conditions between edge servers, and use this information to calculate the optimal request dispatching strategy. However, centralized methods rely heavily on central nodes and thus have serious defects including single point of failure, etc. Therefore, this paper utilizes decentralized mechanism for service request dispatching in edge computing systems. In decentralized frameworks, due to the lack of central nodes, it is hard to get the global real-time information of the system. Hence, centralized methods are not available for decentralized systems, and edge servers can only obtain the status and request information of each other through continuous communication. In this paper, we propose a game theory-based method to solve this problem. Service request dispatching among edge servers is modeled as a distributed non-cooperative game, where edge servers, as the game participants, iteratively communicate with each other and make their own decisions for their received requests according to the status and decisions of each other, with the aim to minimize the average response time of their received service requests. Through multiple rounds of mutual game and competition, these edge servers can finally achieve the Nash equilibrium state, thereby obtaining the optimal service request dispatching strategies. We present the game model and theoretically prove that it can achieve the Nash equilibrium. Moreover, we split service response time to data transmission time and service execution time and give the calculation method of them based on Shannon's Theorem and Queuing Theory, respectively. The experimental results show that compared with benchmark methods, the proposed method can greatly reduce the response time of services, effectively optimize the load balancing between edge servers and perform good scalability as the scale of the system increases.

Keywords service computing; edge computing; request dispatching; game; decentration

1 引 言

近年来,随着移动设备的普及和无线通讯技术的迅速发展,移动数据流量持续高速增长.爱立信公司在其发布的移动报告[1]中指出,全球移动数据流量正以每年 46%的增速迅猛增长,截至 2021 年 3 月已经高达 66 EB 每月.数据流量的迅速增长给核心网(Core Network)和基站带来了前所未有的数据传输压力.与此同时,新型移动应用程序呈现出海量、大规模、高数据流量等发展态势,且对数据处理的实时性、安全性等提出了更高的要求[2].这使得传统的以云中心提供服务、核心网传输数据为特点的云计算服务交互模式在服务速度、带宽、能耗以及用户安全和隐私等方面难以应对.在此背景下,边缘计算模式(Edge Computing)[3-4]应运而生.

边缘计算模式在核心网的边缘增加了具有计算、存储、转发等功能的边缘设备,并将这些边缘设备作为服务载体为用户提供服务,从而将服务从云服务中心迁移到距离用户更近的网络边缘.如此一来,用户可以直接访问边缘设备上的服务,而无需经过核心网与云服务中心进行交互.由于边缘计算模式下,服务数据直接在用户和边缘设备之间传输,因此,与传统的云计算模式相比,边缘计算模式极大地缩减了服务数据在核心网上传输所需的时间与能耗开销,提升了用户的服务体验质量;同时,还可以降低核心网的数据传输量,缓解核心网的数据传输压力;此外,由于服务数据直接在边缘设备上处理,避免了远距离传输,故而降低了数据被窃听和篡改的风险,保护了用户的数据安全和隐私.

边缘计算环境中,网络边缘设备称为边缘服务器,如基站、增强型路由器等[5].边缘服务器可以为其覆盖范围内的用户提供服务.然而,由于物理资源的限制,边缘服务器通常只能部署有限数量的服务

实例，处理有限种类的服务请求，不足以覆盖用户复杂多样的服务需求. 此外，用户的移动性、动态性等因素造成了服务请求分布的不均衡性，这使得边缘服务器集群中容易出现部分边缘服务器过载而部分边缘服务器闲置的问题. 当边缘服务器过载时，服务请求的执行会被延迟，从而导致服务响应时间过长，甚至超过用户调用远程云服务器的服务响应时间[6]. 因此，专家提出将邻近的边缘服务器组成一个边缘计算系统(Edge Computing System)，并将边缘计算系统中的边缘服务器作为一个整体共同为用户提供服务[6-7]. 如此一来，边缘服务器收到服务请求以后，可以将服务请求转发给同一个边缘计算系统中最适合的边缘服务器来处理，以此优化系统的负载、提升服务质量. 此时，如何对边缘计算系统中收到的服务请求进行分发以优化整体服务质量并实现系统的负载均衡成为边缘计算系统中亟待解决的关键问题. 本文将研究服务请求分发策略，以有效避免边缘服务器的任务过载或资源浪费，最终达到最优化服务质量，即最小化服务响应时间的目的.

边缘计算系统中的服务请求分发问题可以抽象为决策优化问题. 在集中式的优化模型(Centralized Model)中，通常假定边缘计算系统中存在一个中心控制节点(Controller)，该节点可以实时获取边缘计算系统中所有边缘服务器的实时可用资源信息、服务执行状态、到达服务请求以及边缘服务器之间的网络信息，从而基于这些信息制定服务请求的分发决策. 然而，集中式模型对系统状态信息更新的实时性要求过高，在大规模的动态系统中，很难达到此要求；频繁的信息交互容易产生高昂的网络传输开销；此外，该模型过分依赖中心控制节点，单点故障(Single Point of Failure)瓶颈凸显，即中心控制节点一旦出现故障，会导致整个边缘计算系统的崩溃，因此，模型还存在易出错、难扩展、鲁棒性低等缺点. 因此，本文将通过去中心化模式(Decentralized Model)来解决该问题.

在去中心模式中，由于缺少了中心控制节点，系统无法获取所有边缘服务器的可用资源、服务执行状态、到达服务请求以及边缘服务器之间的网络条件等全局实时信息，因而无法采用传统的集中式优化方法计算获取最优服务请求分配策略，而只能通过各边缘节点之间互相交互、通讯来获取系统中的边缘服务器状态和服务请求信息，进而共同协商制定系统中服务请求的分发策略[8]，而博弈理论正是用于解决多个个体之间通过互相决策和交互而获取最优决策的理论，恰好适用于解决此问题. 因此，本文基于非合作博弈理论(Non-cooperative Game Theory)设计了面向边缘计算环境的服务请求分发方法. 该方法将边缘服务器之间的服务请求分发问题建模为非合作博弈模型. 该模型中，边缘服务器作为博弈参与者，通过多轮次的相互博弈和竞争来为其收到的服务请求做出最优的分发策略，即最小化这些服务请求的平均响应时间. 本文的主要创新点如下：

(1) 本文提出了一种新的服务响应时间计算模型，综合考虑服务的数据传输时延和服务执行时延，并分别基于香农定理(Shannon's Theorem)和排队理论(Queuing Theory)给出了两种时延的计算方法；

(2) 本文将边缘服务器间的服务请求分发问题建模为非合作博弈模型，并从理论上证明了该博弈模型中存在最优解，即 Nash 均衡状态 (Nash Equilibrium)；

(3) 本文设计了DRDG方法(Decentralized Request Dispatching Game)，实现了面向边缘计算环境的去中心模式的服务请求分发，并通过实验证明了方法的有效性和可扩展性.

本文第 2 节介绍边缘计算环境下服务请求分发和负载均衡问题的相关研究工作；第 3 节通过一个具体的实例介绍本文的研究动机；第 4 节介绍边缘计算系统的架构、相关的概念和模型，并给出服务请求分发问题的形式化定义；第 5 节构建非合作博弈模型，并基于该模型提出 DRDG 方法；第 6 节通过实验证明 DRDG 方法的有效性和可扩展性；第 7 节对全文进行总结，并简要介绍下一步工作.

2　相关工作

在边缘计算环境中，对服务响应时间的优化问题一直以来都是业界的研究热点，众多研究者从不同的角度出发，对该问题展开了研究. 其中，一些研究考虑到不同的服务之间存在响应时间的差异，致力于通过服务选择来优化服务的响应时间，这些研究多根据特定的服务请求，通过计算不同候选服务的响应时间，为用户选取最优的服务，以此达到最小化响应时间的目的[6-9]. 例如，针对边缘计算环境中多任务的服务请求，Wu 等人提出了一种启发式方法来为用户作出最优的服务选择方案[6]. 该方法将用户的移动性考虑在内，根据用户的移动轨迹计算每种服务选择方案下用户与边缘服务器之间的数据传输时延以及各个服务器之间的数据传输时延，从而计算服务请求的总体服务响应时间，以此选取最

优的服务选择方案.一些研究者提出通过虚拟机(Virtual Machine,VM)迁移的方式来加快服务的执行效率,实现对服务请求的快速响应[10-12].例如,Rodrigues 等人考虑服务执行和数据传输两方面因素,综合采用 VM 迁移和传输功率控制两种方式共同实现边缘计算环境中的服务响应时间的最小化[10].此外,许多研究者致力于研究边缘服务器上的服务部署策略,旨在通过优化边缘服务器上的服务部署来提高边缘服务器上的服务命中率,从而降低服务的平均响应时间[13-16].例如,Yang 等人设计了边缘计算环境下用户的移动模型和服务访问模型,基于这两个模型对未来一段时间内潜在的服务请求进行预测,根据预测结果,并结合边缘服务器的计算和存储资源限制以及服务部署成本,拟定边缘服务器上的服务部署方案,达到最小化服务响应时间的目的[14].

相比于上述方法,设计有效的服务请求分发策略来优化服务供应的方式更为多见.例如,为了最小化服务请求的总体响应时间,Han 等人将服务的响应时间分为上传时延、下载时延和服务执行时延,根据服务请求对响应时间的敏感程度,对其赋予不同的优先级,提出了一种在线的服务请求分发策略[17].Fan 等人研究物联网服务的响应时间优化问题,设计了一种应用感知的负载分配方法[18].该方法将服务请求分为不同的类型,根据服务的类型对服务请求进行分发和资源配置,并提出对边缘服务器的负载情况进行实时观测,动态地对服务的资源分配进行调整,以此减小服务的响应时间.为了减少车联网服务的响应时间,Liu 等人考虑服务的时间约束和服务之间的依赖关系,将请求分发问题抽象为约束优化问题,并提出了一种有效的请求分发算法[19].Poularakis 等人研究边缘计算架构下多蜂窝网络的服务响应时间优化问题,将该问题转化为服务部署和请求分发的联合优化问题,并考虑服务请求的动态性,提出了一种双目标优化算法来优化边缘计算服务器的服务执行[20].Peng 等人重点考虑了用户的移动性和实时位置,从决策优化的视角提出了一种移动感知和迁移支持的服务请求分配方法,对服务请求按照用户的位置进行实时的在线分配[21].

上述对服务请求分发问题的研究均采用集中式的方法来实现对服务响应时间的优化,因此这些研究都严重依赖中心控制节点.然而,在实际的边缘计算系统中很难找到满足要求的中心控制节点.此外,集中式的架构容易产生高昂的数据传输开销,且存在单点故障问题和易出错、难扩展、鲁棒性低等缺点,实际可应用性不高.因此,本文采用去中心模式来对边缘计算中的服务的响应时间进行优化,以解决集中式架构中存在的这些问题.

目前已有较多针对边缘计算系统去中心化模式的研究,但是这些研究大都集中在服务部署[22]、服务信任[23]等方面,而对去中心化服务请求分发的研究较少.Cui 等人针对物联网边缘计算系统,提出了一种去中心化的服务请求分发策略[24],基于区块链实现了边缘计算系统中服务的去中心化交互和可信执行,但是服务请求的分发仍然依赖中心控制节点完成,因此,并没有完全实现服务请求分发的去中心化.Wu 等人提出了一种基于模糊控制理论的服务请求分发方法[25],实现了用户服务请求分发的完全去中心化,但是由于算法仅面向单个用户,且只能将用户的服务请求分发到其直接连接的边缘服务器,不能做到边缘计算系统的全局优化,因此其优化效果较为有限.Ning 等人针对边缘计算使能的医疗监测物联网系统,提出了一种去中心化的服务请求分发方法[26],由于医疗系统的特殊性,该方法对服务请求分发基于服务的危急程度、数据新鲜度和能耗三个指标,没有考虑对服务请求的响应时间的优化.因此,以上针对边缘计算系统的去中心化服务请求分发方法都不能解决本文提出的服务响应时间感知的去中心化全局服务请求分发问题.

3 研究动机

本节将通过一个具体的实例介绍本文所提方法的动机,并分析中心化和去中心化的服务请求分发方法的不同所带来的实际影响与后果.

假设一家公司开发了一款智能手环,该手环可以实时采集用户的生命体征数据(如体温、心率、运动情况等),并对这些数据进行实时分析以检测用户的身体状态、给出健康建议.

由于手环体积较小,其能力有限,无法在本地对这些数据进行分析.一种解决方案是将用户数据上传到云端对数据进行实时的分析、处理并将结果返回给用户.然而,由于手环会不断地采集用户数据,这种方式会带来极大的网络传输开销.边缘计算模式可以在网络的边缘部署大量的小型边缘服务器,这样可以在用户附近的边缘服务器上对用户数据进行实时处理和响应,以此减小网络传输开销,降低用户数据的传输风险,并缓解核心网的数据传输压力.此外,由于数据传输距离缩短,该模式可以显著减少

数据传输所花费的时间.用户的生命体征数据、身体健康状况、运动情况等都属于用户的隐私数据,用邻近的本地边缘服务器代替远程云服务器处理这些用户数据还可以避免这些数据在核心网上的远距离传输和中转,因此,可以在一定程度上降低这些数据被窃听和篡改的风险,有效避免用户收到错误的健康分析和建议,并保护用户的数据安全和隐私.

用户地理分布的不均衡性会造成边缘服务器负载的不均衡性,例如,在白天工作时间,用户多集中在写字楼等工作区域,而晚上则集中在住宅区域,这会导致白天工作区域的边缘服务器过载而住宅区域的边缘服务器空闲,晚上则相反.因此,需要将这些边缘服务器组成协作空间,对用户的服务请求进行再分发,共同处理用户任务,以此提高边缘服务器资源的利用率,并提升服务的响应速率.

集中式边缘计算模型中,会在系统中指定一个中心控制节点,由中心控制节点计算系统中所有服务请求的再分发策略,因此需要所有边缘服务器不断将自身的实时可用资源信息、服务执行状态、到达服务请求等发送到中心控制节点,从而造成极大的网络传输开销,中心控制节点将承受极大的网络传输压力.此外,中心控制节点一旦出现故障,会导致整个系统的崩溃,因此,该模式容易出错、难扩展、鲁棒性低.而采用去中心模式,边缘服务器无需向中心控制节点不断发送自身的状态信息,解决了中心控制节点的通讯瓶颈问题,边缘服务器间通过互相交互和协商获取最优服务请求再分配策略,从而取消了中心节点的控制,因此,还解决了集中式模型中存在的单点故障问题,使得系统更容易扩展,并提高了系统的鲁棒性.

4　系统模型

本节介绍边缘计算的系统架构模型、服务请求分发模型和服务请求的响应时间计算模型,并对边缘计算环境下的服务请求分发问题做出形式化定义.

4.1　边缘计算系统模型

边缘计算系统由云服务中心、边缘服务器集合和用户三部分构成,如图 1 所示.其中,云服务器由 e_0 表示,边缘服务器集合表示为 $E=\{e_i\}_{i=1}^{N}$.由于边缘计算系统内的边缘服务器服务能力有限,因此需要云服务中心作为边缘服务器的扩展和补充.例如,在第 3 节中的智能手环实例中,一个边缘计算系统中包含手环公司的云服务中心和部署在网络边缘的多个边缘服务器.通常情况下,手环用户的数据由附近的边缘服务器进行分析和处理,而当某一区域用户量过多导致该区域所有边缘服务器过载,使得数据处理等待时间过长,或其它原因导致手环用户请求的服务无法被满足时,可以将这些服务请求发送到云服务中心处理以优化服务的响应时间.为了更好地对边缘计算环境下的服务请求分发问题进行建模分析,本文对边缘计算系统作出如下假设:

(1)云服务中心与所有的边缘服务器互联,且包含用户所需的所有服务类型;

(2)边缘服务器资源有限,因此每个边缘服务器只部署有限种类的服务,不足以覆盖所有用户请求的服务类型,当边缘服务器因没有部署相应的服务类型而无法满足服务请求时,需要将服务请求进行转发,此外,服务请求转发要求目标服务器必须部

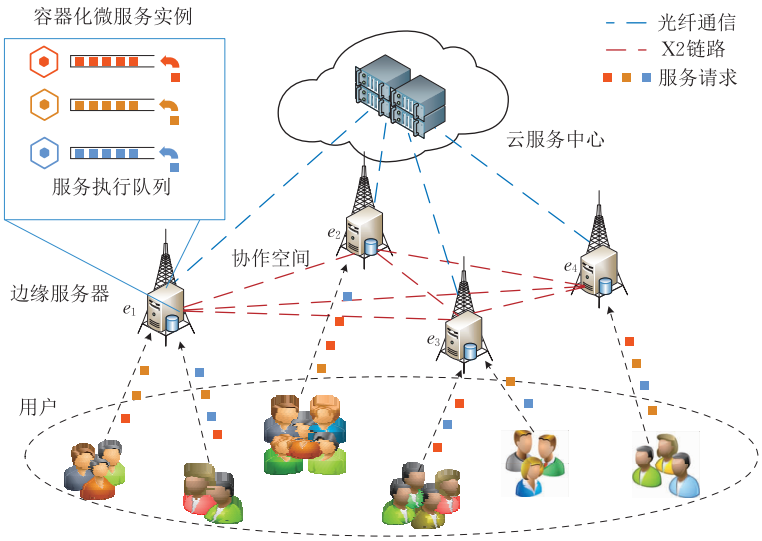

图 1　边缘计算系统模型

署有相应的服务；

(3)边缘服务器之间互相平等、互相之间有路径连通[6-7,15-16]，边缘计算系统内的服务器之间可以共同协作制定服务请求的再分发策略，并根据再分发策略相互调用、相互转发接收到的服务请求来完善资源的协调，服务器之间的数据传输速率由物理距离、基站信号发射功率、信道内的高斯噪声功率等因素共同决定；

(4)边缘服务器依靠基站部署，服务器资源通过 Docker 等容器化技术划分为多个容器，每个容器内部署一个具有特定功能的服务实例，处理一种类型的服务请求(例如，图 1 中边缘服务器 e_1 被划分为 3 个容器，每个容器中部署了一个微服务实例，处理一种服务请求)，由于物理资源的限制，边缘服务器只能部署有限多个功能不同的服务实例，这些实例之间资源隔离，相互独立，从而保证各自的服务请求处理队列中服务处理速率的稳定性；

(5)每个边缘服务器覆盖一个有限范围的 2D 区域，区域内的用户通过无线接入网(Radio Access Network，RAN)等与边缘服务器进行交互，即用户向覆盖他的边缘服务器提交服务请求并从该服务器接收服务的执行结果.

4.2 服务请求分发模型

用户提出的服务请求可以由服务器中部署的相应的微服务实例来实现. 当用户提出的服务请求较为复杂时，可以将服务请求分解为多个子服务请求，使得每个子服务请求可以由一种微服务实例来实现. 各个子服务之间相互独立，无调用关系. 服务器完成服务请求以后将执行结果反馈给用户.

给定边缘服务器 e_i 和服务 s，e_i 覆盖范围内的用户向其发送的关于 s 的服务请求具有随机性，且用户之间相互独立，因此，e_i 收到的关于 s 的服务请求近似服从泊松过程(Poisson Process)[27]，其平均到达速率(单位时间内服务请求的数量)λ 随用户数量、时间等因素动态变化，但在足够短的时间内可认为保持不变[28-29]. 例如，在第 3 节中的智能手环边缘计算系统中，经过预测得到某个住宅小区的边缘服务器在未来的一个小时收到的手环用户的服务请求数量服从泊松分布，其平均到达速率 λ 为 5 个/s. 服务请求的到达速率可以通过自回归模型[30-31]等机器学习方法进行预测，现有的技术方法已经能够达到较高的准确率[29]，因此，本文假设边缘服务器的服务请求到达速率已知.

为了解决边缘服务器服务资源受限和边缘服务器之间的负载均衡问题，边缘服务器会对收到的服务请求进行再分发，以便系统对所有的服务请求进行统筹处理，并优化服务质量. 例如，在白天工作时间，工作区域的边缘服务器过载时可以将部分服务请求发送给住宅区域的空闲边缘服务器处理，晚上则可以将住宅区域的边缘服务器收到的过量服务请求部分转发给工作区域的空闲边缘服务器处理. 因此，每个边缘服务器都会将其收到的服务请求按照一定的分发策略分发到其所在的边缘计算系统中的服务器(包括自身边缘服务器、其它边缘服务器和云服务器)上执行.

给定边缘服务器 e_i 和服务 s，e_i 关于 s 的服务请求分发策略可以表示为向量：

$$\boldsymbol{X}_i^s \triangleq (x_{i0}^s, x_{i1}^s, \cdots, x_{iN}^s) \tag{1}$$

其中，x_{ij}^s 表示 e_i 向服务器 e_j 分发的关于 s 的服务请求的比例，N 表示 e_i 所在边缘计算系统中边缘服务器的数量. 因此，$\forall j \in [0..N]$，$0 \le x_{ij}^s \le 1$ 且 $\sum_{j=0}^{N} x_{ij}^s = 1$. 如果 e_i 自身部署有服务 s，则 e_i 可以将部分服务请求分发给其自身来执行，此时 $x_{ii} \neq 0$；$\forall j \in [0..N]$，如果服务器 e_j 没有部署服务 s，则一定有 $x_{ij}^s = 0$.

图 2 给出了服务请求的处理过程，给定边缘服务器 e_i 和服务 s，服务过程如下：

(1)e_i 收到到达速率为 λ_i^s 的关于 s 的服务请求泊松流；

(2)e_i 与周围的边缘服务器进行通讯、协商，共同制定关于 s 的服务请求再分发策略；

(3)e_i 按照服务请求再分发策略，将该泊松流划分成 $N+1$ 个子泊松流 $\lambda_{i0}^s, \lambda_{i1}^s, \cdots, \lambda_{iN}^s$，并将其分发到系统中相应的云服务器和边缘服务器上进行处理，其中 $\forall j \in [0..N]$，$\lambda_{ij}^s = \lambda_i^s x_{ij}^s$；

(4)服务器收到服务请求以后，将服务请求加入到其服务执行队列中(如图 2 所示)，并按照队列顺序执行这些服务；

(5)服务执行完以后将服务执行结果反馈给 e_i；

(6)e_i 收到服务执行结果以后，将结果转发给对

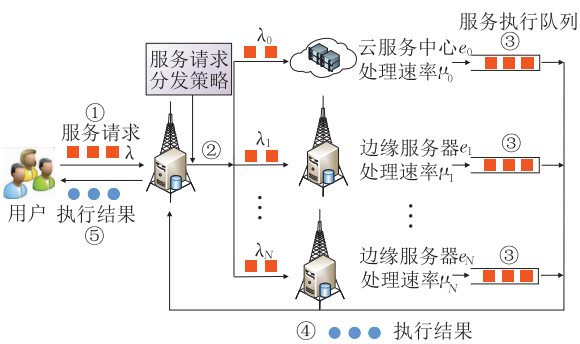

图 2 服务请求处理过程

应的用户.

4.3　服务响应时间计算模型

服务响应时间表示从边缘服务器接收到服务请求到收到服务执行结果之间的时间间隔.比如,边缘服务器 e_i 将收到的关于服务 s 的服务请求分发到服务器 e_j 处理,则该部分请求的响应时间可以通过下式计算:

$$T_{ij}^{s}=T_{i\to j}^{s}+T_{j-}^{s}+T_{i\leftarrow j}^{s} \tag{2}$$

其中,$T_{i\to j}^{s}$ 表示边缘服务器 e_i 将服务请求及其相关数据上传到 e_j 产生的上传时延,包括信道排队时延和数据传输时延;T_{j-}^{s} 表示服务请求在服务器 e_j 上的服务处理时延,包括排队等待时延和服务执行时延;$T_{i\leftarrow j}^{s}$ 表示服务执行结果回传到 e_i 所产生的下载时延,包括信道排队时延和数据传输时延.T_{j-}^{s} 的计算方法将在 4.3.1 节中给出,$T_{i\to j}^{s}$ 和 $T_{i\leftarrow j}^{s}$ 的计算方法将在 4.3.2 节中给出.

4.3.1　服务处理时延

由于边缘服务器所接收到的服务请求的到达过程近似服从泊松过程,因此,本文采用指数函数计算边缘服务器对服务请求任务的处理时间[32-34].

给定边缘服务器 e_j、服务 s,以及边缘计算系统中关于 s 的服务请求到达速率 $\boldsymbol{\lambda}=(\lambda_1^s,\lambda_2^s,\cdots,\lambda_N^s)$,其中 λ_i^s 表示边缘服务器 e_i 收到的关于 s 的服务请求到达速率,根据 4.2 节中的服务请求分发模型,e_i 会向 e_j 以 x_{ij}^s 的比例分发其收到的关于 s 的服务请求,那么 e_j 收到的 e_i 转发的关于 s 的服务请求的到达速率可以表示为 $\lambda_i^s x_{ij}^s$,据此可以计算服务器 e_j 收到的同一边缘计算系统中被转发的关于 s 的服务请求的到达速率:

$$\lambda_j'^{s}=\sum_{i=1}^{N}(\lambda_i^s x_{ij}^s) \tag{3}$$

边缘服务器的资源通过 Kubernetes、Docker 等容器化技术由多个容器共享(如图 1 中的六边形所示).每个容器部署一个特定功能的服务实例并处理相应的服务请求,不同的容器之间互不干扰.因此,边缘服务器上给定服务的执行序列可以描述为一个 $M/M/1$ 排队模型,其中服务等待队列的长度不受限制,服务的执行顺序采用先来先服务(FCFS)的规则.因此,给定 e_j 对服务 s 的请求处理速率 μ_j^s(单位时间内可以处理的服务请求数量),依据 Little 法则[32],可以计算 e_j 上服务 s 的平均处理时间:

$$T_{j-}^{s}=\frac{1}{\mu_j^s-\lambda_j'^{s}} \tag{4}$$

为了保证用户的服务体验质量,被分配的服务请求的到达速率不能超过服务器对该服务的执行能力,即:

$$\lambda_j'^{s}<\mu_j^s \tag{5}$$

根据式(4),当 $\lambda_j'^{s}$ 接近 μ_j^s 时,边缘服务器 e_j 对服务 s 的平均处理时间会不断增长,直至服务 s 的平均响应时间超过将请求转发给其它边缘服务器或者云服务时服务 s 的响应时间,此时,需要将部分服务请求转发给其它边缘服务器或云服务器处理来缩短服务 s 的平均响应时间,否则,服务器 e_j 上等待处理的服务队列将不断增长,导致服务响应时间不断增长,因此式(5)是服务请求分发需要满足的一个约束条件.

4.3.2　服务传输时延

边缘服务器之间通常通过 X2 链路进行数据传输[35],在数据的传输过程中会有排队延时,特别是当边缘计算系统中的服务请求较多或服务数据量较大时,容易导致链路拥塞,产生较高的数据传输排队延时.现有的对服务响应时间的计算模型中通常不考虑边缘服务器之间的数据传输产生的排队延时或者对其进行简化处理[7-9].然而,这部分排队延时对服务请求的分发策略具有较大的影响,忽略这部分排队延时会导致服务请求的分发策略失准,进而降低用户的服务体验质量.因此,本文对服务的数据传输时间进行计算时,将数据传输产生的排队延时考虑在内,以此提高服务请求的分发策略的有效性,从而进一步降低服务请求的平均响应时间.

本文采用香农定理(Shannon's Theorem)[36]计算边缘计算系统中服务器之间的数据传输速率:

$$R_{i,j}=B\log_2\left(1+\frac{P_i H_{ij}}{w_i}\right),\quad \forall i\in[0..N],\forall j\in[1..N] \tag{6}$$

其中,$R_{i,j}$ 表示服务器 e_i 和 e_j 之间的数据传输速率,B 表示边缘计算系统内的信道带宽,P_i 表示边缘服务器 e_i 的发射功率,w_i 表示信道内的高斯噪声功率,H_{ij} 表示信道增益,可以通过下式计算:

$$H_{ij}=l_{ij}^{-\alpha} \tag{7}$$

其中,$l_{i,j}$ 表示边缘服务器 e_i 和 e_j 之间的物理距离,α 表示路径衰减因子.

结合式(4)和(6),我们可以计算服务请求的上传和下载时延:

$$T_{i\to j}^{s}=\frac{1}{R_{ij}/Z_u^s-\lambda_{ij}^s} \tag{8}$$

$$T_{i\leftarrow j}^{s}=\frac{1}{R_{ji}/Z_d^s-\lambda_{ij}^s} \tag{9}$$

其中,Z_u^s 和 Z_d^s 分别表示服务 s 的输入和输出参数的数据量;式(8)中 λ_{ij}^s 表示边缘服务器 e_i 对服务器 e_j 关于服务 s 的请求分发速率(单位时间内转发服务请求的数量);式(9)中 λ_{ij}^s 表示服务 s 执行结果从边

缘服务器 e_j 到 e_i 的回传速率. 边缘服务器 e_j 在接收到 e_i 发送的服务请求以后要执行所有这些服务请求,然后把所有服务请求的执行结果返回 e_i. e_i 按照恒定的速率 λ_{ij}^s 向 e_j 转发关于 s 的服务请求,由于 e_j 处理关于 s 的服务请求的速率大于服务请求到达的速率,所以 e_j 可以按时完成这些服务请求并返回服务的执行结果. 因此,e_j 向 e_i 回传关于 s 的服务执行结果的速率同样是 λ_{ij}^s,即 e_j 到 e_i 回传服务执行结果的速率等于 e_i 向 e_j 发送服务请求的速率. 例如,如果边缘服务器 e_i 每秒向 e_j 转发 5 个关于服务 s 的服务请求,那么 e_j 每秒都可以完成 5 个来自 e_i 的关于 s 的服务请求,并向 e_i 返回 5 个关于 s 的服务请求的执行结果.

4.4 问题定义

给定边缘计算系统中包含的云服务器 e_0 和边缘服务器集合 $E=\{e_i\}_{i=1}^N$,以及该边缘计算系统中关于服务 s 的服务请求到达速率 $\lambda=(\lambda_1^s,\lambda_2^s,\cdots,\lambda_N^s)$,则关于服务 s 的请求分发问题可以定义为:制定服务请求分发策略

$$\boldsymbol{X}^s \triangleq (\boldsymbol{X}_1^s,\boldsymbol{X}_2^s,\cdots\boldsymbol{X}_N^s) \tag{10}$$

使得系统中收到的所有关于 s 的服务请求的平均响应时间最短,即

$$\min_{\boldsymbol{x}^s}\frac{\sum_{i=1}^{N}(\lambda_i^s T_i^s)}{\sum_{i=1}^{N}\lambda_i^s} \tag{11}$$

其中,$\boldsymbol{X}_i^s$ 表示边缘服务器 e_i 关于服务 s 的请求分发策略(参见式(1)定义),T_i^s 表示边缘服务器 e_i 收到的关于 s 的服务请求的平均响应时间,通过下式进行计算:

$$T_i^s=\sum_{j=0}^{N}(x_{ij}^s T_{ij}^s) \tag{12}$$

其中,T_{ij}^s 表示边缘服务器 e_i 分发到服务器 e_j 的关于 s 的服务请求的响应时间,计算方式已由式(2)给出.

以上定义了关于服务 s 的请求分发问题. 由于边缘计算系统中的服务器通过容器化技术处理不同类型的服务请求,所以,不同类型的服务请求之间互不影响、互不干扰. 因此,当边缘计算系统中存在多种类型的服务请求时,边缘计算系统对这些不同类型的服务请求分别处理、独立制定服务请求分发策略.

5 去中心化服务请求分发方法

本节介绍去中心化模式的服务请求分发方法. 首先,基于非合作博弈理论将边缘计算环境下的服务请求分发问题抽象为非合作博弈模型;然后,基于该模型设计去中心化的服务请求分发算法,实现边缘服务器间的服务请求分发.

5.1 非合作博弈模型

根据式(1),边缘服务器 e_i 关于服务 s 的请求分发策略集合可以表示为

$$\mathbb{X}_i^s=\left\{\boldsymbol{X}_i^s \,\middle|\, \sum_{j=0}^{N}x_{ij}^s=1,0\le x_{ij}^s\le 1\right\} \tag{13}$$

令 $\boldsymbol{\mu}_i^s=(\mu_{i0}^s,\mu_{i1}^s,\cdots,\mu_{ij}^s,\cdots,\mu_{iN}^s)$ 表示当前时刻边缘计算系统中服务器 e_i 可用的服务处理能力,其中,μ_{ij}^s 表示服务器 e_j 可向 e_i 提供的关于服务 s 的请求处理速率. 令 $\boldsymbol{X}_{-i}^s=\{\boldsymbol{X}_1^s,\cdots,\boldsymbol{X}_{i-1}^s,\boldsymbol{X}_{i+1}^s,\cdots,\boldsymbol{X}_N^s\}$ 表示边缘计算系统中除 e_i 外其它边缘服务器的服务请求分发策略集合,根据式(12),边缘服务器 e_i 所接收服务请求的平均响应时间可以表示为

$$T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)=\sum_{j=0}^{N}x_{ij}^s\left(T_{i\to j}^s+\frac{1}{\mu_{ij}^s-x_{ij}^s\lambda_i^s}+T_{i\leftarrow j}^s\right) \tag{14}$$

其中,λ_i^s 表示边缘服务器 e_i 接收到的关于 s 的服务请求的到达速率. $T_{i\to j}^s$ 和 $T_{i\leftarrow j}^s$ 可以通过式(8)和(9)计算得到.

定义 1. 给定边缘服务器 e_i 和边缘计算系统中除 e_i 外其它边缘服务器的服务请求分发策略集合 $\boldsymbol{X}_{-i}^s$,称 $\hat{\boldsymbol{X}}_i^s\in\mathbb{X}_i^s$ 是边缘服务器 e_i 的最优服务请求分发策略,当且仅当:

$$\forall \boldsymbol{X}_i^s\in\mathbb{X}_i^s, T_i^s(\hat{\boldsymbol{X}}_i^s,\boldsymbol{X}_{-i}^s)\le T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s) \tag{15}$$

定义 1 给出了最优服务请求分发策略的定义. 通过定义 1,可以将边缘服务器间的服务请求分发问题建模为非合作博弈模型 $G=(E,\{\mathbb{X}_i^s\}_{i\in E},\{T_i^s\}_{i\in E})$,其中 E 表示博弈参与者即边缘服务器集合;$\mathbb{X}_i^s$ 表示边缘服务器 e_i 关于 s 的请求分发策略集合;$T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 表示博弈参与者 e_i 的效益函数,即边缘服务器 e_i 接收的服务请求的平均响应时间. 在边缘计算系统中,最优服务请求分发策略能够使得系统中所有服务请求的响应时间最低,该策略对应非合作博弈模型中的 Nash 均衡状态. 下面给出 Nash 均衡状态的定义.

定义 2. 给定非合作博弈 G,称 G 在 $\hat{\boldsymbol{X}}^s=\{\hat{\boldsymbol{X}}_1^s,\hat{\boldsymbol{X}}_2^s,\cdots,\hat{\boldsymbol{X}}_N^s\}$ 达到 Nash 均衡状态,当且仅当:

$$\forall e_i\in E,\forall \boldsymbol{X}_i^s\in\mathbb{X}_i^s, T_i^s(\hat{\boldsymbol{X}}_i^s,\hat{\boldsymbol{X}}_{-i}^s)\le T_i^s(\boldsymbol{X}_i^s,\hat{\boldsymbol{X}}_{-i}^s) \tag{16}$$

其中,$\hat{\boldsymbol{X}}_{-i}^s=\{\hat{\boldsymbol{X}}_1^s,\hat{\boldsymbol{X}}_2^s,\cdots,\hat{\boldsymbol{X}}_{i-1}^s,\hat{\boldsymbol{X}}_{i+1}^s,\cdots,\hat{\boldsymbol{X}}_N^s\}$.

定义 2 给出了 Nash 均衡状态的定义. 通过定义 2 可以得出,在 Nash 均衡状态,没有任何一个边缘服务器可以通过单方面改变自己的分发策略来进一步降低其收到的服务请求的平均响应时间. Nash

均衡状态意味着所有边缘服务器都可以在该状态获得满意的服务请求分发策略. 因此，Nash 均衡状态是边缘计算系统中服务请求分发的稳定性保证. 在求解 Nash 均衡状态之前，我们首先要确定服务请求分发问题的博弈模型中存在 Nash 均衡状态.

定理 1.　非合作博弈 G 存在 Nash 均衡状态.

证明.　根据文献[37]，非合作博弈 G 存在 Nash 均衡状态需要满足两个充分条件：(1) 边缘服务器 e_i 的策略集 $\mathbb{X}_i^s$ 是欧几里得空间上的一个非空有界闭凸子集；(2) 在已知边缘计算系统中其它边缘服务器分发策略集合 $\boldsymbol{X}_{-i}^s$ 的条件下，边缘服务器 e_i 的效用函数 $T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 连续可微，且对任意策略 $\boldsymbol{X}_i^s\in\mathbb{X}_i^s$ 是凸的. 显然，边缘服务器 e_i 的策略集 $\mathbb{X}_i^s$ 是一个凸集和紧集，且其效用函数 $T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 对 $\boldsymbol{X}_i^s$ 连续可微. 因此，只需证明效用函数 $T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 是 $\boldsymbol{X}_i^s$ 的凸函数即可，这可以通过证明 $T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 的 Hessian 矩阵是半正定矩阵来证得. 下面我们对此进行证明. 根据式(14)，可以得到效用函数 $T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 对变量 $\boldsymbol{X}_i^s$ 的一阶导数：

$$\begin{aligned}\nabla_{\boldsymbol{X}_i^s}T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)&=\left[\frac{\partial T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)}{\partial x_{ij}^s}\right]_{j=0}^N\\&=\left[\frac{R_{ij}/Z_u}{(R_{ij}/Z_u-x_{ij}^s\lambda_i^s)^2}+\frac{\mu_{ij}^s}{(\mu_{ij}^s-x_{ij}^s\lambda_i^s)^2}+\right.\\&\left.\frac{R_{ji}/Z_d}{(R_{ji}/Z_d-x_{ij}^s\lambda_i^s)^2}\right]_{j=0}^N\end{aligned}\tag{17}$$

因此，$T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 对变量 $\boldsymbol{X}_i^s$ 的 Hessian 矩阵可以表示为

$$\begin{aligned}\nabla^2_{\boldsymbol{X}_i^s}T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)&=diag\left\{\left[\frac{\partial^2 T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)}{\partial x_{ij}^s}\right]_{j=0}^N\right\}\\&=diag\left\{\left[\frac{2\lambda_i^s(R_{ij}/Z_u+\mu_{ij}^s+R_{ji}/Z_d)}{(R_{ij}/Z_u-x_{ij}^s\lambda_i^s)^3+(\mu_{ij}^s-x_{ij}^s\lambda_i^s)^3+}\right.\right.\\&\left.\left.(R_{ji}/Z_d-x_{ij}^s\lambda_i^s)^3\right]_{j=0}^N\right\}\end{aligned}\tag{18}$$

根据式(5)、(8)、(9)可知，不等式 $\mu_{ij}^s>x_{ij}^s\lambda_i^s$，$R_{ij}/Z_u>x_{ij}^s\lambda_i^s$ 和 $R_{ji}/Z_d>x_{ij}^s\lambda_i^s$ 为服务请求分配的约束条件，必然成立. 因此，式(18)中对角矩阵所有对角元素均为正. 综上，边缘服务器 e_i 的效用函数 $T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)$ 的 Hessian 矩阵是正定的.　　证毕.

定理 1 证明了边缘计算环境下的服务请求分发问题对应的非合作博弈模型存在 *Nash* 均衡状态，即边缘计算环境下的服务请求分发问题存在最优解. 因此，可以通过非合作博弈模型的求解得到边缘计算系统的最优服务请求分配策略.

5.2　博弈求解

根据式(15)和(16)可以得到，为了最小化自身收到的服务请求的响应时间，每个边缘服务器都应当根据其它服务器的服务请求分发策略以及可用的服务处理能力，将其接收到的服务请求分发到最适合的边缘服务器上进行处理，从而实现服务平均响应时间的最小化. 因此，对于边缘服务器 e_i，该问题可以定义为

$$\text{Min } T_i^s(\boldsymbol{X}_i^s,\boldsymbol{X}_{-i}^s)\tag{19}$$

$$\text{s. t.}\sum_{j=0}^N x_{ij}^s=1\tag{20}$$

$$0\le x_{ij}^s\le 1\tag{21}$$

$$\sum_{i=1}^N x_{ij}^s\lambda_i^s<\mu_j^s\tag{22}$$

由于在边缘计算系统中，用户对服务的响应时间通常比较敏感，因此需要对服务的响应时间进行约束，为此，本文定义 $T^{\max}$ 表示服务的最大可容忍时延，即服务需要在最大可容忍时延之内完成：

$$T_{i\to j}^s+\frac{1}{\mu_{ij}^s-\lambda_{ij}^{s,\max}}+T_{i\leftarrow j}^s\le T^{\max}\tag{23}$$

其中，

$$T_{i\to j}^s=\frac{1}{R_{ij}/S_u-\lambda_{ij}^{\max}}\tag{24}$$

$$T_{i\leftarrow j}^s=\frac{1}{R_{ji}/S_d-\lambda_{ij}^{\max}}\tag{25}$$

上式中 $\lambda_{ij}^{s,\max}$ 表示在满足 $\lambda_{ij}^{s,\max}<\mu_{ij}^s$ 的条件下，边缘服务器 e_i 可以向服务器 e_j 发送服务请求的最大速率. 此外，还需要满足 $T_{i\to j}^s+T_{i\leftarrow j}^s<T^{\max}$，即服务请求的输入参数和执行结果的传输时延总和不超过服务请求的最大可容忍时延.

本文采用 Nash 竞价解(Nash Bargaining Solution)[33]计算边缘服务器的最优服务请求分发策略. 首先将边缘服务器之间的初始服务处理速率设置为 $\mu_{ij}^s=\lambda_{ij}^{s,\max}$. 为简化模型，对式(14)取对数操作，则原问题等价转化为

$$\text{Min}-\sum_{j=0}^N\ln(\mu_{ij}^s-x_{ij}^s\lambda_i^s),\forall e_i\in E\tag{26}$$

$$\text{s. t. }(20),(21),(22)$$

将边缘服务器的当前可用服务处理速率按降序排序，即 $\mu_{i1}^s\ge\mu_{i2}^s\ge\cdots\ge\mu_{iN}^s$，于是，得到该问题的最优解：

$$x_{ij}^s=\begin{cases}\dfrac{\mu_{ij}^s}{\lambda_i^s}-\dfrac{\sum_{j=0}^k\mu_{ij}^s-\lambda_i^s}{k\lambda_i^s}, & j=1,2,\cdots,k\\ 0, & j=k+1,\cdots,N\end{cases}\tag{27}$$

其中，k 为满足以下不等式的最大值：

$$\mu_{ik}^s>\frac{\sum_{j=0}^k\mu_{ij}^s-\lambda_i^s}{k},k=1,2,\cdots,N\tag{28}$$

5.3 算法设计

本节介绍去中心模式的边缘计算环境下服务请求的分发过程.该过程由两个算法实现,其中算法 1 描述边缘计算系统中各边缘服务器之间为获取最优服务请求分配策略而进行的互相交互、互相协商的过程.此过程是一个不断迭代、逐步趋近最优的过程.在每次迭代中,每个边缘服务器都会根据其它边缘服务器的决策为自己所收到的服务请求作出最优分发策略.算法 2 详细描述了边缘服务器作出该最优分发策略的步骤.因此,算法 1 执行过程中会不断调用算法 2,即算法 2 是算法 1 的子算法.

算法 1 的目的是获得边缘计算环境中的服务请求分发对应的非合作博弈模型的 Nash 均衡状态.为此,算法首先将每个边缘服务器的平均响应时间和其服务请求分发比例向量初始化为 0(第 2 行);然后,边缘计算系统中边缘服务器采用轮转的方式分别运行算法 2 来计算各自的最优服务请求分发策略(第 5～12 行);为了检测系统的 Nash 均衡状态,在算法中引入累计迭代误差变量 sum,每轮迭代开始时,sum 值初始化为 0(第 4 行),每个边缘服务器重新做完决策以后都会将决策前后响应时间的差值加入到累计迭代误差 sum 中(第 10 行),当一轮迭代完成以后,如果 sum 不为 0,表示此次迭代过程中仍有边缘服务器改变了自己的决策,即系统仍未达到 Nash 均衡状态,此时,算法进入下一轮迭代,直到系统中所有边缘服务器前后两次迭代的平均响应时间不再变化,即 $sum=0$ 时,表示边缘计算系统中的服务请求分发达到了 Nash 均衡状态,算法终止(第 13 行).

算法 1. 去中心化的服务请求分发算法 DRDG (Decentralized Request Dispatching Algorithm).

输入:边缘计算系统中的边缘服务器集合 E,
服务器之间的物理距离 $L=\{l_{ij}\mid e_i,e_j\in E\}$,
边缘服务器所在基站的传输功率 $\boldsymbol{P}=\{P_i\}_{i=1}^{N}$,
边缘计算系统内的带宽 B,
路径衰减因子 α,
服务请求参数大小 Z_u,
服务执行结果的返回大小 Z_d,
服务最大的可容忍时延 $\mathcal{T}^{\max}$

输出:服务请求分发策略集合 $\boldsymbol{X}^s=(\boldsymbol{X}_1^s,\boldsymbol{X}_2^s,\cdots,\boldsymbol{X}_N^s)$

1. 计算服务器之间的传输速率 $\mathcal{R}=\{R_{ij}\mid e_i,e_j\in E\}$
2. $\forall e_i\in E$, $\boldsymbol{X}_i^s\leftarrow 0$, $T_i^s\leftarrow 0$,
3. REPEAT
4. 累计迭代误差 $sum\leftarrow 0$
5. FOR each $e_i\in E$ do
6. $\forall e_j\in E,\mu_{ij}^s=\mu_j^s-\sum_{k=0,k\neq i}^{N}x_{kj}^s\lambda_k^s$
7. $\boldsymbol{\mu}_i^s=(\mu_{i0}^s,\mu_{i1}^s,\cdots,\mu_{ij}^s,\cdots,\mu_{iN}^s)$
8. $\boldsymbol{X}_i^s\leftarrow$ORDA($\lambda_i^s,\boldsymbol{\mu}_i^s,\mathcal{R},Z_u,Z_d,\mathcal{T}^{\max}$)
9. $T_i^s{}'\leftarrow$边缘服务器 e_i 的加权平均响应时间
10. $sum\leftarrow sum+|T_i^s{}'-T_i^s|$
11. $T_i^s\leftarrow T_i^s{}'$
12. END FOR
13. UNTIL $sum=0$
14. RETURN $\boldsymbol{X}^s=(\boldsymbol{X}_1^s,\boldsymbol{X}_2^s,\cdots,\boldsymbol{X}_N^s)$

根据 5.2 节的博弈求解模型,本文设计了边缘服务器的最优服务请求分发策略,如算法 2 所示.算法首先计算边缘服务器基于 Nash 竞价解的初始参与处理速率(第 2 行),并将它们按降序排序(第 3 行);然后,依据式(28)搜索最大值 k(第 4～8 行),为了简化算法描述,引入变量 v 计算式(28)的后半部分(第 4 行),判断当前边缘服务器队列中的最后一个边缘服务器是否满足最优解的条件(第 5 行),如果不满足条件,则将当前最后一个边缘服务器从服务器队列中删除(第 6 行),并计算下一个边缘服务器的 v 值(第 7 行),继而判断该边缘服务器是否满足最优条件,如此循环往复,直到找到满足最优解条件的边缘服务器(第 8 行);然后计算对其它边缘服务器的最优服务请求分发比例(第 9～11 行),根据式(27),经过前面的搜索和删除,目前留在边缘服务器队列中的边缘服务器均为需要分发服务请求的边缘服务器,因此对这些边缘服务器(第 9 行)按照式(27)进行赋值(第 10 行),其余边缘服务器为初始参与处理速率较低的边缘服务器,这些边缘服务器保留其初始值 0,即不向它们分发服务请求;最后将策略向量 $\boldsymbol{X}_i^s$ 返回作为算法执行结果(第 12 行).

算法 2. 最优服务请求分发算法 ORDA(Optimal Request Dispatching Algorithm).

输入:边缘服务器 e_i 关于服务 s 的请求到达速率 λ_i^s,
边缘服务器 e_i 当前可用的服务处理能力 $\boldsymbol{\mu}_i^s$,
服务器间的数据传输速率 $\mathcal{R}=\{R_{ij}\mid e_i,e_j\in E\}$,
服务请求输入参数数据量 Z_u,
服务执行结果数据量 Z_d,
服务的最大可容忍时延 $\mathcal{T}^{\max}$

输出:边缘服务器 e_i 关于服务 s 的最优请求分发策略 $\boldsymbol{X}_i^s$

1. 初始化边缘服务器 e_i 的请求分发向量 $\boldsymbol{X}_i^s\leftarrow 0$
2. 根据式(23)计算边缘服务器初始服务处理速率 $\boldsymbol{\mu}_i^s=(\mu_{i0}^s,\mu_{i1}^s,\cdots,\mu_{ij}^s,\cdots,\mu_{iN}^s)$
3. 将 $\boldsymbol{\mu}_i^s$ 中的初始服务处理速率按照降序排序
4. $v\leftarrow\left(\sum_{j=0}^{|\boldsymbol{\mu}_i^s|}\mu_{ij}^s-\lambda_i\right)/|\boldsymbol{\mu}_i^s|$
5. WHILE $v>\mu_{i|\boldsymbol{\mu}_i^s|}^s$
6. $\boldsymbol{\mu}_i^s\leftarrow\boldsymbol{\mu}_i^s\backslash\mu_{i|\boldsymbol{\mu}_i^s|}^s$

7. $\quad v \leftarrow \left(\sum_{j=0}^{|\boldsymbol{\mu}_i^s|} \mu_{ij}^s - \lambda_i^s\right) / |\boldsymbol{\mu}_i^s|$

8. END WHILE

9. FOR $j \leftarrow 0$ to $|\boldsymbol{\mu}_i^s|$

10. $\quad x_{ij}^s \leftarrow (\mu_{ij}^s - v)/\lambda_i^s$

11. END FOR

12. RETURN $\boldsymbol{X}_i^s = (x_{i0}^s, x_{i1}^s, \cdots, x_{iN}^s)$

算法 2 仅包含一次查找和一次赋值过程，因此其复杂度为 $O(n)$，其中 n 表示系统中边缘服务器的数量. 算法 1 包含一个迭代过程，每次迭代中每个边缘服务器都需要做出决策，结合算法 2 的复杂度可以得出算法 1 的复杂度为 $O(mn^2)$，其中 m 表示算法 1 迭代的次数. 因此，整个决策过程的时间复杂度是多项式级的，算法的执行时间仅与迭代次数和系统中的边缘服务器数量相关，与服务请求的数量等其它参数无直接关系. 随着系统规模的增长，算法的执行时间呈多项式级增长，具有良好的可扩展性和收敛性.

6　实验分析

为了验证方法的可行性和有效性，本文利用 Python 语言实现了 ORDA 和 DRDG 算法①. 实验环境为 Intel Core i5-4210H 处理器（双核 CPU，2.9 GHz）、8 GB RAM、Windows 操作系统.

6.1　实验设置

数据集. 实验数据采用 EUA 数据集[38]，该数据集中包含澳大利亚 CBD 地区所有基站的位置信息，即边缘服务器的位置信息. 图 3 给出了其中部分边缘服务器的部署位置.

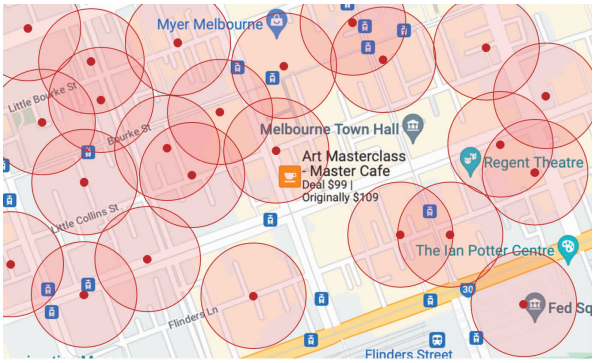

图 3　实验环境

参数设置. 边缘计算系统的规模设置为 10 个边缘服务器，为此我们从 EUA 数据集中随机选取 100 组不同的边缘服务器，每组包含 10 个不同地理位置的边缘服务器作为一个边缘计算系统. 实验过程中，根据给定的参数配置，对每组边缘服务器重复相同的实验操作，最后取平均值作为实验结果.

边缘服务器的最大服务处理速率从集合{10，20，40，50，100}（tasks/s）中随机获取[39]. 在实际的边缘计算环境中，系统负载率描述了系统对用户提出的服务请求的满足程度，是刻画系统运行状态的重要指标. 因此，本文后续的实验中将采用系统负载率作为变量验证本文方法的有效性. 服务请求的到达率 λ 可以通过系统的总处理速率与系统负载率的乘积计算得到. 服务请求在给定边缘服务器 e_i 上的到达率 λ_i 通过比例因子 f_i 与服务总到达率 λ 的乘积来计算，其中，每个边缘服务器的比例因子从区间[0，1]内随机生成，且满足 $\sum_{i=1}^{N} f_i = 1$.

对于服务请求，考虑到不同用户发出的服务请求的异构性，本文将实验中服务请求的输入参数的数据量 S_u 从区间[200，400] KB 中随机选取[40]，而服务执行结果数据量 S_d 从区间[50，150] KB 中随机选取. 服务的最大可容忍时延 $\mathcal{T}^{\max}$ 从区间[200，280] ms 中随机选取.

对于边缘服务器间的通信网络，本文将边缘计算系统内的信道带宽 B 设置为 20 MHz，边缘服务器所有基站的发射功率 P_i 设置为 20 Watts[41]. 最后，参考文献[42]中的物理干扰模型，边缘服务器间的路径衰减因子 α 设置为 4，信道内的高斯噪声功率设置为 −100 dBm.

6.2　对比方法

在实验中，本文设计了以下四种不同的对比方法来和 DRDG 算法进行对比分析：

（1）Optimal[18,21]. 该方法是目前解决边缘计算环境中服务请求分发问题的常用方法，其采用集中式的边缘计算模式，假定边缘计算系统中存在中心控制节点，将服务请求分发问题视为一个静态的全局优化问题，并通过求解以下非线性优化问题来获取服务请求的最优分发策略：

$$\text{Min} \quad \frac{1}{\sum_{i=0}^{N} \lambda_i^s} \sum_{i=0}^{N} \lambda_i^s T_i^s(\boldsymbol{X}_i^s, \boldsymbol{X}_{-i}^s) \tag{29}$$

其约束条件为式(20)、(21)和(22)；

（2）PRDA[33]. 该方法是按比例请求分发方法（Proportional Request Dispatching Approach），该方法中每个边缘服务器按照当前系统可用的服务

① https://github.com/qgmzhna/DRDG

处理速率的比例分发其收到的服务请求，即 $x_{ij}^{s}=\mu_{ij}^{s}\Big/\sum_{j=1}^{N}\mu_{ij}^{s}$；

(3) Greedy. 边缘服务器按照当前可用服务处理速率从高到低的顺序依次分发其收到的服务请求，直至分发完毕；

(4) Random. 边缘服务器随机分发其收到的服务请求，直至分发完毕.

6.3 有效性评估

首先，我们评估系统负载对服务平均响应时间的影响，实验参数按照 6.1 节设置. 考虑到服务请求的输入和输出数据量、请求的到达速率等实验参数具有随机性，我们将所有实验重复执行 400 次，取平均值作为实验结果.

实验结果如图 4 所示. 结果显示，通过 5 种方法得出的服务平均响应时间都随着系统负载的增加而提高. 这是因为，随着系统负载的增加，系统会执行更多的服务请求，使得边缘服务器间的数据传输时延和边缘服务器上的服务处理时延增加，因此，服务的平均响应时间会随之提高. 此外，Optimal 方法可以得到全局最优解，因此其平均响应时间最低，然而 Optimal 方法需要已知边缘计算系统中的所有服务器的实时服务执行状态、到达服务请求以及边缘服务器之间的网络信息作为前提条件，因此，该方法实际应用范围有限，只适用于集中式的优化模型中，不能应用于本文所研究的去中心的边缘计算模式中. 本文所提的 DRDG 方法与 Optimal 全局最优方法得到的实验结果非常接近，且明显优于其它对比方法，该实验结果证明了 DRDG 方法的有效性. 此外，当系统负载较高时，DRDG 方法的平均响应时间接近于 PRDA 方法. 根据式(29)，当系统负载较高的时候，DRDG 方法的请求分发策略接近于 PRDA，因此，这两种方法的平均响应时间之间的差距会随着系统负载的增加而逐渐减小.

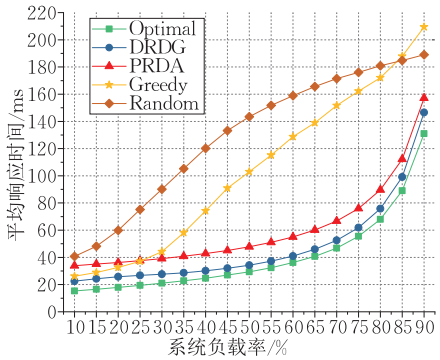

图 4 系统负载率对服务平均响应时间的影响

为了考察边缘服务器的服务处理速率对服务的平均响应时间的影响，我们变化服务器的平均处理速率并记录服务平均响应时间的变化，实验结果如图 5 所示. 结果表明，随着服务器处理速率的增长，服务平均响应时间逐渐降低. 根据式(4)，服务器处理速率的提高会使得服务的平均处理时间降低，因此服务的平均响应时间会相应降低. 五种方法之间的对比与图 4 保持一致，DRDG 方法与 Optimal 方法的平均差距在 5 ms 以内，而与另外三种方法的差距都在 10 ms 以上，该结果同样证明了 DRDG 方法的有效性.

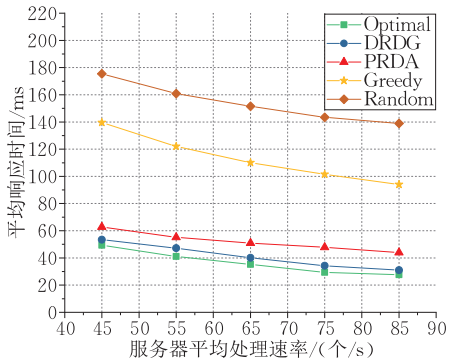

图 5 服务处理速率对服务平均响应时间的影响

此外，我们考察了边缘计算系统的规模对服务请求的平均响应时间的影响，为此，我们对系统中边缘服务器的数量进行改变，同时记录服务平均响应时间的变化，实验结果如图 6 所示. 结果表明系统中边缘服务器数量越多，服务的平均响应时间越短. 由于实验中我们保持服务请求的数量不变，边缘服务器增多时，每台服务器被分配的服务请求数量会减少，根据式(4)可知服务的平均处理时间会降低，因此服务的平均响应时间降低. 五种方法之间的对比与图 4 和图 5 一致，即 DRDG 方法的结果与 Optimal 较为接近，并远远优于其它三种方法.

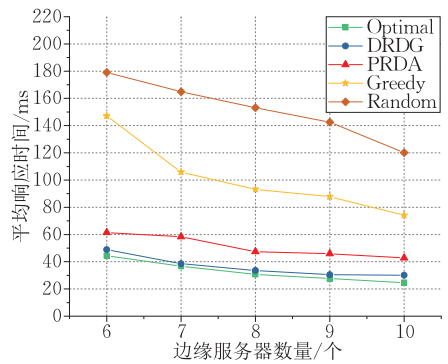

图 6 服务器数量对服务平均响应时间的影响

同时，为了对比五种方法的执行效率，我们记录了这些方法的执行时间，结果如表 1 所示. 结果显

示，随着边缘计算系统规模的增长，几种方法的执行时间都会相应增长. Optimal 方法的执行时间远高于其它四种方法，因此，该方法虽然其能够获得最优解，但是由于其执行时间过长且增长过快，并不适用于大规模边缘计算系统. 此外，由于 PRDA、Greedy 和 Random 三种方法较为简单，DRDG 方法的执行时间略高于这三种方法，但仍在可用的范围内.

表 1　算法执行效率评估

服务器数量	Optimal	DRDG	PRDA	Greedy	Random
5	179.8	3.5	3.3	1.8	1.8
8	681.3	15.5	5.9	2.8	2.7
11	978.8	50.0	9.4	4.4	4.2
14	1462.9	88.3	16.9	6.5	5.7
17	2246.8	146.7	22.1	11.0	8.4

为了对比五种方法对边缘服务器之间负载均衡的优化效果，我们在上述实验中记录了十台服务器的平均服务响应时间，结果如图 7 所示. 通过实验结果看出，Greedy 和 Random 方法得到的分配策略中，不同边缘服务器之间的服务响应时间之间存在很大的差异，这意味着系统中存在负载分配不均的情况，即部分边缘服务器过载而部分边缘服务器较为空闲. DRDG 方法在负载均衡方面的优势非常明显，DRDG 方法得到的分发策略中所有边缘服务器的平均响应时间几乎相等，该实验结果证明 DRDG 方法可以有效优化边缘计算系统中边缘服务器之间的负载均衡，避免负载分配不均的问题. 此外，PRDA 方法由于按比例对服务请求进行分配，所以同样可以达到较好的负载均衡效果，而 Optimal 方法的负载均衡效果稍弱，其得到的分发策略中边缘服务器之间的平均响应时间存在明显差异. 同时，我们记录了五种方法下每个边缘服务器的资源利用率的情况，结果如图 8 所示. 实验结果与图 7 一致，即本文提出的 DRDG 方法下各服务器的资源利用率较为均衡，PRDA 方法由于按比例对服务请求进行分配，因此该方法下各服务器之间的负载最为均衡，

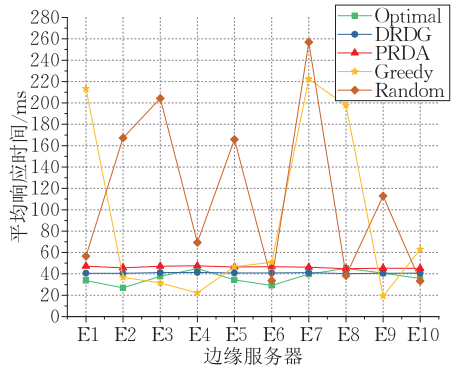

图 7　每个边缘服务器的平均响应时间

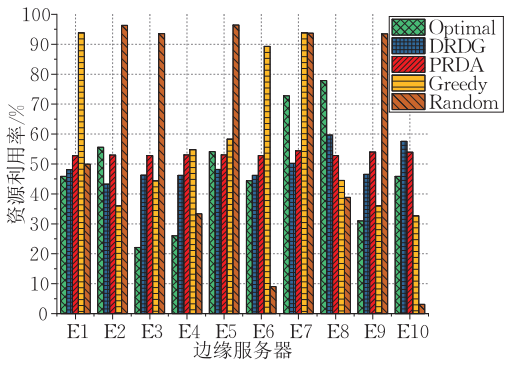

图 8　每个边缘服务器的资源利用率

而 Optimal、Greedy 和 Random 三种方法下各服务器的资源利用率差别较大. 该结果也证明了本文提出的 DRDG 方法具有良好的负载均衡效果.

综上，Optimal 方法虽然可以得到最小的平均服务响应时间，但是由于其时间复杂度远高于本文提出的方法，且存在过度依赖于中心控制节点等缺点，因此，该方法实际应用范围有限，不适合用于去中心化的大规模边缘计算环境中；本文所提出的 DRDG 方法得到的服务请求分配策略在优化服务请求的平均响应时间上接近于全局最优，且相较于其它三种对比方法优势明显，可以有效降低服务请求的平均响应时间，又能获得良好的负载均衡效果.

6.4　可扩展性评估

为了评估 DRDG 方法的可扩展性，实验记录了不同系统负载率下算法收敛的平均迭代次数. 实验结果如图 9 所示. 结果显示，随着系统负载的提高，系统收敛的平均迭代次数逐渐缓慢增加，并在 75% 负载率之后，逐渐趋于稳定；当系统负载超过 80% 以后，平均收敛迭代次数开始下降. 根据 6.3 节的结果分析，当系统负载较高时，DRDG 方法的优化空间会变小且优化策略接近于 PRDA 方法，所以算法收敛的迭代次数会相应减少. 因此，该实验结果表明 DRDG 方法可以随着系统负载的增加表现出具有良好的可扩展性.

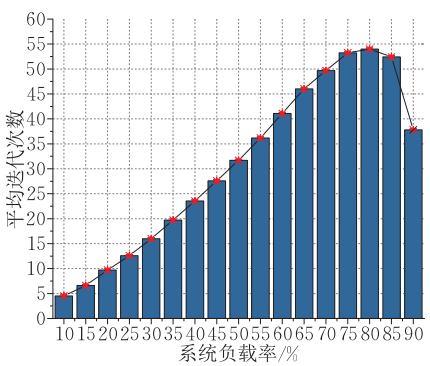

图 9　系统负载率对收敛迭代次数的影响

为了评估服务请求数量的变化对 DRDG 方法执行时间的影响，我们记录了不同请求到达速率下算法的平均执行时间，实验结果如图 10 所示. 结果显示，随着服务请求到达速率的增长，算法的执行时间呈先增长后下降的趋势. 本实验中，服务请求数量增加会使得系统中边缘服务器的负载率增加. 根据图 9 中实验结果可知，当系统负载率到达阈值之前，算法的迭代次数会随系统负载率的增加而增加，因此，算法的执行时间相应增长，而当系统负载率到达阈值以后，算法的迭代次数开始下降，从而算法的执行时间也随之不断降低.

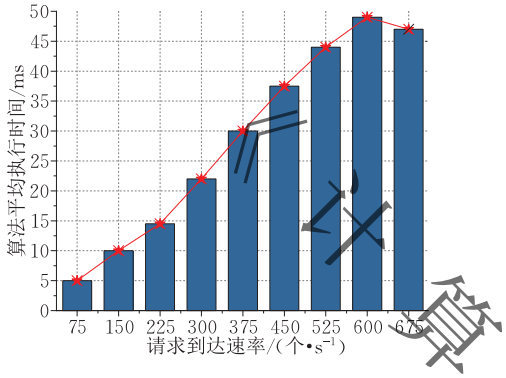

图 10 请求到达速率对算法执行时间的影响

最后，对边缘计算系统的规模对算法执行时间的影响进行了验证，实验结果如图 11 所示. 结果显示，算法的执行时间随着系统中边缘服务器数量的增长而增长. 这是因为算法的迭代过程中需要每个边缘服务器作出决策，且每个边缘服务器作决策时需要对系统中的所有服务器进行计算和比较，因此算法的执行时间会随系统规模的增长而增长. 该实验结果与 5.3 节中对方法复杂度的分析一致，验证了 DRDG 方法随着系统规模的增长具有良好的可扩展性.

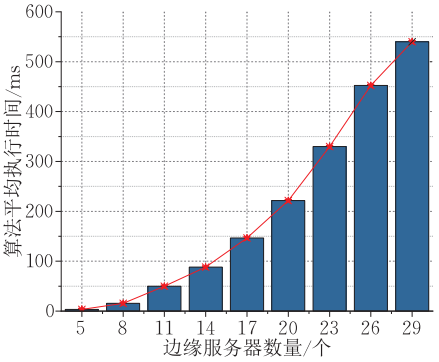

图 11 系统规模对算法执行时间的影响

7 总结与展望

本文研究边缘计算环境下多服务器之间的服务请求分发问题. 为了解决该问题，我们将其抽象描述为非合作博弈模型，并利用 Nash 竞价解设计了一种去中心化的服务请求分发方法，实现了边缘计算系统中的服务请求分发和边缘服务器之间的负载均衡. 实验结果表明，本文所提出的方法可以有效缩减服务请求的响应时间，并且随着系统规模的增加表现出良好的可扩展性.

在未来的工作中，我们将完善本文所提出的服务请求分发方法，从而使其可以更好地应用于现实世界中. 例如，用户的移动性会对服务请求的分发效果产生较大的影响，因此可以将用户的移动性考虑在内，对本文所提出的方法进行优化改进；此外，本文仅考虑服务请求的响应时间对服务请求进行分发，而未考虑服务的其它质量属性，因此未来将考虑将服务的能耗、可靠性等其它质量属性考虑在内，对服务请求进行分发，进一步提高服务质量.

参 考 文 献

[1] Ericsson mobility report. https://www.ericsson.com/4a03c2/assets/local/mobility-report/documents/2021/june-2021-ericsson-mobility-report.pdf, 2021

[2] Wu H, Deng S, Li W, et al. Revenue-driven service provisioning for resource sharing in mobile cloud computing//Proceedings of the 15th International Conference on Service-Oriented Computing (ICSOC). Malaga, Spain, 2017: 625-640

[3] Hu Y, Patel M, Sabella D, et al. Mobile edge computing—A key technology towards 5G. ETSI White Paper, 2015, 11(11): 1-16

[4] Shi Wei-Song, Zhang Xing-Zhou, Wang Yi-Fan, Zhang Qing-Yang. Edge computing: State-of-the-art and future directions. Journal of Computer Research and Development, 2019, 56(1): 69-89(in Chinese)
(施巍松，张星洲，王一帆，张庆阳. 边缘计算：现状与展望. 计算机研究与发展，2019，56(1)：69-89)

[5] Zhou Yue-Zhi, Zhang Di. Near-end cloud computing: Opportunities and challenges in the post-cloud computing era. Chinese Journal of Computers, 2019, 42(4): 677-700 (in Chinese)
(周悦芝，张迪. 近端云计算：后云计算时代的机遇与挑战. 计算机学报，2019，42(4)：677-700)

[6] Wu H, Deng S, Li W, et al. Mobility-aware service selection in mobile edge computing systems//Proceedings of the 26th International Conference on Web Services (ICWS). Milan, Italy, 2019: 201-208

[7] Wu H, Deng S, Li W, et al. Service selection for composition in mobile edge computing systems//Proceedings of the 15th International Conference on Web Services (ICWS). San Francisco, USA, 2018: 355-358

[8] Jiang Y. A survey of task allocation and load balancing in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 2015, 27(2): 585-599

[9] Huang J, Lan Y, Xu M. A simulation-based approach of QoS-aware service selection in mobile edge computing. Wireless Communications and Mobile Computing, 2018, 18(1): 1-10

[10] Rodrigues T, Suto K, Nishiyama H, Kato N. Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control. IEEE Transactions on Computers, 2016, 66(5): 810-819

[11] Rodrigues T, Suto K, Nishiyama H, et al. Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration. IEEE Transactions on Computers, 2018, 67(9): 1287-1300

[12] Yang L, Yang D, Cao J, et al. QoS guaranteed resource allocation for live virtual machine migration in edge clouds. IEEE Access, 2020, 8: 78441-78451

[13] Zhang Kai-Yuan, Gui Xiao-Lin, Ren-De-Wang, et al. Survey on computation offloading and content caching in mobile edge networks. Journal of Software, 2019, 30(8): 2491-2516(in Chinese)
(张开元，桂小林，任德旺等. 移动边缘网络中计算迁移与内容缓存研究综述. 软件学报，2019, 30(8): 2491-2516)

[14] Yang L, Cao J, Liang G, Han X. Cost aware service placement and load dispatching in mobile cloud systems. IEEE Transactions on Computers, 2016, 65(5): 1440-1452

[15] Gao B, Zhou Z, Liu F, Xu F. Winning at the starting line: joint network selection and service placement for mobile edge computing//Proceedings of the 38th International Conference on Computer Communications(INFOCOM). Paris, France, 2019: 1459-1467

[16] Ma X, Zhou A, Zhang S, Wang S. Cooperative service caching and workload scheduling in mobile edge computing//Proceedings of the 39th International Conference on Computer Communications(INFOCOM). Toronto, Canada, 2020: 2076-2085

[17] Han Z, Tan H, Li X, et al. OnDisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds. IEEE/ACM Transactions on Networking, 2019, 27(6): 2472-2485

[18] Fan Q, Ansari N. Application aware workload allocation for edge computing-based IoT. IEEE Internet of Things Journal, 2018, 5(3): 2146-2153

[19] Liu Y, Wang S, Zhao Q, et al. Dependency-aware task scheduling in vehicular edge computing. IEEE Internet of Things Journal, 2020, 7(6): 4961-4971

[20] Poularakis K, Llorca J, Tulino A, et al. Joint service placement and request routing in multi-cell mobile edge computing networks//Proceedings of the 38th International Conference on Computer Communications(INFOCOM). Paris, France, 2019: 10-18

[21] Peng Q, Xia Y, Zeng F, et al. Mobility-aware and migration-enabled online edge user allocation in mobile edge computing//Proceedings of the 26th International Conference on Web Services(ICWS). Milan, Italy, 2019: 91-98

[22] Aral A, Tolga O. A decentralized replica placement algorithm for edge computing. IEEE Transactions on Network and Service Management, 2018, 15(2): 516-529

[23] Liu Y, Yu F, Li X, et al. Decentralized resource allocation for video transcoding and delivery in blockchain-based system with mobile edge computing. IEEE Transactions on Vehicular Technology, 2019, 68(11): 11169-11185

[24] Cui L, Yang S, Chen Z, et al. A decentralized and trusted edge computing platform for Internet of Things. IEEE Internet of Things Journal, 2019, 7(5): 3910-3922

[25] Wu C, Peng Q, Xia Y, et al. Online user allocation in mobile edge computing environments: A decentralized reactive approach. Journal of Systems Architecture, 2021, 113: 101904

[26] Ning Z, Dong P, Wang X, et al. Mobile edge computing enabled 5G health monitoring for Internet of Medical Things: A decentralized game theoretic approach. IEEE Journal on Selected Areas in Communications, 2020, 39(2): 463-478

[27] Kingman J. Poisson Processes. USA: John Wiley & Sons, Ltd, 2005

[28] Mao Y, Zhang J, Letaief K. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605

[29] Xu J, Chen L, Zhou P. Joint service caching and task offloading for mobile edge computing in dense networks//Proceedings of the 37th International Conference on Computer Communications (INFOCOM). Honolulu, USA, 2018: 207-215

[30] Zhang Q, Zhani M, Boutaba R, Hellerstein J. Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Transactions on Parallel and Distributed Systems, 2014, 2(1): 14-28

[31] Hines M, Gopalan K. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning//Proceedings of the 2009 International Conference on Virtual Execution Environments (SIGPLAN/SIGOPS). Washington, USA, 2009: 51-60

[32] Yang B, Li Z, Chen S, et al. Stackelberg game approach for energy-aware resource allocation in data centers. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3646-3658

[33] Grosu D, Chronopoulos A, Leung M. Load balancing in distributed systems: An approach using cooperative games//Proceedings of the 16th International Parallel and Distributed Processing Symposium. Fort Lauderdale, USA, 2002: 1-10

[34] Yang B, Li Z, Jiang S. Cooperative game approach for energy-aware load balancing in clouds//Proceedings of the 2007 International Symposium on Parallel and Distributed Processing with Applications and 2017 International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). Guangzhou, China, 2017: 9-16

[35] Ndikumana A, Tran N, Ho T, et al. Joint communication, computation, caching, and control in big data multi-access edge computing. IEEE Transactions on Mobile Computing, 2020, 19(6): 1359-1374

[36] Chen X. Decentralized computation offloading game for mobile

cloud computing. IEEE Transactions on Parallel and Distributed Systems, 2014, 26(4): 974-983
[37] Fadlullah Z, Quan D, Kato N, Stojmenovic I. GTES: An optimized game-theoretic demand-side management scheme for smart grid. IEEE Systems Journal, 2014, 8(2): 588-597
[38] Zou G, Liu Y, Qin Z, et al. ST-EUA: Spatio-temporal edge user allocation with task decomposition. IEEE Transactions on Services Computing, 2022, Early Access
[39] Kishor A, Niyogi R, Veeravalli B. A game-theoretic approach for cost-aware load balancing in distributed systems. Future Generation Computer Systems, 2020, 109: 29-44
[40] He X, Zheng J, He Q, et al. Online computation offloading for deadline-aware tasks in edge computing. Wireless Networks, 2022, 26: 1-20
[41] Xiao A, Wang X, Wu S, et al. Mobility-aware resource management for integrated satellite-maritime mobile networks. IEEE Network, 2021, 36(1): 121-127
[42] Feng H, Guo S, Zhu A, Wang Q, Liu D. Energy-efficient user selection and resource allocation in mobile edge computing. Ad Hoc Networks, 2020, 107: 102202

WU Hong-Yue, Ph. D., associate professor. His current research interests include service computing, edge computing and mobile cloud computing.

CHEN Zhi-Wei, M. S. candidate. His main research interests include service computing and edge computing.

SHI Bo-Wen, M. S. candidate. His main research interests include service computing and edge computing.

DENG Shui-Guang, Ph. D., professor. His main research interests include service computing, edge computing, process management and big data.

CHEN Shi-Zhan, Ph. D., associate professor. His main research interests include service computing and service ecosystem mining and analysis.

XUE Xiao, Ph. D., professor. His main research interests include service computing, swarm intelligence and computational experiment.

FENG Zhi-Yong, Ph. D., professor. His main research interests include knowledge engineering, service computing and computer cognition.

Background

Service computing is an advanced research direction of software engineering and distributed computing. It takes software services as the research object and supports the construction, operation and management of large-scale Internet service systems. Services are defined as software artifacts that are autonomous, self-described, reusable and highly portable. By using services as the basic units to build rapid, low-cost, secure and reliable applications, service computing saves the development costs that would otherwise be spent on creating new software components.

The emergence of edge computing paradigm has further promoted the development and application of service computing. By catching cloud services on edge servers, it migrates services from remote cloud servers to close proximity to users. This can greatly reduce the time and energy consumption on service data transmission, improve the quality of services and relieve the traffic load on the core network. However, due to the limitation of physical resources of edge servers, one key problem of edge computing is dispatching service requests to appropriate edge servers with the objective of optimizing the load balancing between edge servers and improving the quality of services.

Existing methods usually adopt centralized methods to solve this problem by assuming that there is a control center in each edge computing system that can acquire the real-time information of all edge servers and makes decisions on global request dispatching based on this information. However, it is hard to find a control center in real edge computing systems and updating the state of an entire edge computing system in real-time is costly and hard to achieve. Moreover, these methods suffer from the serious problem of single point of failure. Besides, due to the centralized architecture, these methods are error-prone, difficult to expand and of low robustness.

In this paper, we propose a novel game-theoretic method to solve the problem in a decentralized manner. It allows edge servers to communicate and negotiate with each other to determine the dispatching of their received service requests collaboratively. Benefiting from its decentralized structure, this method can avoid the real-time information update on edge servers, alleviate the single point of failure, and achieve high flexibility, scalability and reliability.

This work is supported by the National Natural Science Key Foundation of China under Grant No. 61832014 and No. 62032016, the National Natural Science Foundation of China under Grant Nos. 61972276, 62102281, U20A20173 and 62125206, the National Key Research and Development Program of China grant No. 2021YFF0900800, the Shandong Key Laboratory of Intelligent Buildings Technology under Grant No. SDIBT202001, and the Key R&D Program of Zhejiang Province under Grant No. 2022C01145.