doi:10.3969/j.issn.1001-4616.2022.02.016

多边缘服务器协作环境下基于时延感知的服务选择算法

谢　娜[1],谭文安[1,2],孙　勇[3],赵　璐[1],黄　黎[1,4]

(1.南京航空航天大学计算机科学与技术学院,江苏 南京 211106)
(2.上海第二工业大学计算机与信息工程学院,上海 201209)
(3.南京师范大学地理科学学院,江苏 南京 210023)
(4.江苏开放大学信息与机电工程学院,江苏 南京 210017)

[摘要]　移动边缘计算可为用户提供低时延的服务. 然而,随着用户的需求变得日益复杂多样,单个边缘服务器难以满足其需求. 因此,多边缘服务器协作环境下的服务选择问题成为服务计算领域的热点难题. 本文首先将该问题建模成带约束的最优化问题,然后提出了一种启发式的服务选择算法-LLMES 算法. 该算法是在边缘服务器网络中根据迪杰斯特拉算法求解当前本地服务器的邻居节点作为候选服务器,并基于低时延多有效服务的贪心选择策略选择为用户提供有效服务最多且时延最小的服务器作为当前最优服务器. 从而选择出一组相互协作的边缘服务器集合共同为用户提供服务,即选中一组满足用户需求的服务. 最后,实验结果表明本文提出的 LLMES 算法性能明显优于其他 3 种具有代表性的算法.

[关键词]　移动边缘计算,启发式算法,时延感知,服务选择
[中图分类号]TP391　[文献标志码]A　[文章编号]1001-4616(2022)02-0126-10

Service Selection Algorithm Based on Latency-Aware in a Multiple Edge Server Cooperation Environment

Xie Na[1], Tan Wenan[1,2], Sun Yong[3], Zhao Lu[1], Huang Li[1,4]

(1.School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)
(2.School of Computer and Information, Shanghai Polytechnic University, Shanghai 201209, China)
(3.School of Geographical Sciences, Nanjing Normal University, Nanjing 210023, China)
(4.School of Information and Electromechanical Engineering, Jiangsu Open University, Nanjing 210017, China)

Abstract: Mobile edge computing can provide low latency services for users. However, as the requirements of users become increasingly complex and diverse, it is difficult for a single edge server to meet their needs. Therefore, the service selection problem in a multiple edge server cooperation environment has become a hot issue in the field of service computing. In this paper, the problem is modeled as a constrained optimization problem, and then a heuristic service selection algorithm named the LLMES algorithm is proposed. The algorithm selects the neighbor nodes of the current local server as candidate servers according to the Dijkstra algorithm in the edge server network, chooses an edge server that provides the most effective services with the least latency for users as the current optimal server based on the greedy selection strategy of low latency and multiple effective services. Thus, a group of cooperative edge servers is selected to provide services for users, that is, a group of services that meet the requirement of users is selected. Finally, experimental results show that the performance of the LLMES algorithm proposed in this paper outperforms significantly three representative approaches.

Key words: mobile edge computing, heuristic algorithm, latency-aware, service selection

近年来,随着移动设备和无线网络技术的快速发展,服务不再局限于传统的平台而变得更加复杂多样. 用户可以不受时空约束而任意选择所需的服务来处理自己业务. 这也进一步加剧了服务数量和种类的快速增长. 同时无人驾驶、智慧城市[1]、增强现实[2]等延迟敏感型和计算密集型新应用服务的出现和流

收稿日期:2021-08-04.
基金项目:国家自然科学基金项目(61672022、U1904186).
通讯作者:谭文安,博士,教授,博士生导师,研究方向:服务计算、边缘计算、群智协同计算等. E-mail:wtan@foxmail.com

行,使得用户对服务质量和聚合粒度提出更高要求[3]. 如何为用户选择合适的服务是服务选择的热点问题,并引起了学者们的密切关注.

Deng 等[4]在移动云环境中分析了服务持续移动性和基于位置变化的特点,提出了移动模型和移动感知的 QoS 计算规则,采用基于教-学优化可移动性的选择算法获得近似最优解,实验验证该方法优于当前标准的优化算法. 文献[5]分析了在移动环境中服务请求者和提供者均具有移动性,提出了一个移动服务共享社区的移动服务供应体系结构,采用 Krill-Herd 算法解决服务组合的问题. 以上在移动云环境中的服务选择工作是针对服务的移动性展开的,而忽略服务能量的消耗问题. 文献[6]针对移动云环境下移动设备电池容量有限的问题,以移动状态下某用户调用工作流的总能耗最小为目标进行服务选择,提出了不同结构组合服务的能耗聚合规则并采用遗传算法求解能耗最低的高质量服务组合. Tong 等[7]提出了一种能量感知的保证 QoS 的工作流管理机制,设计了一个兼顾服务能量和服务质量的高效服务选择方案,并提出了基于平衡能量消耗的自适应机制的约束分解的服务选择方法. 文献[8]是在满足延迟约束的基础上以最小化移动设备的能量消耗为目标,选择每个子任务在移动设备端或云端执行的策略. 将协同任务执行定义为延迟约束的工作流调度问题,根据不同的情况使用两种算法调度任务. 对于无执行限制的特殊情况采用单攀策略求解. 对于部分任务必须在移动设备或云上执行的一般情况则采用 LARAC 算法求解. 通过仿真实验表明协同任务执行比本地执行和远程执行更节能.

以上服务选择的研究均是以服务资源在云端为前提的. 然而,随着移动网络服务资源日益增多,有限的网络带宽容易给用户传输造成网络阻塞,使得访问云端服务的时延过高,不能满足用户低延迟服务的需求,如无人机,虚拟/增强现实[2]等. 云端的服务选择方式已不适应于时延敏感型的服务. 移动边缘计算(mobile edge computing,简称 MEC)作为新兴的一种计算模式[9-10]应运而生,它可以将服务资源下沉到靠近用户端,并通过无线网将移动设备与边缘服务器相连,为用户提供低时延的服务[11]. 部分研究者把服务选择的环境转向边缘端. 如文献[12]针对网络质量引起的边缘计算协同服务效率和质量低等问题,构建了基于盟主的边缘计算协同服务器组织模型,并提出了协同服务池的算法解决边缘计算服务节点过载问题. 实验验证该算法能有效提高边缘计算协同服务能力和质量. 文献[13]研究了在移动边缘计算系统中的服务选择问题,以进一步减少服务请求的总体响应时间为目的,考虑了交付服务的边缘服务器选择. 提出了一种结合遗传算法和模拟退火算法的新启发式算法. 通过实验仿真证明所提方法的高效性.

在上述边缘环境下的服务选择研究中,虽然能满足用户低延时的服务需求,但由于边缘服务器端的计算能力、存储容量和通信资源相对有限[14],故在其上部署的服务数量有限. 而用户需求复杂多变,需要多个服务相互协作共同完成其复杂业务[15],单个边缘服务器已不能提供用户所需的所有服务. 如何在移动边缘环境下为用户提供复杂低延迟的服务是当前需要解决的问题. 针对该问题,本文从多个边缘服务器相互协作共同为用户提供服务的思想出发,将边缘协作的服务选择(Edge Cooperation Service Selection,ECSS)问题建模成一个带约束的最优化问题,在边缘服务器网络中根据 Dijkstra 算法求解当前用户的本地服务器到其他边缘服务器的最短距离,基于服务器的路由阈值选择本地服务器的邻居节点作为候选服务器集合;并提出了一种基于低时延多有效服务的贪心选择策略(Low-latency Much-effective service,简称为 LLMES)的新启发式算法,用于从候选服务器集合中选择一组延迟最小且相互协作共同为用户提供服务的边缘服务器,即选中了一组满足用户需求的近似最优服务. 通过实验仿真验证了本文提出的 LLMES 算法性能明显优于其他 3 种具有代表性的方法.

1　问题定义及模型

假设在某个区域内放置 n 个边缘服务器 $ES=\{es_1,es_2,\cdots,es_n\}$,每个边缘服务器 es_i 上已部署若干服务,记作 $SE_i=\{\cdots s_j\cdots\}(1\leqslant j\leqslant m)$. 其中覆盖用户的边缘服务器集合为 $ESC(ESC\subset ES)$,边缘服务器之间可通过高速链路与邻近边缘服务器相互连接并共享信息资源[16-17],从而构成一个边缘服务器网络,用二元组 $G=<ES,E>$表示. 其中 $E=\{\cdots<es_i,es_j>\cdots\}(es_i,es_j\in ES)$表示服务器之间通过高速链路直接连接边的集合. 假设用户需要一组服务 $R=\{s_1,s_2,\cdots,s_m\}$,为了简化模型,进一步假设 R 中的服务之间没有逻辑关系,可以并行处理,且这些服务分布在不同服务器上. 如果想要满足用户服务需求则要从覆盖用户的某个服务器出发,寻找其邻近一组边缘服务器组成一个服务器集合相互协作共同为用户提供所有服务. 且保

证用户访问服务的总时延最小. 所以,选择服务的过程就转换成选择边缘服务器的过程. 如何选择一组边缘服务器能够提供用户所需的服务并保证服务的总时延最小是本文要解决的问题.

如图 1 所示,用户需要的服务是$\{s_1,s_2,s_3,s_4,s_5,s_6\}$,而边缘服务器中有一个本地服务器节点 es_1 覆盖用户,用户从 es_1 出发选择 es_2 和 es_6 组成的服务器集合$\{es_1,es_2,es_6\}$相互协作共同为用户提供一组服务,且这组服务器提供的服务时延最小. 在现实 ECSS 问题场景中,服务器数量和服务数量较大,因此在 ECSS 中发现最优解有一定的难度. 所以边缘环境服务提供商急需一种有效的方法来解决 ECSS 问题.

1.1　边缘服务器访问模型

定义 1　本地服务器　当一个边缘服务器 es_i 能够被用户 u 直接访问时,我们称该边缘服务器 es_i 是用户 u 的本地服务器,或者说 u 被 es_i 覆盖. 将本地服务器组成的集合记为 $ESC(ESC\subset ES)$,为了描述方便,我们采用一维向量 C 来描述服务器集合 ES 中包含的本地服务器,即:$C=\{c_1,c_2,\cdots,c_n\}$,$c_i\in\{0,1\}$. 其中 c_i 是二进制变量,表示 es_i 是否为本地服务器,若 $c_i=1$ 表示 es_i 是本地服务器,用户 u 可直接访问;反之则不是本地服务器,用户 u 不能直接访问.

在边缘服务器网络中,服务器之间可以直接相互通信,也可通过其他服务器进行间接路由通信. 如图 2 所示,在边缘服务器相互协作的拓扑结构中,es_1 与 es_2 直接通信,而 es_1 与 es_4 则需要通过 es_2 或 es_6 进行路由间接通信. 本文为了构建一个简洁且通用的模型,用路由跳数 hp_{ij} 来衡量边缘服务器 es_i 到 es_j 之间的路由距离[16-17]. 当用户必须通过本地边缘服务器 es_i 的高速链路才能间接访问边缘服务器 es_j 时,我们称用户间接访问 es_j,且满足 $hp_{ij}\leqslant hp_{\text{limit}}$. 其中 hp_{limit} 是服务器之间能够路由的最大跳数,也称路由阈值,图 2 中本地服务器为 es_1.

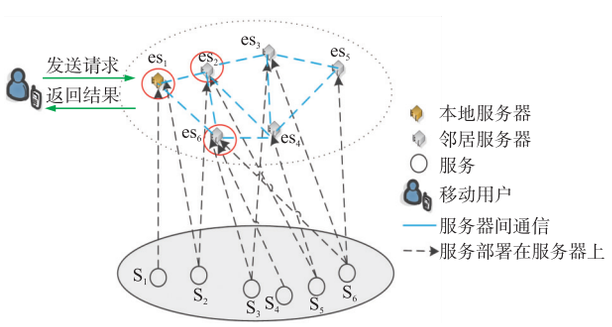

图 1　一个边缘服务器协作的场景

Fig. 1　A scene of edge server collaboration

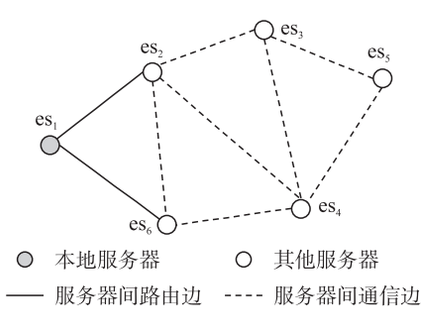

图 2　边缘服务器协作的拓扑结构

Fig. 2　A topology of edge server collaboration

定义 2　邻居服务器　给定任意一个边缘服务器 es_i,存在其他服务器 es_j 和 es_i 在路由阈值范围内($0\leqslant hp_{ij}\leqslant hp_{\text{limit}}$)进行数据信息资源共享和传输,则称 es_j 是 es_i 的邻居服务器. 把 es_i 的所有邻居服务器节点组合起来的集合称为 es_i 的邻居服务器集合,记作 NE_i,且 $NE_i\subset ES$.

当用户可以访问(直接访问或者间接访问)边缘服务器 es_i 时,才能充分利用其资源. 我们采用了一维向量 A 表示用户对边缘服务器集合 ES 的访问情况,即 $A=\{a_1,a_2,\cdots,a_n\}$,$\forall a_i\in\{0,1\}$,$i\in[1,n]$. 其中 a_i 是一个二进制变量,表示用户 u 是否可以访问 es_i. 若 $a_i=1$ 表示用户 u 可访问 es_i;反之,则不可访问. a_i 具体的定义如下所示:

$$a_i=\begin{cases}1, & c_i=1,\quad hp_i=0,i\in[1,n],\\ 1, & c_k=1,\quad \forall es_k\in NE_i,hp_{ik}\leqslant hp_{\text{limit}},k\in[1,n],\\ 0, & \text{otherwise } hp_{ik}>hp_{\text{limit}},\end{cases}\tag{1}$$

式中,NE_i 表示边缘服务器 es_i 的邻居节点集合,hp_{ik} 表示边缘服务器 es_i 和 es_k 之间的路由距离. 当用户 u 被 es_i 覆盖时,即 $c_i=1$,用户 u 可直接访问本地服务器 es_i 上的应用服务资源,即 $a_i=1$;当用户没有被 es_i 覆盖而被 es_i 的邻居节点 $es_k(es_k\in NE_i)$ 覆盖时,即 $c_k=1$,用户可通过 es_k 到 es_i 之间的通信路径($es_k\cdots es_i$)间接访问 es_i,即 $a_i=1$;当 es_i 没有覆盖用户且其邻居节点均无覆盖用户 u,则用户 u 不可访问 es_i,即 $a_i=0$. 如图 2 中,假设服务器的路由阈值 hp_{limit} 是 1,则边缘服务器 es_5 不能被访问,而其他服务器可以被访问,对应的访问向量为 $A=\{1,1,1,1,0,1\}$.

1.2　服务时延感知的计算模型

管理移动设备用户的上行/下行通信的无线基站一般是3G/4G宏蜂窝或小蜂窝基站[18]. 当多个用户同时访问边缘服务器时,传输的数据速率公式可以参考文献[18]和[19],这里我们不做详细描述. 用户调用服务的时延和很多因素有关,如上行和下行传输的数据量,上行和下行的传输速率,服务的运行时间. 参考文献[19]和[20],当一个用户向边缘服务器发送服务请求 s_k 时,服务请求的生命周期主要分为4个部分,首先通过移动网络将服务 s_k 请求由用户上传至最近的边缘服务器 es_i(这里为本地服务器)上;如果 es_i 不能满足用户所有服务需求时,则会向邻近的服务器发送服务请求,路由选择合适边缘服务器 es_j 来处理请求并将请求数据传输到该服务器上;接着服务器 es_j 接到请求后运行服务 s_k 并返还运行结果;最后再把服务的执行结果传送给本地边缘服务器 es_i 并由 es_i 返还给用户. 由于服务上行的数据量远大于下行数据量的传输,所以上行数据传输时延远大于下行数据时延,这里我们只考虑上行数据传输时延. 为了简化模型,我们进一步假设服务执行时延远小于数据传输时延. 所以用户访问服务的时延只和用户访问本地服务器的传输速率,服务器之间的路由传输速率,以及服务上行数据量的大小有关.

解决ECSS问题的关键就是选择一组部署在不同边缘服务器上的服务,且保证用户访问所有服务 R 的总时延最低. 首先从当前本地服务器集合中选择一个本地边缘服务器 es_i,然后选择 es_i 中能够提供的服务,接着从 es_i 出发路由周边的邻居服务器节点,根据低时延多有效服务的贪心选择策略选择合适的服务器节点,直到选中一组服务器中包含的服务能够满足用户的所有需求. 所以总时延的计算分为两部分,一部分是用户将服务需求数据传输到被选的本地服务器 es_i 上的访问时延,另一部分是将服务从本地服务器 es_i 传输到其他被选中服务器之间的路由时延. 接下来我们给出这两部分计算时延的方法.

(1)用户访问本地边缘服务器的时延

假设用户选择的本地服务器是 es_i,所需要的服务集合是 $R=\{s_1,s_2,\cdots,s_m\}$,则用户访问本地服务器 es_i 的时延为:

$$T_{C2E}(i)=\begin{cases}\sum_{k=1}^{m}\dfrac{U_k}{\alpha_i}, & c_i=1,\\ +\infty, & \text{otherwise},\end{cases}\tag{2}$$

式中,U_k 表示服务 s_k 需要输入的数据量大小,α_i 是用户到本地服务器 es_i 的上行传输速率,可通过文献中的香农公式获得[18,20],这里不做详细描述. $T_{C2E}(i)$ 表示用户的服务需求 R 上传到本地服务器 es_i 的访问时延,c_i 用来控制边缘服务器 es_i 为本地服务器,由定义1可知,$c_i=1$ 表示 es_i 为本地服务器,否则为非本地服务器.

(2)本地服务器到其他服务器之间的路由时延

假设用户选择的本地服务器为 es_i,es_i 的邻居服务器节点集合为 NE_i,则本地服务器 es_i 到邻居节点服务器 $es_j(es_j\in NE_i)$ 的路由时延为:

$$T_{E2E}(i,j)=\begin{cases}\dfrac{\sum_{s_l\in SE_j}U_l}{\beta_{i,j}}, & a_j=1,\\ +\infty, & \text{otherwise},\end{cases}\tag{3}$$

式中,U_l 表示服务 s_l 需要传输的数据量大小,SE_j 表示服务器 es_j 能够提供的服务集合. 由文献[18]和[21]可知服务器之间的传输速率和路由的带宽及拓扑结构相关. 而服务器之间的路由距离可通过Dijkstra算法求解两点之间的最小路径来获取. 为了简化模型,我们采用 $\beta_{i,j}$ 表示从本地服务器 es_i 到邻居节点服务器 es_j 的路由路径上的平均传输速率. 由定义2中可知,a_j 为二进制变量,用来控制选择的邻居节点服务器是否能被用户访问. 假设用户所需服务不存在逻辑先后顺序,可以并行运行,则从本地服务器 es_i 到所有选中的其他服务器路由的总时延是 es_i 到所选的其他服务器之间时延的最大值,计算公式如下所示:

$$\mathrm{MAX}_{es_j\in NE_i}T_{E2E}(i,j),\tag{4}$$

式中,NE_i 为本地服务器 es_i 的邻居服务器节点集合. 用户以 es_i 为本地服务器,选择 NE_i 中的部分服务器相互协作共同提供服务需求. 用户访问所有服务 R 的总时延为:

$$T=T_{C2E}(i)+\mathrm{MAX}_{es_j\in NE_i}T_{E2E}(i,j), \tag{5}$$

式中,T 表示用户到被选中的一组服务器的总时延.

1.3 边缘协作服务选择问题

基于以上描述,我们对 ECSS 问题进行建模,模型描述如下.

定义 3 ECSS 问题 假设给定一组边缘服务器 $ES=\{es_1,es_2,\cdots,es_n\}$,任意一个边缘服务器 $es_i(es_i\in ES)$ 上已部署的服务集合为 SE_i,用户的本地服务器集合为 $ESC(ESC\subset ES)$. 用户服务需求的集合为 $R=\{s_1,s_2,\cdots,s_m\}$. ECSS 问题就是要发现一组最优的边缘服务器 ES^*,使得 ES^* 中所有服务器能够相互协作共同为用户提供服务 R,且保证用户访问这些服务的总时延最小. 即:

$$\min\ T. \tag{6}$$

约束为:

$$R\subseteq \bigcup_{es_w\in ES^*} SE_w, w\in[1,n], \tag{7}$$

$$\forall\, c_i,a_j\in\{0,1\}, es_i\in ESC. \tag{8}$$

目标函数(公式 6)表示用户选择一组边缘服务器集合 ES^* 的访问时延总和,值越小越好. 约束(7)保证所选的一组边缘服务器 ES^* 中所有服务器提供的服务集合包含了用户所需要的服务需求 R. 而集合 ES^* 中的任何一个服务器 es_w 都是从本地服务器节点 es_i 及其邻居节点 NE_i 所构成的集合 $\{es_i\}\cup NE_i$ 中选择的. 约束(8)是两个二进制变量,c_i 表示边缘服务器 es_i 是否为本地服务器节点,如果是本地服务器,则 $c_i=1$,否则为非本地服务器;而 a_j 用来控制所选的邻居节点 es_j 是否可以访问;如果 $a_j=1$,表示 es_j 可以访问,否则不可访问;$es_i\in ESC$ 用来控制本地服务器节点的选择必须从本地服务器集合 ESC 中选择.

由 ECSS 问题定义可知,该问题的任何解都可以在多项式时间内搜索到并验证所求的解是否满足约束条件(7)和(8),因此该问题是 P 问题也是 NP 问题. 此外,ECSS 问题可通过经典的加权集合覆盖问题(Weighted Set covering,WSC)约简获得(由于篇幅原因,在这里不做详细的证明). 所以 ECSS 问题是一个 NP-hard 问题.

2 方法设计

解决 ECSS 问题的方法就是在满足用户服务约束的情况下选择一组服务器集合保证提供服务的总时延最小. 因为 ECSS 问题是 NP-hard 问题,所以本文提出了一个解决 ECSS 问题的一般性框架,设计了一种基于低时延多有效服务的贪心选择策略的新启发式算法,即 LLMES 算法. 通过该算法求解 ECSS 问题的最优解.

2.1 LLMES 算法思想

LLMES 算法的主要思想是从当前本地服务器 ESC 中的 1 个元素 $es_i(es_i\in ESC, i\in[1,n])$ 出发,计算 es_i 的邻居节点集合 NE_i. 从 NE_i 中选择 1 组边缘服务器和 es_i 组成集合 ES^*,使得 ES^* 集合中的所有边缘服务器相互协作共同提供的服务组成集合 S^* 能够满足用户需求,即 $S^*\supseteq R$. 在 NE_i 中选择当前最优的边缘服务器要满足两个条件,一是 es_i 到该服务器的路由时延最短,二是能够提供有效服务的数量最多. 我们用排序指标(Ranking Criterion)作为选择当前最优服务器策略的衡量标准. 排序指标的计算公式如下所示:

$$cr_j=\frac{T_{ij}}{|(R-S^*)\cap SE_j|}\ \forall\, es_j\in NE_i, \tag{9}$$

式中,cr_j 表示边缘服务器 es_j 的排序指标值,T_{ij} 表示边缘服务器 es_i 与 es_j 的最短路由时延,可以根据公式(3)计算. R 表示用户所需的服务集合,S^* 表示当前已经选中的服务集合,SE_j 表示边缘服务器 es_j 上包含的服务集合. 边缘服务器 es_j 上包含有效服务(即用户所需的且未被选中的服务)数量越多,则公式(9)的分母就越大,对应的 cr_j 的值就越小. 当分子 T_{ij} 的值越小,服务器之间的路由时延越小,对应的 cr_j 值越小. 根据当前服务器的 cr 值来选择当前最优的服务器,值越小越好. 按照低时延多有效服务的选择策略计算邻居集合中每个服务器的 cr_j 值,选择 cr 值最小的服务器并入 ES^* 中,并将该服务器上能够提供的服务放入服务集合 S^* 中,直到选中所有服务 R 为止.

2.2　LLMES 算法描述

整个求解过程分成两个部分,第一部分先求解本地服务器 es_i 的邻居节点 NE_i. 第二部分在 NE_i 中选择最优服务器组合,使得用户到这些服务器上的时延最小,并且组合的服务器能够包含用户需要的所有服务 R. 详细算法描述如下.

步骤 1:初始化边缘服务器集合 ES,用户需求服务为 R,服务需要传输的数据向量为 UC,边缘服务器网络 G 中边集为 E,用户到本地服务器之间的传输速率向量为 CV,服务器之间的传输速率 V,服务器部署服务矩阵 D,本地服务器向量 ESC,服务器路由阈值为 hp_{limit}.

步骤 2:先从 ESC 中选择一个本地服务器 es_i 放入 ES^* 中,根据公式(2)计算用户到 es_i 的时延 T_{C2E}. 在 G 中采用 Dijkstra 算法求 es_i 到其他服务器的路由路径距离 hp_{ij}.

步骤 3:选择满足 $hp_{ij} \in [0, hp_{\text{limit}}]$ 的边缘服务器节点作为本地服务器 es_i 的邻居节点集合 NE_i.

步骤 4:根据公式(9)计算每个边缘服务器节点的排序指标 cr,选择最小值对应的边缘服务器放入 ES^*,根据公式(3)计算 es_i 到该服务器的路由时延 T_{ij}.

步骤 5:重复步骤 4,直到 ES^* 集合中所有服务器提供的服务包含 R,选择 es_i 到 $ES^* - \{es_i\}$ 中服务器时延最大值作为服务器的路由时延 T_{E2E},计算总时延 T.

步骤 6:重复步骤 2 到步骤 5,选择总时延 T 最小的一组边缘服务器 ES^*.

步骤 7:输出 ES^* 和对应的时延 T.

在 LLMES 算法运行过程中,采用 Dijkstra 算法求解图中的一个顶点到其他顶点的最短路径的时间复杂度是 $O(n^2)$,故求解本地服务器的邻居节点 NE_i 的时间复杂度为 $O(n^2)$. 而每一次从本地服务器 es_i 及 NE_i 中寻找最优解时,都要遍历 NE_i 判断是否存在解. 最坏情况下,需要选择 NE_i 中所有的服务器来提供服务,这个过程的时间复杂度为 $O(n^2)$. 所以从一个本地服务器 es_i 开始选择一组服务的时间复杂度为 $O(n^2)+O(n^2)$. 在实际应用中对于用户来说本地服务器个数一般为常量 k. 因此我们提出的 LLMES 算法的时间复杂度为 $O(n^2)$.

3　实验分析

本文仿真实验的环境是 Intel(R)Core(TM)i7-7500U CPU 笔记本,主频为 2.70~2.9 GHz,内存为 8.00 GB. 仿真软件为 MATLAB R2016a. 实验数据来自于 EUA 真实数据集[21],该数据集中包含墨尔本 1 464 个真实世界基站的地理位置. 我们从 EUA 中随机选择某个更小区域内 n 个基站作为边缘服务器. 此外,我们参考文献[18]设置用户到边缘服务器上行速率 α_i 为(1,3)MB/s,服务器之间的路由速率为 $\beta_{i,j}$ 为(80,100)MB/s.

为了验证本文提出 LLMES 方法的性能和效率,我们选择 Random 方法,GES 方法和 GMT 方法与其进行比较. 这 3 种方法详细描述如下:

Random:该方法是在 es_i 的邻居节点 NE_i 中随机选择一个边缘服务器 es_j,为用户提供所需服务,如果不能满足需求 R,则继续随机选择其他服务器,直到选中的服务器集合能够提供用户所需的服务 R 为止.

GES[22]:给定一组边缘服务器 ES 和若干个本地服务器 ESC,选择 ESC 中某个节点 es_i 及其邻居 NE_i 构成候选服务器集合,选择服务器所提供的有效服务最多作为当前最优边缘服务器. 重复这个选择步骤,直到选中的服务器能够提供所有的服务 R 为止. 最后在所有候选解中选择总延迟最小的解作为最终解.

GMT:给定一组边缘服务器 ES 和若干个本地服务器 ESC,选择 ESC 中某个节点 es_i 及其邻居 NE_i 构成候选服务器集合. 选择从本地服务器 es_i 到该服务器的路由延迟最小的那个服务器作为当前最优边缘服务器. 重复这个选择步骤,直到选中的边缘服务器能够提供所有的服务 R 为止. 最后在所有候选解中选择总延迟最小的解作为最终解.

3.1　评价指标和参数

为了更直观体现本文提出的 LLMES 方法的有效性,我们选择访问时延和运行时间两个评价指标. 并深入分析与 ECSS 问题场景相关的 5 个参数对运行结果的影响. 这 5 个参数分别是:

(1)边缘服务器数量($n=|ES|$):该参数表示边缘服务器网络图的规模大小.

(2)服务数量($m=|R|$):该参数用来表示当前用户所需的服务数量.

(3)图密度($gd=|E|/n$):图的边数和顶点个数的比值称为图密度. gd 越大,图边数越多,对应服务器之间相互协作的效果越好.

(4)路由阈值 hp_{limit}:允许服务器之间能够路由的最大跳数,两个服务器的路由距离不能大于 hp_{limit},否则它们不能相互通信.

(5)服务部署规模 ss:每个服务器能够部署的服务数量.

对每个参数进行不同取值范围的设置. 每次设置时选择一个参数在一定范围内变化,而其他参数不变. 参数详细的设置情况如表 1 所示,分别是#1,#2,#3,#4 和#5. 例如#1 设置中只有参数 n 是变化的,从初值 10 开始以步长为 5 的方式增加到 30. 其他参数值固定不变,分别是:$m=7$,$gd=1.4$,$hp_{\text{limit}}=3$,ss 取值为[3,4]. 每一组实验都是在参数设置相同的环境下运行 100 次取平均值作为最后结果.

表 1　实验参数设置

Table 1　Experimental parameter setting

参数	#1	#2	#3	#4	#5
n	10,15…,30	15	15	15	15
m	7	5,7…,13	7	7	7
gd	1.4	1.4	1.2…,1.6	1.4	1.4
hp_{limit}	3	3	3	1,2,3,4	3
ss	[3,4]	[3,4]	[3,4]	[3,4]	[1,2]…,[5,6]

比较 LLMES,Random,GES 和 GMT 4 种方法在不同参数设置情况下选择服务的总时延和运行时间. 分析每个参数对运行结果的影响如图 3-图 12 所示.

3.2　实验结果分析

(1)边缘服务器数量的影响

图 3 和图 4 分别是在#1 设置情况下 4 种方法获得的时延和运行时间. 由图 3 可知,LLMES 算法选择服务的时延相比其他 3 种方法最低,Random 方法最高. 在 EUA 数据集上,LLMES 算法计算的平均时延比 Random,GES 和 GMT 方法分别降低了 79.3%,69%和 52.3%. 且随着边缘服务器数量的增加,4 种方法的时延略有变化. LLMES 方法变化幅度最小,其他 3 种方法略大. 这是因为服务数量固定不变,服务器数量虽然增多,但被选中的服务器数量变化不大,所以对时延没有太大影响. 由于 4 种方法选择的策略不同,所以表现出不同的变化幅度. 图 4 中,LLMES 方法的运行时间相比其他 3 种方法最长. 但是和时延节省的时间相比,可忽略不计. 其平均运行时间和平均时延分别为 12.61 ms 和 50.47 ms;Random 方法的运行时间最短,但时延最高,分别为 3.85 ms 和 243.78 ms;GES 为 9.17 ms 和 163.62 ms;GMT 为 6.29 ms 和 105.89 ms. 所以本文提出的方法的性能优于其他 3 种方法. 此外,随着边缘服务器数量增多,4 种方法的运行时间整体是增加趋势. 这是因为随着服务器数量增加,需要比较的候选服务器数量也随之增加,所以运行时间呈现增加趋势. 服务器数量对时延的影响较小,对运行时间的影响较大.

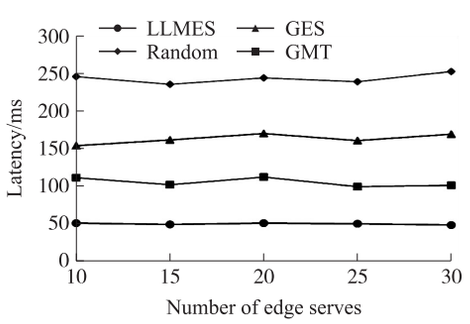

图 3　不同边缘服务器数量的时延

Fig. 3　Latency for different number of edge servers

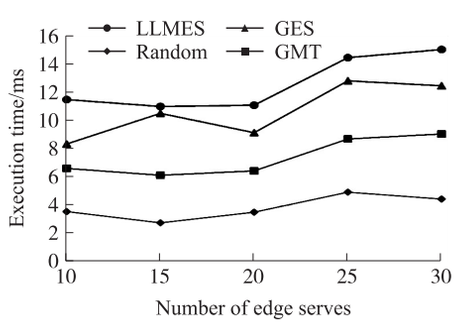

图 4　不同边缘服务器数量的运行时间

Fig. 4　Execution time for different number of edge servers

(2)服务数量的影响

图 5 和图 6 分别是在#2 设置情况下的运行结果. 由图 5 可知,LLMES 算法选择服务的时延最低,比 Random,GES 和 GMT 方法分别降低了 75.5%,66.51%和 54.4%. 随着服务数量增加,4 种方法获得的时延

均随之增加. LLMES 方法增速较慢,其他 3 种方法增速较快. 这是因为随着用户所需服务数量的增加,服务要传输的数据量也随之增加,且需要更多的边缘服务器来提供服务. 所以用户调用服务的时延表现出增加趋势. 由于选择的策略不同,因此表现出增速不同. 由图 6 可知,LLMES 方法运行的时间最长,但时延最小,其平均运行时间和平均时延分别为 11.53 ms 和 63.27 ms;Random 运行时间最短,但总时延最长,平均值分别为 3.84 ms 和 250.87 ms;GES 为 182.83 ms 和 9.82 ms;GMT 为 136.76 ms 和 6.36 ms. 所以本文提出的方法性能优于其他 3 种方法. 随着服务数量的增多,需要选择更多的服务器提供服务,整个搜索过程增加,所以 4 种方法的运行时间整体上都在增加. 服务数量的变化对 4 种方法的运行结果影响较大.

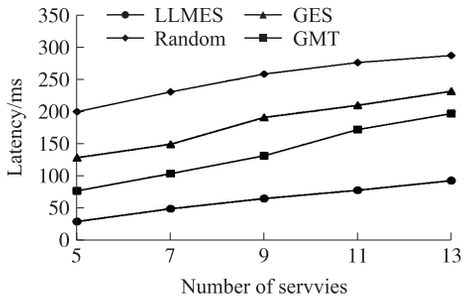

图 5　不同服务数量的时延

Fig. 5　Latency for different number of services

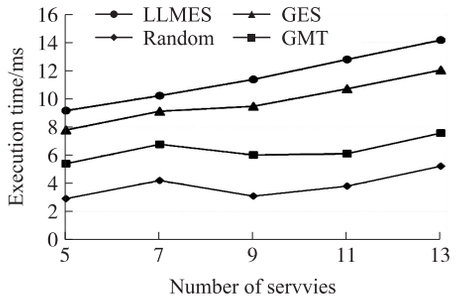

图 6　不同服务数量的运行时间

Fig. 6　Execution time for different number of services

(3)图密度的影响

图 7 和图 8 分别是在#3 设置情况下获得的运行结果. 由图 7 可知,LLMES 算法的时延最低,比 Random,GER 和 GMT 方法分别降低了 78%,68.4%和 52.05%. 图密度的增加使得时延有所降低但幅度较小. 这是因为随着图密度增加,与本地服务器相连且跳数小的服务器数量也随之增加,所以时延降低. 图密度增加使得边缘网络中的边数增加,但与服务器数量相比可忽略不计. 所以图密度对时延影响较小.

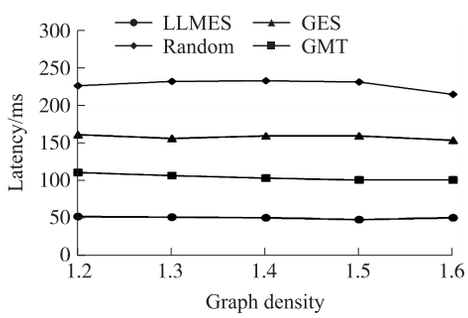

图 7　不同图密度的时延

Fig. 7　Latency for different graph densities

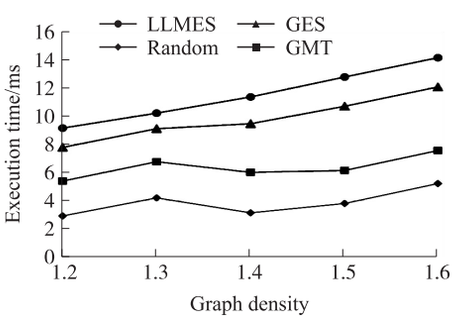

图 8　不同图密度的运行时间

Fig. 8　Execution time for different graph densities

由图 8 可知,LLMES 方法运行时间最长,Random 最短. LLMES 方法的平均运行时间和平均时延分别是 10.42 ms 和 49.75 ms;Random 方法分别是 2.95 ms 和 226.72 ms;GES 为 8.65 ms 和 157.46 ms;GMT 为 5.83 ms 和 103.82 ms. 所以本文提出的方法的性能优于其他方法. 随着图密度的增多,4 种方法因为选择的策略不同而使得运行时间变化趋势不同,但整体上呈现增加趋势. 这是因为随着图密度的增加对应候选服务器的数量增加,所以需要更多时间去选择服务. 图密度对服务选择的时延影响较小,但对运行时间的影响较大.

(4)路由阈值的影响

图 9 和图 10 分别是在#4 设置情况下的运行结果. 在图 9 中,LLMES 算法的时延比 Random、GES 和 GMT 方法分别降低了 74.4%、63.6%和 50.3%. 随着路由阈值的增加,LLMES 算法和 GMT 方法的时延基本上没有什么变化,而 Random 方法和 GES 方法的时延却随之增加. 这是因为前两者的方法是根据服务器所提供服务的时延最小进行选择,而跳数小的服务器数量固定,所以获得时延基本不变. Random 方法是随机选择服务器,随着路由阈值的增加,候选服务器之间最大的路由距离增加,时延也随之增加. GES 方法是依据能够供有效服务数量最多来选择有效服务器的,所以时延也随之增加.

在图 10 中,4 种方法的运行时间性能对比与图 8 相同. 随着 hp_{limit} 值的增加,LLMES 和 GES 方法的运行时间均增加,而 Random 和 GMT 方法运行时间虽然有所减少,但变化不大. 这是因为随着路由阈值的增加,能够相互路由的服务器数量增加. 所以候选服务器的数量随之增加. LLMES 和 GMT 方法在选择能够提供最多有效服务的服务器时间也随之增加,所以总体运行时间增加. 而 Random 和 GMT 方法由于服务器节点的增多,可以选择的服务器数量增加,而遇到不能提供有效服务的服务器概率变小,所以运行时间减少. 参数 hp_{limit} 对 4 种方法的时延和运行时间的影响不同.

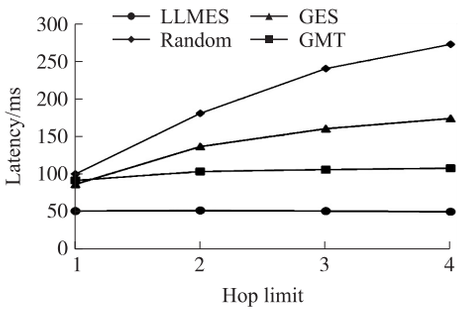

图 9　不同路由阈值的时延

Fig. 9　Latency for different routing thresholds

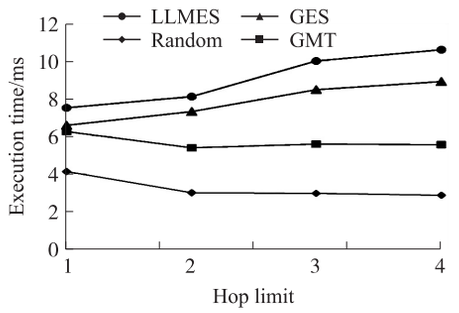

图 10　不同路由阈值的运行时间

Fig. 10　Execution time for different routing thresholds

(5)服务部署规模的影响

图 11 和图 12 是在#5 设置情况下的运行结果. 由图 11 可知,LLMES 算法获得的时延最低,比 Random、GES 和 GMT 方法分别降低了 78.87%、67.54%和 53.50%. 随着参数 ss 的增加,4 种方法的时延均随之降低. 这是因为参数 ss 的增加,与本地服务器路由距离较近的服务器上部署的服务数量也随之增加,所以服务的时延降低.

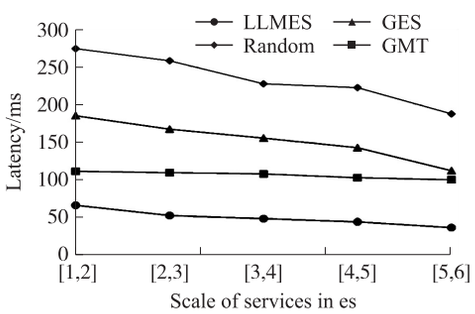

图 11　不同服务规模的时延

Fig. 11　Latency for different scale of services

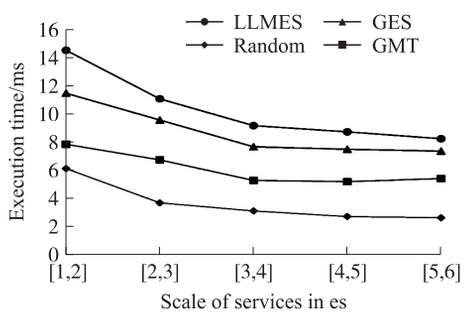

图 12　不同服务规模的运行时间

Fig. 12　Execution time for different scale of services

在图 12 中,LLMES 方法运行时间最长但时延最低,Random 运行时间最少但时延最高. LLMES 方法的平均运行时间和平均时延分别是 10.32 ms 和 49.47 ms;Random 分别是 3.61 ms 和 234.28 ms;GES 为 8.66 ms 和 152.46 ms;GMT 为 6.05 ms 和 106.42 ms. 所以 LLMES 方法的性能优于其他方法. 随着边缘服务上部署的服务数量的增加,4 种方法的运行时间均减少. 这是因为随着参数 ss 的增加,只需选择较少的服务器就能为用户提供所有服务,所以运行时间也随之减少. 参数 ss 的增加能够降低服务时延和运行时间.

通过上述 5 组实验的对比分析,本文提出的 LLMES 算法选择服务的总时延最低. 虽然在运行时间上比其他 3 种方法略高,但是和时延节省的时间相比可忽略不计. 所以本文提出的 LLMES 算法的性能优于其他 3 种算法.

4　结论

本文针对边缘计算环境下用户业务需求复杂多样和时延敏感的问题,提出了多个边缘服务器相互协作的思想. 将 ECSS 问题建模成带约束的最优化问题并给出一种新的启发式算法 LLMES 来解决该问题. 该算法采用低时延多有效服务的贪心选择策略选择当前能够提供最小时延服务集的最优服务器集合. 通过在 EUA 数据集上仿真实验分析,LLMES 算法比 Random、GES 和 GMT 方法的运行时间平均增加

了 7.35 ms,1.69 ms 和 4.57 ms;而时延平均减少了 179.29 ms,107.11 ms 和 58.53 ms. 本文提出的 LLMES 算法在边缘协作环境下选择服务的总时延更低,整体性能更好.

[参考文献]

[1] ZANELLA A,BUI N,CASTELLANI A,et al. Internet of Things for smart cities[J]. IEEE internet of things journal,2014,1(1):22-32.

[2] DEBAUCHE O,MAHMOUDI S,MAHMOUDI S A,et al. Edge computing and artificial intelligence for real-time poultry monitoring[J]. Procedia computer,2020,175(1):534-541.

[3] 崔勇,宋健,缪葱葱,等. 移动云计算研究进展与趋势[J]. 计算机学报,2017,40(2):273-295.

[4] DENG S G,HUANG L T,HU D N,et al. Mobility-enabled service selection for composite services[J]. IEEE transactions on services computing,2016,9(3):394-407.

[5] YU H Y,LIU C Y,REN Y L,et al. Service node selection optimization for mobile crowd sensing in a road network environment[J]. Vehicular communications,2020,22(4):100203.1-100203.14.

[6] DENG S G,WU H Y,TAN W,et al. Mobile service selection for composition:an energy consumption perspective[J]. IEEE transactions on automation science and engineering,2017,14(3):1478-1490.

[7] TONG E D,CHEN L,LI H Z. Energy-aware service selection and adaptation in wireless sensor networks with QoS guarantee[J]. IEEE transactions on services computing,2020,13(5):829-842.

[8] ZHANG W W,WEN Y G. Energy-efficient task execution for application as a general topology in mobile cloud computing[J]. IEEE transactions on cloud computing,2015,6(3):708-719.

[9] 赵梓铭,刘芳,蔡志平,等. 边缘计算:平台、应用与挑战[J]. 计算机研究与发展,2018,55(2):327-337.

[10] HU 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.

[11] MAO Y Y,YOU C S,ZHANG J,et al. A survey on mobile edge computing:the communication perspective[J]. IEEE communications surveys & tutorials,2017,19(4):2322-2358.

[12] 乐光学,戴亚盛,杨晓慧,等. 边缘计算可信协同服务策略建模[J]. 计算机研究与发展,2020,57(5):1080-1102.

[13] WU H Y,DENG S G,LI W,et al. Service selection for composition in mobile edge computing systems[C]//2018 IEEE international conference on Web services,San Francisco,CA,USA,2018:355-358.

[14] 刘伟,黄宇成,杜薇,等. 移动边缘计算中资源受限的串行任务卸载策略[J]. 软件学报,2020,31(6):309-328.

[15] XIE N,TAN W A,ZHENG X R,et al. An efficient two-phase approach for reliable collaboration-aware service composition in cloud manufacturing[J]. Journal of industrial information integration,2021,23(2):1-12.

[16] GUO H Z,LIU J J. Collaborative computation offloading for multi-access edge computing over fiber-wireless networks[J]. IEEE transactions on vehicular technology,2018,67(5):4514-4526.

[17] CHEN L X,ZHOU S,XU J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks[J]. IEEE/ACM transactions on networking,2018,26(4):1619-1632.

[18] DENG S G,XIANG Z,TAHERI J,et al. Optimal application deployment in resource constrained distributed edges[J]. IEEE transactions on mobile computing,2020,20(5):1907-1923.

[19] GUI G M,HE Q,CHEN F F,et al. Trading off between multi-tenancy and interference:a service user allocation game[J/OL]. IEEE transactions on services computing,2020. Available:10.1109/TSC.2020.3028760.

[20] DENG S G,XIANG Z Z,ZHAO P,et al. Dynamical resource allocation in edge for trustable IoT systems:a reinforcement learning method[J]. IEEE transactions on industrial informatics,2020,16(9):6103-6113.

[21] XIA X Y,CHEN F F,HE Q,et al. Cost-effective app data distribution in edge computing[J]. IEEE transactions on parallel and distributed systems,2021,32(1):31-44.

[22] PASTERIS S,WANG S Q,HERBSTER M,et al. Service placement with provable guarantees in heterogeneous edge computing systems[C]//IEEE conference on computer communications. Chengdu,China,2019:514-522.

[责任编辑:顾晓天]