计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(12):28−36 [doi: 10.15888/j.cnki.csa.008289]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

边缘计算中的计算卸载综述①

郑会吉[1], 余思聪[1], 崔翛龙[1,2], 朱　利[1,2], 柏财通[1]

[1](中国人民武装警察部队工程大学, 西安 710086)
[2](中国人民武装警察部队工程大学 反恐指挥信息工程联合实验室, 西安 710086)
通讯作者: 崔翛龙, E-mail: ecxl@qq.com

摘　要: 边缘计算可以有效解决传统云计算中传输时延大、用户数据安全性不够高、传输带宽压力大以及终端移动设备计算能力受限、能耗大等问题. 计算卸载是边缘计算中的关键技术, 针对当前计算卸载技术的研究现状和存在的不足, 本文围绕计算卸载, 首先介绍边缘计算的体系架构以及部分应用和分析 4 种主要的影响因素以及相应具体的条件; 其次针对 3 种决策目标分析了算法策略及对应变量在算法中的作用; 最后总结目前在计算卸载中存在的不足.

关键词: 边缘计算; 计算卸载; 卸载策略; 优化算法; 安全性

引用格式: 郑会吉,余思聪,崔翛龙,朱利,柏财通.边缘计算中的计算卸载综述.计算机系统应用,2021,30(12):28–36. http://www.c-s-a.org.cn/1003-3254/8289.html

Survey on Computing Offloading in Edge Computing

ZHENG Hui-Ji[1], YU Si-Cong[1], CUI Xiao-Long[1,2], ZHU Li[1,2], BAI Cai-Tong[1]

[1](Engineering University of PAP, Xi'an 710086, China)
[2](Joint Counter-Terrorism Command Information Engineering Laboratory, Engineering University of PAP, Xi'an 710086, China)

Abstract: Edge computing can effectively solve the problems of large transmission delay, insufficient user data security, high transmission bandwidth pressure, limited computing capabilities of terminal mobile devices, and high energy consumption in traditional cloud computing. Computing offloading is a key technology in edge computing. Concerning the research status and existing deficiencies of computing offloading technology, this study first introduces the architecture of edge computing and some applications and analyzes the four main influencing factors and corresponding specific conditions. Secondly, the algorithm strategy and the role of corresponding variables in the algorithm are analyzed in three decision objectives. Finally, it summarizes the current deficiencies in computing offloading.

Key words: edge computing; computing offloading; offloading strategy; optimization algorithm; security

1　引言

边缘计算的概念源于早期的内容分发网络, 发展于传统的云计算, 其核心思想是在网络边缘部署服务器和存储设备——使网络的边缘也具备强大的计算和存储能力. 其整体结构可按其计算能力、服务功能划分, 如图 1 所示, 分别是云-边-端三层, 云层主要是由大型云服务器组成, 具有丰富的计算资源和超强的计算能力; 边缘层主要是由基站、边缘网关、边缘服务器等组成, 相比云层略弱一些, 但可以支持就近用户的部分数据处理需求; 端层主要是由传感器、移动设备等组成, 分布广泛, 数量众多, 产生大量原始数据, 但由于其单个设备计算和存储容量有限, 往往需要将计算任

① 基金项目: 网信融合项目基金 (LXJH-10(A)-09)
Foundation item: Foundation of Internet and Information Integration Project (LXJH-10(A)-09)
收稿时间: 2021-03-25; 修改时间: 2021-03-31; 采用时间: 2021-05-14

务卸载到外部服务器. 传统的方式是上传到云服务器, 即云计算. 但随着用户数据爆炸式的增长, 这种方法暴露了一些不足, 即由于传输带宽、数据处理和物理距离带来的时延问题, 较高的传输成本. 基于此, 边缘计算应运而生, 它是一种新型计算模式, 但边缘计算并不是取代云计算, 而是对云计算的补充和延申. 顾名思义, 边缘就是指在传统的数据源到云中心路径中, 靠近数据端设置的计算节点, 而在边缘计算中, 计算卸载是一项关键技术. 计算卸载, 可以是指设备与设备之间的任务传输; 也可以指终端设备将计算量大的任务按照一定策略, 通过无线网络分配到计算资源充足的服务器进行处理, 服务器再把计算结果返回给终端设备的过程.

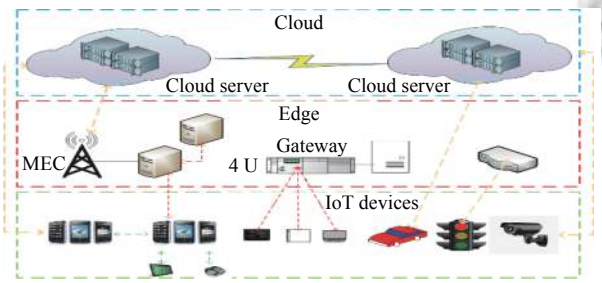

图 1 边缘计算整体架构

计算卸载的研究中, 主要包括卸载策略和资源管理两部分内容, 这两部分内容也是边缘计算中的关键技术, 卸载策略是解决是否需要卸载、部分卸载还是全部卸载、卸载到哪的问题; 资源管理是针对边缘节点, 即如何分配资源能使利益最大化. 本文将结合边缘计算环境下计算卸载的研究现状, 对有关影响因素和卸载策略进行详细论述, 并提出了目前仍存在的问题, 为下一步研究提供参考和方向. 在整个结构中, 卸载可以发生在不同层级的不同组成部分之间, 如图 2 所示, 不仅有终端层设备间[1](例如智能手表和手机间)、终端层到边缘层、终端层到云层等平级和由下级到上级之间, 也有云层到终端层 (监控搜索)、边缘层到终端层 (收集数据) 等不太常见的情况[2]. 终端设备群成功利用了边缘节点的优势, 一定程度上缓解了计算资源不足, 时延、带宽和能耗等问题, 相对应的, 终端设备群的海量数据信息则可以为边缘节点和云中心所用, 通过数据分析满足用户需求.

基于前文对边缘计算的介绍, 其优势使得边缘计算可以应用于在线游戏、视频服务、无人驾驶等多个应用场景, 而且随着万物互联时代的到来, 边缘计算的应用场景会更加广泛. 例如视频分析、内容缓存于分发等视频业务, 增强现实/虚拟现实、物联网网关、车联网、智慧城市以及工业互联网场景等. 例如边缘计算产业联盟 ECC 发布的“2020 边缘计算十大解决方案”中[3], 铸管行业生产过程中承口随机铸字进行识别, 以获取准确的铸管编码字符, 但相关工艺操作后会对原有字符造成不可逆的破坏, 导致识别难度极大增加.

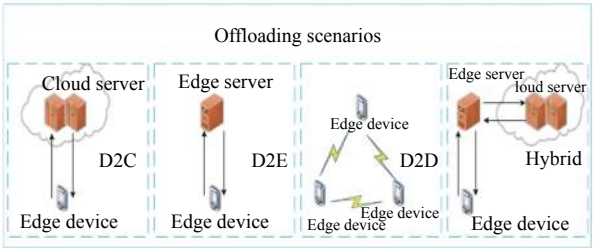

图 2 卸载场景

该项目的核心是早边缘侧部署了自主研发的 Thing-Origin 边缘计算平台, 该平台不仅完成边缘侧设备的数据采集、存储工作, 且在边缘侧直接对设备数据和图像进行基于深度学习的人工智能算法处理计算, 如图 3 所示, 通过大量字符图像数据的迭代训练, 代替人工实现在高温环境作业下的残缺字符结果识别[3].

针对视频边缘计算, 中国移动研究院在《视频边缘计算白皮书 (2020)》中, 专门分析了其发展背景, 边缘计算重新塑造了运营商的网络价值, 其场景丰富, 而且设备数量巨大, 其中视频是边缘计算主要载体. 同时, 白皮书中也指出, 如何构建一套统一的视频边缘计算能力框架, 模块化的应用于视频边缘计算各个场景, 并解决各类场景中出现的各类问题, 是业界讨论的一个话题[4].

论文剩余部分的结构安排如下: 第 2 节主要分析了影响卸载效果的 4 种主要因素; 第 3 节主要围绕 3 种计算卸载目标分析了有关算法策略; 最后一节主要总结了计算卸载目前研究中的几点不足.

2 影响因素

在实际应用中, 计算卸载方案会受到诸多因素的限制, 包括设备、网络、服务器和用户需求等. 设备因素主要是指硬件结构、处理能力、存储能力和操作系统[5]. 我们选择边缘计算就是为了弥补终端移动智能设备在自身资源和计算能力的不足, 所以对设备因素的理解比较简单. 另外, 在边缘计算网络中, 设备众多, 比

如移动电话、平板、监控等, 不同设备一定程度上会影响卸载效果, 所以必须考虑适应设备的异构性; 对于车辆这种对实时性要求很高的应用, 设备的移动性也不能忽视[6], 边缘服务器很难覆盖所有的区域, 如果在卸载结果返回之前超出当前服务器服务范围, 就可能需要找新的服务器, 这样就会造成很大延时.

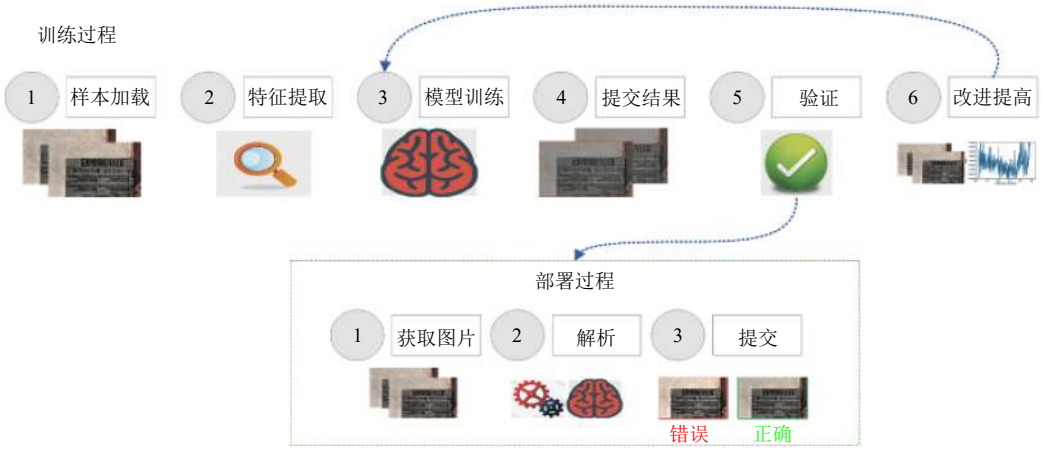

图 3 边缘视觉识别流程[7]

网络质量主要包括无线信道链路质量、终端和服务器之间的带宽和网络干扰等; 通信方面, 无线信道链路质量是重要影响因素, 在文献 [8] 中重点介绍了联合无线电和联合无线电技术. 在边缘计算中, 计算卸载的实现是靠无线网络, 无线网络则受当前环境影响较大, 在城市范围内, 效果比较好, 利用 5G 网络可以实现很好的效果, 但对于一些偏僻的地区, 通信资源差, 还容易对无线信道造成衰减, 通信成本势必会增加; 设备和服务器, 服务器之间的带宽决定了数据传输速率, 直接影响传输时间; 文献 [9] 中介绍了基于编译器代码分析的新颖技术, 该技术通过仅传输必需的堆对象来有效减少传输的数据大小, 从而减小传输时间.

服务器主要包括计算能力、物理距离和访问技术等; 当边缘设备需要卸载时, 会根据特定的卸载策略, 考虑服务器的计算能力、物理距离等选择适合的服务器, 同时, 终端设备与服务器之间的访问技术对卸载效果也有影响; 在引言中提到卸载的情境有很多, 不只有平级和下级到上级的情形, 也有云层到终端层、边缘层到终端层等不太常见的情况. 在整个边缘网络中, 设备距服务器的距离是不一样的, 这将导致不同的延时; 文献 [10] 中考虑了有限的缓存空间, 应仔细选择在 BS 缓存的内容, 以提高缓存效率, 通过考虑通过设备到设备 (D2D) 通信的流量分流来研究 BS 处的边缘缓存, 以最大程度地降低传输成本, 将边缘缓存问题建模为马尔可夫决策过程, 并提出基于 Q 学习的分布式缓存替换策略; 文献 [6] 中提出了一个受任务完成时间和有限的移动边缘云计算能力约束的能源成本最小化问题, 然后, 利用基于凸函数 (DC) 编程和线性编程之差的替代优化 (AO), 设计了一种用于时钟频率控制, 传输功率分配, 卸载比和功率分配比的迭代算法, 以解决非凸优化问题.

用户需求主要包括安全性、成本和能耗等; 边缘计算的其中一个特点是可以保护用户数据安全; 通信成本也是必须考虑的, 包括服务费和流量成本, 取决于用户使用服务器的种类、次数和时长[11]; 设备能耗直接影响了用户体验.

如表 1 所示, 本节主要总结分析了影响卸载策略的 4 种主要因素, 在不同场景中选择合适的卸载策略必须综合考虑到这些因素. 其中有几个问题比较突出, 例如设备种类中的设备异构性问题和移动性管理问题, 在传统的蜂窝网中, 用户在服务区之间移动时, 有一套严格的切换方案保证服务连续性, 类似地, 当终端设备的计算任务卸载到边缘节点, 如何保证服务连续性是要解决的关键问题; 网络质量中的干扰问题, 由于边缘计算采用分布式架构, 海量终端的卸载处理请求和复杂多变的网络环境降低了资源使用率, 从而产生严重的干扰问题; 用户需求中的安全性问题, 边缘计算虽然减少了终端用户与云中心的交互过程, 一定程度上降低了用户隐私数据被窃取的概率, 其他场景下的卸载过程本质上也是数据传输, 所以如何保障边缘节点及通信信道的安全性仍需进一步研究.

表 1 影响卸载策略的因素

影响因素	参考文献	具体条件
设备种类	[5]	硬件结构、处理能力、存储能力和操作系统
网络质量	[8,9]	无线信道链路质量、终端和服务器之间的带宽和网络干扰等
服务器	[6,10]	计算能力、物理距离和访问技术等
用户需求	[11]	安全性、成本和能耗等

3 目标策略

计算卸载就是把终端设备需要执行但自身难以支持的任务上传到就近边缘服务器执行而后把计算结果传送回终端的过程. 它的基本过程包括: 程序区分、节点选择、数据传输、边缘执行、结果回传. 程序区分, 即终端设备将总任务根据它们之间的依赖关系分解为若干子任务并决策哪些需要卸载, 哪些本地执行. 如图 4 所示, 某 APP 要实现导航功能, 则可将导航这个总任务分解为地图和交通两个子任务[12], 根据设计的策略, 可把两个子任务卸载到边缘服务器上, 使得执行总时延最小, 两者相互独立但又缺一不可; 节点选择, 即终端设备根据自身计算能力、附近边缘服务器计算能力、与自身物理距离等选择合适的边缘服务器; 数据传输, 即终端设备选择好合适服务器后传输数据的过程, 通常采用无线网络传输; 边缘执行, 即边缘服务器执行终端设备卸载的任务; 结果回传, 即边缘服务器将计算结果返回到终端设备.

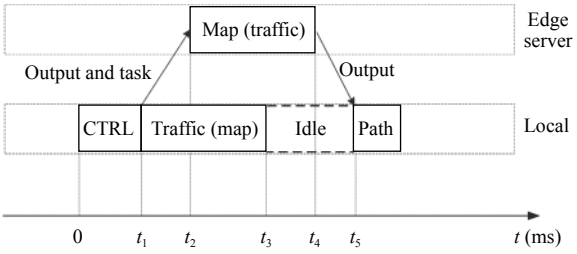

图 4 APP 子任务卸载

卸载策略是在不同应用场景下, 根据实际情况, 比如设备类型、通信资源环境、现有服务器类型以及具体用户需求, 结合不同算法特点而设置的. 计算卸载的衡量因子有很多, 时延和能耗的优化是核心点. 主要可以分为 3 种目标类型: 时延最小、能耗最小、时延能耗和最小.

3.1 能耗最小

系统中的能耗主要包括终端设备本地计算的能耗、传输能耗以及边缘服务器计算的能耗, 研究中通常用功率来量化.

不同应用和设备的能耗是不一样的, 文献 [13] 中就研究了不同类型的应用和设备的能耗, 提出了一个分析模型, 该模型有助于表征在云和非云应用场景下的移动设备能耗; 文献 [14] 提出了一种基于能耗的深度学习节能算法, 以设备能量、网络条件、计算量、数据传输量和通信延迟为参数设计成本函数, 通过组件卸载策略的所有可能组合成本, 训练了深度学习网络作为决策模型, 结果表明, 该模型是有效的.

文献 [11] 研究了多用户、多小蜂窝网场景下的卸载, 将每个任务延迟限制在阈值以下, 考虑到小蜂窝网络的前端和后端链路, 有效降低了设备能耗; 文献 [15] 研究了基于 Lyapunov 优化的动态卸载算法, 该算法具有较低的复杂性, 可以根据无线环境的变化, 将部分应用程序的计算动态地卸载到专用服务器上, 在满足移动应用程序执行时间约束的同时, 比现有算法节省更多的能量; 文献 [16] 开发了一个卸载框架, 称为三元决策者 (TDM), 旨在缩短响应时间, 同时减少能源消耗. 其执行目标包括车载 CPU、车载 GPU 和云, 所有这些结合起来为移动应用程序提供了更灵活的执行环境.

文献 [17] 介绍了用于多组件应用程序的无线感知联合调度和计算分载 (JSCO) 的概念, 其中对哪些组件需要分载以及这些组件的调度顺序进行了最佳决策. JSCO 方法通过从编译器为组件预先确定的调度顺序转移到更具无线意识的调度顺序, 从而在解决方案中提供了更大的自由度数; 计算卸载时, 分解的子任务之间可能是互相独立的, 也可能存在数据依赖, 目前对卸载部分考虑最多的是相互独立, 但实际中绝大部分都是相互依赖的, 对于某些组件依赖图结构, 如图 5 所示, 顶点集$M=(m_1,m_2,\cdots,m_6)$表示需要卸载的应用程序可分解的子任务, $V=(v_{12},v_{23},\cdots,v_{16})$表示子任务之间的相关性, 所提出的算法可以通过并行处理移动设备和云中的适当组件来缩短执行时间.

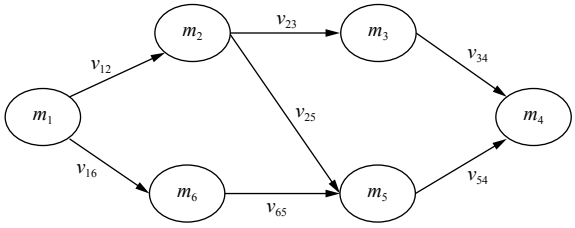

图 5 子任务依赖关系

文献 [18,19] 中确定了卸载实时视频应用程序所面临的独特挑战和机遇, 并在此情况下相应开发了用

于节能计算卸载的通用模型, 提出了一种调度算法, 它可以在动态无线网络条件下的精细粒度下做出自适应卸载决策, 并通过跟踪驱动的仿真来验证其有效性; 文献 [20] 中采用马尔可夫决策过程方法来处理该问题, 根据任务缓冲区的排队状态、本地处理单元的执行状态以及传输单元的状态来调度计算任务. 通过分析移动设备上每个任务的平均延迟和平均功耗, 提出了一个功率约束的延迟最小化问题, 并提出了一种高效的一维搜索算法来寻找最优的任务调度策略. 仿真结果验证了该方法的有效; 文献 [21] 中为使终端设备平均能耗最小, 在平均时延约束下, 联合优化资源调度和计算卸载, 提出的离线动态策略是基于应用率和信道状态, 从而在不同信道状态下做决策; 文献 [22] 提出了在 5G 异构网络下的节能算法, 首先将终端设备分为 3 类, 然后依据算法给终端设备计算优先级, 最后按照此优先级分配无线电资源.

能耗问题是移动终端设备的大问题, 虽然用户可以随身携带续能电池, 但势必会影响体验需求, 例如在军事上, 单兵在野外执行任务, 负荷过大势必会严重影响单兵作战能力. 本小节总结的以最小化能耗为目标的卸载策略很大程度缓解了这个短板, 但最终的结果只能是缓解而难以克服.

3.2 时延最小

系统的时间包括计算任务上传和将计算结果回传到设备和边缘服务器执行的时间, 但在研究中, 往往为了简化问题, 将回传时间视作一个与上传和计算所需时间相比可忽略的值[23]. 研究中计算时间通常是由计算能力、计算量、网络状况等因素决定.

有的场景下, 适合将所有计算任务进行卸载, 在文献 [24,25] 中, 在每个间隔依据缓冲区的排队状态、本地和边缘服务器的执行状态以及传输信道状态, 采用马尔科夫决策过程的方法处理子任务是否卸载的问题, 通过分析设备上每个子任务的平均时延和功耗, 提出了一个功率约束的时延最小问题并通过一维搜索算法寻找最优策略.

并不是所有的计算任务都需要卸载或者只有卸载执行才更优化, 需要算法量化优先级作为是否卸载的决策依据, 文献 [10] 中就提出了针对单用户单服务器的 EFO (Earliest Finish-time Offloading) 算法, 通过量化可以卸载的每个子任务的优先级, 对比本地和卸载的执行时间, 以此来决策优先卸载哪些子任务; 而后将该算法拓展到多用户多服务器情境, 以此来协调不同用户之间的竞争; 最后对集中式和分布式的算法都进行了仿真实验, 结果表明总延时相比本地处理可以明显减少.

文献 [7,26] 中, 作者提出了针对单用户全部卸载时的动态卸载策略, 通过设置在相等时间间隔内检测卸载所需的成本, 动态的涵义是在卸载时, 改变终端设备的 CPU 主频和发射功率来降低时延. 仿真结果表明, 此方法相比本地执行可减少 64% 的时延; 文献 [27] 中, 作者利用软件定义网络 (software defined network) 的思想, 将计算卸载表示为混合整数非线性计算过程, 将降低时延问题转化为计算卸载放置问题和资源分配两个子问题, 仿真结果表明, 该方法相比统一卸载可节约 20% 的时延.

随着机器学习方法的发展, 深度强化学习在卸载优化时延中也起到了较好的应用. 文献 [28] 中, 为了最小化缓冲队列中任务的平均时间, 提出了一种基于深度强化学习 (DRL) 的算法, 将优化问题转化为学习问题; 上一小节中提到任务之间的相关性也应该考虑, 文献 [29–31] 中研究了依赖感知的计算卸载决策问题, 其目标是在有能耗约束情况下, 最小化任务执行时间, 对此文中提出了一种基于强化学习的无模型方法——Q 学习, 通过与网络环境交互, 自适应学习以优化卸载决策; 文献 [6,32] 研究了车载网络场景下多平台智能卸载与资源分配问题, 针对计算资源分配和非局部计算中的系统复杂性, 采用强化学习方法解决资源分配的优化问题, 实验结果表明, 该方案显著降低了延迟成本.

如果只考虑降低卸载时延, 而忽视了设备的能耗问题, 则终端设备可能会因为供电不足而强行终止卸载, 因此, 需要综合考虑时延和能耗的优化.

3.3 时延能耗和最小

时延和能耗是计算卸载系统中两个十分重要的性能指标, 在不同场景、不同应用以及不同用户需求下, 对两者的考虑权重可能会有差别[8,33], 某些系统中, 用户更希望系统的时间和能耗总和能达到最小, 本小节就时延能耗和最小进行总结分析.

文献 [34] 中, 作者将有时延约束的能耗最小问题转化为马尔可夫决策过程, 引入了基于在线学习的动态资源分配和预先计算离线策略, 根据应用程序属性的先验知识, 比如周期内任务到达率和无线信道干扰状况等来决定资源分配; 文献 [35] 中, 作者针对单用户提出了确定性和随机性策略, 考虑了在卸载过程中时变信道的动态环境、无线电资源调度和计算卸载的联合动态优化计算; 文献 [36] 中, 作者针对多用户全部卸

载的策略, 该策略通过使用联合优化每个移动终端的调度和计算分流策略保证用户终端的体验质量和各个终端的公平性; 文献 [37] 中, 作者提规定的出了通信和计算资源联合分配的卸载策略, 考虑多用户单服务器场景下, 在应用程序规定的平均时延约束下联合优化发送功率、分配给每个应用程序的 CPU 周期和比特数; 文献 [38] 中, 作者通过将需要卸载的任务拆分, 用贪婪算法解决最小化能耗问题, 每个子任务的优先级用最大节能衡量, 同时对比了该算法在不同边缘服务器和信道数量情况下的节能情况.

3.4 总结

本节以主要的 3 种卸载目标对有关算法进行了总结, 通过分析可以发现, 文献中会给定一个应用场景, 分析其中存在的不足与难点, 然后会设置一个系统模型, 系统模型通常包括通信模型和计算模型, 也有计算任务处理模型. 计算模型又主要包括本地计算模型和卸载模型, 当然也有一些研究中会讨论其他模型. 如表 2 所示, 给出了系统模型中研究的主要变量, 表中变量在不同文献中表示形式不一样.

表 2 系统模型中的变量

通信模型	计算模型
N: 用户服务器数 w: 系统传输带宽 P: 信道传输功率 g: 信道传输增益 σ : 信道噪声功率 b: 传输数据大小 α: 任务生成率 Q: 任务缓冲区大小 f: CPU时钟频率 (计算能力) τ: 时隙间隔 a: 所需CPU时钟周期量 γ: 能耗系数	E: 能量 t: 时间 r: 数据传输速率 ε: 优先级 ε_0: 单位CPU耗能

系统模型是计算卸载研究的背景, 虽然总体上都可以分为云-边-端-三层结构, 但在具体组成上, 还是有区别. 通常是设置一个终端设备集和边缘节点集, 终端设备通过无线传输信道访问边缘节点, 可以选择本地计算或者卸载到边缘服务器. 例如文献 [34,39,40] 中, 选择一个双层小单元网络场景, 包括用户设备、微蜂窝 (带 MEC 服务器)、核心网 3 层结构, 值得注意的是, 该论文在用户设备和微蜂窝之间加了不带 MEC 服务器的毫微微蜂窝基站, 它可以带来更低的延迟和更高的数据传输速率, 导致用户更倾向访问这些基站, 鉴于其部署密集灵活的特点, 设置了前端链路和回程链路, 从而访问 MEC 服务器; 文献 [16] 中, 场景设置比较常规, 只考虑一个终端设备和一个 MEC 服务器, 重点研究了排队过程中的时间问题, 将设备设置为 4 部分, 其中任务缓冲队列用于设备待处理任务的缓冲区, 任务调度器负责将任务调度到执行单元和传输单元, 还有本地执行单元和传输单元, 其目标是加快任务处理时间, 同时减少任务超时时间.

通信模型主要变量包括无线传输信道、设备参数、传输速率 r、优先级ε等; 无线传输信道参数包括带宽 w、传输功率 P、噪声功率σ、路径损耗因子、衰减因子、信道增益 g、状态等; 每一个设备参数包括其计算任务, 计算能力 f (一般由 CPU 时钟频率量化), 计算任务也可以一组元组的形式表示, 比如可表示为 $A(b,\tau,x)$其中 b 表示输入数据大小, τ表示时延期限, x 表示每计算 1 bit 所需 CPU 周期数; 在本地可能会设置一个任务缓冲区, 参数包括其大小 Q, 任务生成率α, 执行状态等; 系统传输速率可通过香农定理求出, 如式 (1) 所示:

$$r = w\log_2\left(1+\frac{gp}{nw}\right) \tag{1}$$

所得结果可作为系统时延约束的计算依据; 系统还会设置优先级 (通常是由单个任务的能耗和时间、本地计算和卸载到服务器计算的能耗差、时延差决定), 优先级作为选择策略的依据之一.

计算模型中主要是计算系统的各个部分的能耗和时间. 能耗通常包括两部分, 任务处理过程和数据传输过程消耗的能量, 处理过程中消耗的能量, 如式 (2) 所示:

$$e_{处} = kbxf^2 \tag{2}$$

其中, k 是与设备结构有关的常数; 传输过程中能耗, 如式 (3) 所示:

$$e_{传} = pt_{传} \tag{3}$$

整个过程的时间往往是被离散化为相等长度的时间槽τ, 任务在卸载过程中的时延主要是从设备到边缘节点的传输时间和边缘节点的处理时间, 而为了简单起见, 通常假设回传时间很小可忽略[23], 传输时间通常是由设备与边缘节点之间的无线信道传输速率 r 和任务大小 b 决定, 如式 (4) 所示:

$$t_{传} = \frac{b}{r} \tag{4}$$

同理, 处理时间则和边缘节点处理能力 f、任务大小 b 有关, 如式 (5) 所示:

$$t_{处} = \frac{b}{f} \tag{5}$$

可以发现, 要想降低整体时延, 由公式可知可以提高设备的 f, 公式又限制我们不能无限提高, 所以需要我们根据不同的任务需要, 找到时延和能耗之间的平衡点. 不同文献研究方法和重点不同, 但基本都是围绕以上变量进行组合优化.

4 总结

计算卸载可以实现将移动设备上的计算任务卸载到计算资源较丰富的边缘服务器, 从而提高了计算效率, 减缓了移动设备负担. 虽然计算卸载计算已经取得了较大发展, 但仍存在一些不足.

4.1 算法

关于计算卸载的算法策略, 现有的的特点第一个是种类多, 包括针对单用户单服务器的 EFO 算法[10]、将计算卸载表示为混合整数非线性计算过程的软件定义网络的思想[27]、马尔科夫决策过程 (MDP)[34]、博弈论等; 第二个特点就是效果较好, 基于这些算法和思想进行处理的效果还是比较好的, 在降低时延、优化能耗以及两者的综合衡量方面都表现不错, 但随着边缘计算规模的扩大以及场景的复杂化, 传统的算法或者某一种算法已经难以适应, 这也是最后一个特点, 即传统算法的适应性或者可扩展性差. 同时, 对于不同的边缘计算应用, 合适的卸载模型很关键, 比如类似社交网络等非实时性应用应使用软期限模型, 而对无人驾驶、增强现实等对时延要求很高的应用则需要有时延约束的优化模型.

计算卸载, 实质上就是对无线通信和移动计算的有效利用[41], 目前对计算卸载的研究都是通过模拟仿真进行验证, 设置的实验条件和真实的场景存在很大不同. 前面提到的文献以及分析的计算模型、通信模型等, 一定程度上都是基于假设的, 仿真实验简化甚至忽略了实际因素, 比如时变性, 无线信道的通信质量, 边缘服务器同时处理多移动终端设备的计算卸载时用户设备间的影响[42]; 有的考虑了但一般给出的也是恒定值, 存在很大的局限性.

随着机器学习、深度强化学习的不断发展, 可以考虑利用有关技术研究计算卸载问题[43,44]. 比如强化学习可以通过智能体与环境的交互形成决策, 设定适合的奖励函数完成训练, 可更适合时变的环境. 同时, 由于外部环境有着不确定性和高度复杂性, 使得对系统状态函数的考虑变得多维和复杂, 导致对状态空间的搜索变得困难, 算法收敛速度缓慢, 所以对算法的要求会更高. 基于此, 对高维状态空间的搜索和奖励函数的设置是利用强化学习理论研究计算卸载主要有两个研究点, 状态空间搜索, 主要是贪婪算法 (greedy algorithm), 是一种局部优化算法, 但并不适合高维的状态空间, 可以考虑蚁群算法和遗传算法[45]; 一个合适的奖励函数, 直接关系到最终结果能否准确体现出作者所提算法或者方案[46].

4.2 安全性

现有的策略缺少对安全性的考虑, 即如何保障边缘服务器及通信信道的安全性, 防止数据泄露窃取等问题, 值得下一步研究.

一方面, 各移动设备和边缘服务器的异构性使得传统的信任和认证机制不适用[47], 需要一个统一的信任和认证机制来评估边缘服务器的可靠性; 通信技术和网络管理机制带来新的安全威胁, 是因为支持边缘计算的各通信协议有各自的信任域, 可以使用加密属性来解决[48]; 另一方面, 边缘服务器的部署导致单节点防御能力薄弱, 单点的影响可能会影响整体系统. 针对以上的安全性问题, 一方面可以考虑边缘计算的架构设计, 另一方面边缘计算可以联合各种安全性解决方案对边缘计算安全性解决方案进行设计, 达到互补的效果.

参考文献

1 Shi WS, Cao J, Zhang Q, *et al*. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 2016, 3(5): 637–646. [doi: 10.1109/JIOT.2016.2579198]

2 吕洁娜, 张家波, 张祖凡, 等. 移动边缘计算卸载策略综述. 小型微型计算机系统, 2020, 41(9): 1866–1877. [doi: 10.3969/j.issn.1000-1220.2020.09.013]

3 边缘计算产业联盟 ECC. 基于边缘计算的铸管感算控一体化智能视觉解决方案. https://www.sohu.com/a/440060489_100019702. (2020-12-23).

4 中国移动研究院. 视频边缘计算白皮书. http://www.bianyuanyun.com/archives/1644. (2020-11-20).

5 Mao YY, You CS, Zhang J, *et al*. A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322–2358.

6 Yang C, Liu Y, Chen X, *et al*. Efficient mobility-aware task

offloading for vehicular edge computing networks. IEEE Access, 2019, 7: 26652–26664. [doi: 10.1109/ACCESS.2019.2900530]

7 Mao YY, Zhang J, Letaief KB. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590–3605.

8 Yang S, Kwon Y, Cho Y, *et al*. Fast dynamic execution offloading for efficient mobile cloud computing. 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom). San Diego: IEEE, 2013. 20–28.

9 Wang W, Lan RN, Gu JX, *et al*. Edge caching at base stations with device-to-device offloading. IEEE Access, 2017, 5: 6399–6410. [doi: 10.1109/ACCESS.2017.2679198]

10 Zhao PT, Tian H, Qin C, *et al*. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access, 2017, 5: 2169–3536.

11 Zhang HL, Guo J, Yang LC, *et al*. Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC. 2017 IEEE Conference on Computer Communications Workshops (INFOCOM). Atlanta: IEEE, 2017. 115–120.

12 Shu C, Zhao ZW, Han YP, *et al*. Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach. IEEE Internet of Things Journal, 2020, 7(3): 1678–1689. [doi: 10.1109/JIOT.2019.2943373]

13 Namboodiri V, Ghose T. To cloud or not to cloud: A mobile device perspective on energy consumption of applications. 2012 IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM). San Francisco: IEEE, 2012. 1–9.

14 Ali Z, Jiao L, Baker T, *et al*. A deep learning approach for energy efficient computational offloading in mobile edge computing. IEEE Access, 2019, 7: 149623–149633. [doi: 10.1109/ACCESS.2019.2947053]

15 Chen X, Jiao L, Li WZ, *et al*. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking, 2016, 24(5): 2795–2808. [doi: 10.1109/TNET.2015.2487344]

16 Lin YD, Chu ETH, Lai YC, *et al*. Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds. IEEE Systems Journal, 2015, 9(2): 393–405. [doi: 10.1109/JSYST.2013.2289556]

17 Mahmoodi SE, Uma RN, Subbalakshmi KP. Optimal joint scheduling and cloud offloading for mobile applications. IEEE Transactions on Cloud Computing, 2019, 7(2): 301–313. [doi: 10.1109/TCC.2016.2560808]

18 Zhang L, Fu D, Liu JC, *et al*. On energy-efficient offloading in mobile cloud for real-time video applications. IEEE Transactions on Circuits and Systems for Video Technology, 2017, 27(1): 170–181. [doi: 10.1109/TCSVT.2016.2539690]

19 Bhattacharya A, De P. A survey of adaptation techniques in computation offloading. Journal of Network and Computer Applications, 2017, 78: 97–115. [doi: 10.1016/j.jnca.2016.10.023]

20 Liu J, Mao YY, Zhang J, *et al*. Delay-optimal computation task scheduling for mobile-edge computing systems. 2016 IEEE International Symposium on Information Theory (ISIT). Barcelona: IEEE, 2016. 1451–1455.

21 Labidi W, Sarkiss M, Kamoun M. Energy-optimal resource scheduling and computation offloading in small cell networks. 2015 22nd International Conference on Telecommunications (ICT). Sydney: IEEE, 2015. 313–318. [doi: 10.1109/ICT.2015.7124703]

22 Zhang K, Mao YM, Leng SP, *et al*. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access, 2016, 4: 5896–5907. [doi: 10.1109/ACCESS.2016.2597169]

23 Han D, Chen W, Fang YG. Joint channel and queue aware scheduling for latency sensitive mobile edge computing with power constraints. IEEE Transactions on Wireless Communications, 2020, 19(6): 3938–3951. [doi: 10.1109/TWC.2020.2979136]

24 Xu XD, Liu JX, Tao XF. Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation. IEEE Access, 2017, 5: 16406–16415. [doi: 10.1109/ACCESS.2017.2739343]

25 Meng H, Chao DC, Guo QY, *et al*. Delay-sensitive task scheduling with deep reinforcement learning in mobile-edge computing systems. Proceedings of 2019 3rd International Conference on Machine Vision and Information Technology (CMVIT 2019). Guangzhou: IOP Publishing, 2019. 484–491.

26 Ulukus S, Yener A, Erkip E, *et al*. Energy harvesting wireless communications: A review of recent advances. IEEE Journal on Selected Areas in Communications, 2015, 33(3): 360–381. [doi: 10.1109/JSAC.2015.2391531]

27 Chen M, Hao YX. Task offloading for mobile edge computing in software defined ultra-dense network. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 587–597. [doi: 10.1109/JSAC.2018.2815360]

28 Jeong HJ, Jeong I, Lee HJ, *et al*. Computation offloading for machine learning web APPs in the edge server environment. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). Vienna: IEEE, 2018.

1492–1499.
29 Shu C, Zhao ZW, Han YP, *et al*. Dependency-aware and latency-optimal computation offloading for multi-user edge computing networks. 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). Boston: IEEE, 2019. 1–9.
30 Han YP, Zhao ZW, Mo JW, *et al*. Efficient task offloading with dependency guarantees in ultra-dense edge networks. 2019 IEEE Global Communications Conference (GLOBECOM). Waikoloa: IEEE, 2019. 1–6.
31 Pan SL, Zhang ZY, Zhang ZW, *et al*. Dependency-aware computation offloading in mobile edge computing: A reinforcement learning approach. IEEE Access, 2019, 7: 134742–134753. [doi: 10.1109/ACCESS.2019.2942052]
32 Cui YP, Liang YJ, Wang RY. Resource allocation algorithm with multi-platform intelligent offloading in D2D-enabled vehicular networks. IEEE Access, 2019, 7: 21246–21253. [doi: 10.1109/ACCESS.2018.2882000]
33 Abbas N, Zhang Y, Taherkordi A, *et al*. Mobile edge computing: A survey. IEEE Internet of Things Journal, 2018, 5(1): 450–465. [doi: 10.1109/JIOT.2017.2750180]
34 Kamoun M, Labidi W, Sarkiss M. Joint resource allocation and offloading strategies in cloud enabled cellular networks. 2015 IEEE International Conference on Communications (ICC). London: IEEE, 2015. 5529–5534.
35 Huang D, Wang P, Niyato D. A dynamic offloading algorithm for mobile computing. IEEE Transactions on Wireless Communications, 2012, 11(6): 1991–1995. [doi: 10.1109/TWC.2012.041912.110912]
36 Hu HT, Luo XR. Research on cloud task scheduling based on load balancing ant colony optimization. Proceedings of 2018 International Conference on Computer, Communication and Network Technology (CCNT 2018 Volume 1). Tongxiang: DEStech Publications, 2018. 68–72.
37 Barbarossa S, Sardellitti S, Lorenzo PD. Joint allocation of computation and communication resources in multiuser mobile cloud computing. 2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC). Darmstadt: IEEE, 2013. 26–30.
38 Wei F, Chen SX, Zou WX. A greedy algorithm for task offloading in mobile edge computing system. China Communications, 2018, 15(11): 149–157. [doi: 10.1109/CC.2018.8543056]
39 Rajput SS, Kushwah VS. A genetic based improved load balanced min-min task scheduling algorithm for load balancing in cloud computing. 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN). Tehri: IEEE, 2016. 677–681.
40 张文柱, 曹琲琲, 余静华. 移动边缘计算中一种多用户计算卸载方法. 西安电子科技大学报, 2020, 47(6): 131–138.
41 Xu DL, Li T, Li Y, *et al*. Edge intelligence: Architectures, challenges, and applications. arXiv: 2003.12172, 2020.
42 Chen Z, Wang XD. Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach. EURASIP Journal on Wireless Communications and Networking, 2020, 2020: 188. [doi: 10.1186/s13638-020-01801-6]
43 Li J, Gao H, Lv TJ, *et al*. Deep reinforcement learning based computation offloading and resource allocation for MEC. 2018 IEEE Wireless Communications and Networking Conference (WCNC). Barcelona: IEEE, 2018. 1–6.
44 Huang L, Bi SZ, Zhang YJA. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Transactions on Mobile Computing, 2020, 19(11): 2581–2593. [doi: 10.1109/TMC.2019.2928811]
45 Wang TT, Liu ZB, Chen Y, *et al*. Load balancing task scheduling based on genetic algorithm in cloud computing. 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. Dalian: IEEE, 2014. 146–152.
46 Shan NL, Cui XL, Gao ZQ. “DRL + FL”: An intelligent resource allocation model based on deep reinforcement learning for Mobile Edge Computing. Computer Communications, 2020, 160: 14–24. [doi: 10.1016/j.comcom.2020.05.037]
47 Roman R, Lopez J, Mambo M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. Future Generation Computer Systems, 2018, 78: 680–698.
48 Huang XY, Xiang Y, Bertino E, *et al*. Robust multi-factor authentication for fragile communications. IEEE Transactions on Dependable and Secure Computing, 2014, 11(6): 568–581. [doi: 10.1109/TDSC.2013.2297110]