

第3章 边缘计算中的计算卸载

在传统的中心计算时代,人们利用中心云强大的计算和存储能力,享受云计算带来的福祉。随着日益紧张的网络通信资源,加之万物互联的背景,资源从中心下沉到了边缘。正如共享经济的迅猛发展,能否利用网络边缘设备之间的互联关系,使边缘小节点代行中心大节点之职能,是边缘计算模式中至关重要的“边缘思维”。由此衍生出了边缘计算中的关键技术——计算卸载。

在万物互联的背景下,虽然移动设备的处理能力日益强大,但是受制于芯片尺寸和终端尺寸,移动设备的处理能力终归有限。此外,终端尺寸往往直接影响电池容量,进而影响终端设备可以用来处理任务的能量的多少。因此一些计算密集型任务(对算力要求较高)或时间敏感型任务(对处理时间要求较高)无法在本地或传统云计算模式(包括云计算、移动云计算等)下完成。借用边缘思维,试图在周围边缘节点计算能力的协助下处理本地任务,这一过程称为计算卸载。

3.1 什么是计算卸载

计算卸载的提出应追溯到移动云计算概念的提出。在移动云计算模式中,移动用户可以将计算密集型任务通过无线网络的方式传输到远端云服务器执行,达到缓解计算和存储限制、延长设备电池寿命的目的,这个过程称为移动云计算中的计算卸载。值得注意的是,移动云计算这种模式本质上是云计算应用场景的扩展,而不是云计算本身概念的扩展。因此虽然移动云计算为云计算技术提供了更多的应用场景,但是本质上并未解决中心化云计算模式存在的问题。

与之相对应的是边缘计算,在万物互联的背景下边缘计算模式产生了,这是一种与中心化云计算完全不同的分布式计算模式。在边缘计算中,计算卸载指的是用户终端将任务传输到邻近的一个或多个边缘节点(远端云节点也可以加入),在边缘节点的帮助下完成任务处理,以解决设备在资源存储、计算性能以及能效等方面的不足。当然也有文献称之为计算迁移,本章统一称计算卸载。相比于移动终端将计算卸载到云服务器可能导致的不可预测时延、传输距离远等问题,边缘计算能够更快速、高效地为移动终端提供计算服务,同时缓解核心网络的压力。

需要注意的是,在传统移动云计算场景中,更希望利用云来解决计算密集型任务。因为移动设备的处理能力不足,相当于移动设备的“硬伤”,只能通过求助强大的云中心来解决问题。而在边缘计算场景中,更希望利用边缘节点来解决延迟敏感型任务。在边缘计算场景中,虽然边缘节点具有异构性,但同时也表现出了同质性,即边缘节点的处理能力和能量都是有限的。因此在边缘计算模式中,卸载策略应该是以减少网络传输数据量为目的的,也就是主要针对延迟敏感型任务的。表 3-1 列出了传统移动云计算场景和边缘计算场景下计算

卸载的异同。

表 3-1 移动云计算与边缘计算卸载对比

类 目	移动边缘计算	移动云计算
计算模型	分布式	集中式
服务器硬件	小型,中等性能	大型,高性能
与用户距离	近	远
连接方式	无线连接	专线连接
隐私保护	高	低
时延	低	高
核心思想	边缘化	中心化
计算资源	有限	丰富
存储容量	有限	丰富
应用	延迟敏感型	计算敏感型

计算卸载定义中的动词“传输”暗含了多重意义,比如:“是否需要传输”,当前任务是否需要利用卸载技术提升性能(可能会带来额外的传输延迟);“传什么”,传任务还是传任务所需数据,传部分任务还是传全部任务;“什么时候传”,在任务的什么阶段进行传输能最大化效益;“怎么传”,具体采用哪种计算卸载的方法等。定义中的“一个或多个边缘节点”暗含了计算卸载的另一个选择特性——“传给谁”,任务交由哪些边缘节点代为执行。以上问题是计算卸载中的重点研究问题,可以统称为卸载决策问题。

3.2 计算卸载的分类

对计算卸载的分类有助于决定在不同情境下如何快速制定卸载策略,根据分类标准的不同,存在多种分类方法,大体上有以下分类标准,如图 3-1 所示。

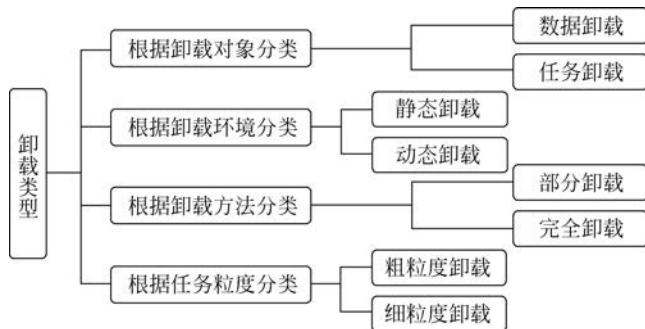


图 3-1 计算卸载的分类

3.2.1 根据卸载对象分类

根据卸载对象可以将计算卸载分为数据卸载和任务卸载。数据卸载是指将处理某个任务的所需数据(图片、文本等信息)卸载到边缘节点。任务卸载是指直接将某个任务打包卸载到边缘节点。

以图像处理中的人脸识别为例,若将人脸图片卸载到边缘节点,边缘节点识别后返回结

果则为数据卸载；若将整个人脸识别程序打包卸载给边缘节点执行则为任务卸载。一般来说数据卸载适用于相同的任务架构和模型，只需要调整相关参数即可获得结果的任务（如相同的神经网络结构，只需要输入不同参数等）。而任务卸载适用于边缘节点和用户设备具有完全相同的任务运行环境的情况（如同样的操作系统、同样的配置环境等）。

3.2.2 根据卸载环境分类

根据卸载环境将计算卸载分为静态卸载和动态卸载。静态卸载是指按照某种配置好的规则和策略进行计算卸载。动态卸载是指在卸载过程中根据实际影响因素的变化动态规划卸载过程。

所谓实际影响因素主要是指“动态性”，一方面在边缘计算模式下，用户和边缘节点都有可能处于运动状态——具象的“动态”；另一方面在计算卸载过程中有可能面临任务的随机性、网络资源波动等不稳定因素——抽象的“动态”。静态卸载和动态卸载中的卸载策略有时也分别被称为离线卸载策略和在线卸载策略。

3.2.3 根据卸载方法分类

将计算卸载按照卸载方法可以分为部分卸载和完全卸载。部分卸载是指将计算过程的一部分交由边缘节点执行，另一部分本地执行。完全卸载是指将整个计算过程交由边缘节点处理。

3.2.4 根据任务粒度分类

将计算卸载按照任务粒度可以分为粗粒度卸载和细粒度卸载。粗粒度卸载是指将整个移动终端应用作为卸载对象，并未根据功能再将其划分为多个子任务。在细粒度卸载中，将一个移动应用划分为多个具有数据依赖关系的子任务，因为划分后的子任务所需的计算复杂度和数据传输量更少，因此可以将部分或者所有任务卸载到多个远程服务器上进行处理，以此节省计算时间和传输时间。

3.2.5 其他分类标准

可以根据应用情景对计算卸载进行分类，如能量约束下的计算卸载、可分负载应用的计算卸载、基于边缘云选择的计算卸载、可充电环境下的计算卸载等。也可以根据卸载决策参数将计算卸载分为能耗卸载、时延卸载等。另外，也可结合网络拓扑关系将计算卸载分为一对一卸载、一对多卸载、多对一卸载和多对多卸载等。

值得注意的是分类方法并不是绝对的，往往根据不同的分类标准可以得到不同的分类结果。仍以图像处理中的人脸识别为例，只将人脸图片卸载到边缘节点，边缘节点识别后返回结果称为数据卸载，若将人脸识别作为整个任务的一部分，则可以称之为部分卸载；若将整个人脸识别程序打包给边缘节点执行为任务卸载，也可称之为完全卸载。某一个卸载决策可能是多种分类的组合，如在可充电环境下根据通信资源变化和用户设备剩余能量将部分任务交由边缘节点处理的过程就是可充电环境下的部分动态卸载。掌握计算卸载的分类方法有助于快速做出通行的卸载决策，但不宜拘泥于分类本身。

3.3 计算卸载的步骤

计算卸载的步骤本质上来说是对提出的卸载决策问题的主要处理流程,如图 3-2 所示。

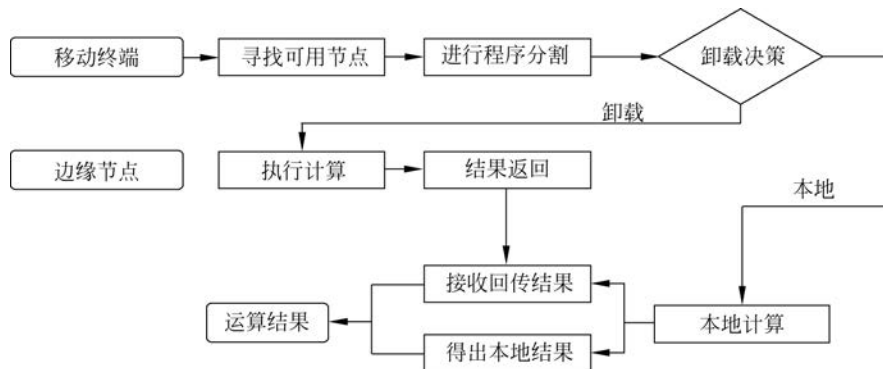


图 3-2 计算卸载的步骤

计算卸载的主要步骤包括以下六方面。

(1) 节点发现。根据需求在边缘网络中寻找可用节点。注意此处可用节点并不局限于边缘节点,也包括和边缘网络连接的中心节点(在端边云协同的计算模式中,中心云节点用以处理计算密集型任务)。

(2) 程序分割。主要针对部分卸载情况下,对任务流程进行分割,应尽量确保分割后各分割部分的自包容性,同时也应注意任务的拓扑结构。

(3) 卸载决策。卸载决策问题是计算卸载的核心问题,主要包括卸载与否、卸载什么、何时卸载、卸载去哪儿、怎样卸载等问题。

(4) 程序传输。主要考虑卸载过程中的网络通信状况,有诸多文献针对网络通信模型差异、网络通信波动等问题对程序传输部分进行优化处理。

(5) 执行计算。该过程主要发生在边缘节点,用户只需等待即可。值得注意的是,在考虑系统性能时往往需要考虑边缘节点的能量损耗,但若只从用户角度出发,这段时间内的能量损耗只有等待时的待机损耗,而这部分往往可以忽略不计。

(6) 结果回传。计算结果回传是计算卸载的最后一个步骤。不论是数据卸载还是任务卸载、部分卸载还是完全卸载,往往回传的结果与上传的数据或任务相比数据量都非常小。以人脸识别为例,上传的可能是一张照片或整个人脸识别程序,而回传的一般是判断性结果,如图片中的人是谁。因此结果回传产生的延迟和能耗在大多数文献中都忽略不计。

3.4 影响卸载决策的因素

3.2 节介绍了卸载决策问题是边缘计算卸载中的关键问题,3.3 节针对计算决策的步骤展开了讨论,这一节重点关注在边缘计算的卸载决策过程中,由什么决定卸载决策,换言之就是在卸载决策过程中谁起主导作用。表 3-2 从卸载决策的参与实体的角度总结了常见的决策因素,针对边缘计算卸载的多数文献也是针对其中的一个或多个因素展开研究的。

表 3-2 计算卸载决策因素

参 与 实 体	决 策 因 素
用户及用户终端	用户偏好(时间、能耗、开支等)；用户信息感知能力；本地设备能耗剩余量；本地设备处理能力；竞争设备数等
任务	数据量；计算量；私密性；类型；是否可分割等
通信条件	网络环境：4G/5G、Wi-Fi 等；网络带宽容量等
边缘云	处理能力(CPU/GPU 类型、速度等)；内存；存储空间；负载情况；相对距离；相对动静状态等

通过对文献的梳理可以发现,在边缘计算卸载的相关研究中,主要考虑的还是以下因素。

(1) 能耗。对于移动终端和边缘节点来说最重要的约束之一就是能耗,因为大部分智能终端都无法靠自身补充能量,因此大多数研究中都会优先考虑能耗约束或以能量有效性为目标。

(2) 时间。计算所需的卸载时间很大程度上决定了完成该计算或任务的整体时间,作为卸载决策的重要参数,时间(或说延迟)直接影响了用户的卸载体验和卸载效率。

(3) 可用资源。可用资源主要指的是边缘节点能够提供的资源。在新兴的边缘缓存领域(边缘计算和内容分发网络的交叉领域),可用资源也包括边缘节点能提供的存储空间等。

用户的目的、行为及其自身的影响因素(如用户偏好、用户信息感知能力)等也在很大程度上影响了卸载决策,而针对此方面的研究多将人的行为简化或不予考虑,这是今后的研究过程中值得关注的部分。

3.5 基于能量约束的计算卸载

3.4 节介绍了能耗作为边缘计算卸载决策中的重要决策因素,对卸载过程有重要影响。这一节将以实例探寻在能量约束的应用情景中如何刻画计算卸载问题。

3.5.1 能量约束的必要性

在计算卸载问题中,可以通过两方面考虑能耗:第一,可以直接以“最小化能耗”“最大化能量有效性”等作为目标;第二,可以将能耗作为约束条件之一对计算卸载问题进行限制,虽不直接以能耗为目标,但是问题刻画受到能耗条件的限制。

下面从用户、服务和角色三个层面来解释能量约束的必要性,也就是在计算卸载问题中考虑能耗的必要性。

(1) 用户层面。在大多数情况下,用户进行计算卸载的目的之一就是节省本地设备的能量消耗。

(2) 服务层面。在边缘计算模式下,大部分提供服务的边缘节点也具有“轻量化”的特点,也就是说提供计算服务的服务节点和提供通信服务的通信节点往往也是能量受限的。

(3) 角色层面。在万物互联时代,边缘节点的角色身份是多元的。在这一时刻提供服

务的节点可能自身也是消费节点,还需要剩余能量维持自身基础运行和计算自身任务。

3.5.2 能量约束计算卸载问题的挑战和目标

在边缘计算环境中,某些服务节点既提供计算服务又提供通信服务,需要新模型来刻画在能量约束下的两种资源(计算和通信)管理之间的相互作用和相互依赖,这称为交互问题。从系统稳定性的角度来看,在实际边缘计算环境中,不论是用户节点还是计算节点往往都工作在随机环境中,对于计算节点来说不同的时空关系上都可能随机的有工作量到达,因此如何在随机环境中维持系统长期低能耗是至关重要的,可称为稳定问题。从用户角度出发,由于边缘计算环境中的节点往往代表的是独立个体,因此如何最大化独立个体所提供的有限资源的价值是至关重要的,这被称为激励问题。

除此之外,在构建能量约束计算卸载问题时还会遇到许多实际或数学刻画问题,需要具体问题具体对待。针对这些问题研究能量约束计算卸载问题的目标包括但不限于在能量约束下最大限度地提高系统的长期性能、在能量约束下提升任务卸载率、增加可卸载设备数量、提出能量有效性高的卸载策略、研究边缘计算环境中关于能量价值的激励问题等。

3.5.3 给定能量约束下最小化系统长期延迟的计算卸载决策

在此实例中,以能量约束下最小化系统长期延迟为例,剖析如何构建一个计算卸载问题。

1. 网络模型

本例模型如图 3-3 所示,在某一个或某几个小区构成的网络中,用户既可以选择本地执行任务,也可以选择将任务卸载到小区内的小型基站。对于小型基站,既可以本地处理任务,也可以将收到的任务通过局域网传输给别的小型基站进行处理。

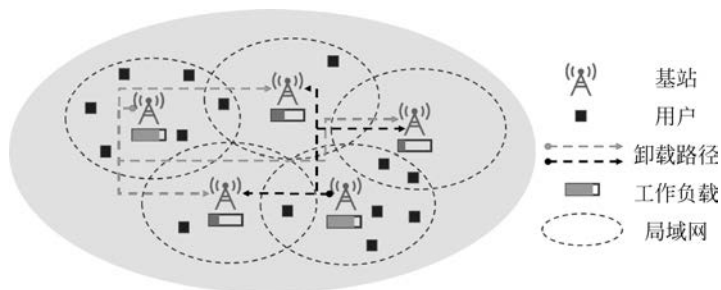


图 3-3 能量约束网络模型示意图

2. 任务传输模型

(1) 传输能耗。传输过程既包括用户到小基站之间的传输,也包括小基站之间的有线传输。由于小基站的能量消耗主要是由于下行链路传输造成的,因此考虑下行链路无线传输的能量消耗。小基站 i 在时间 t 内的传输能耗可以表示为

$$E_i^t = \sum_{m \in \mathcal{M}_i} \frac{P_i^d \omega_m^t}{r_{im}^{d,t}} \quad (3-1)$$

式中: P_i^d 为小基站 i 的下行链路传输功率; ω_m^t 为下行链路的传输流量; $r_{im}^{d,t}$ 为根据香农

公式计算的下行信道速率,其公式可表示为

$$r_{im}^{d,t} = W \log \left(1 + \frac{P_i^d H_{im}^t}{\sigma^2} \right) \quad (3-2)$$

其中, H_{im}^t 表示下行信道状态矩阵, σ^2 表示下行信道噪声, W 是下行信道带宽。

(2) 传输延迟。传输延迟主要是由于用户和小基站之间的任务传输而造成的,即用户与小基站上行链路通信期间所产生的延迟,因此小基站 i 在时间 t 内的传输延迟可以表示为

$$D_i^{u,t} = \sum_{m \in \mathcal{M}_i} \frac{s \pi_m^t}{r_{im}^{u,t}} \quad (3-3)$$

式中: s 为用户上传数据大小; π_m^t 为上行任务到达率; $r_{im}^{u,t}$ 为根据香农公式计算的上行信道速率。

3. 计算卸载模型

计算卸载模型如图 3-4 所示,可以看出当边缘节点收到用户的卸载任务时,就可以选择进入队列等待处理,也可以通过局域网进行调度。若直接在边缘节点计算则会产生计算延迟,局域网调度过程会产生拥塞延迟。图中的 ϕ_i 是任务到达率,而 β_{ij} 是一个取值区间为 $(0, 1]$ 的连续决策变量,决定了将多少任务分配出去。因此将计算卸载延迟表示为计算延迟和拥塞延迟的和,即卸载延迟 = 计算延迟 + 拥塞延迟。

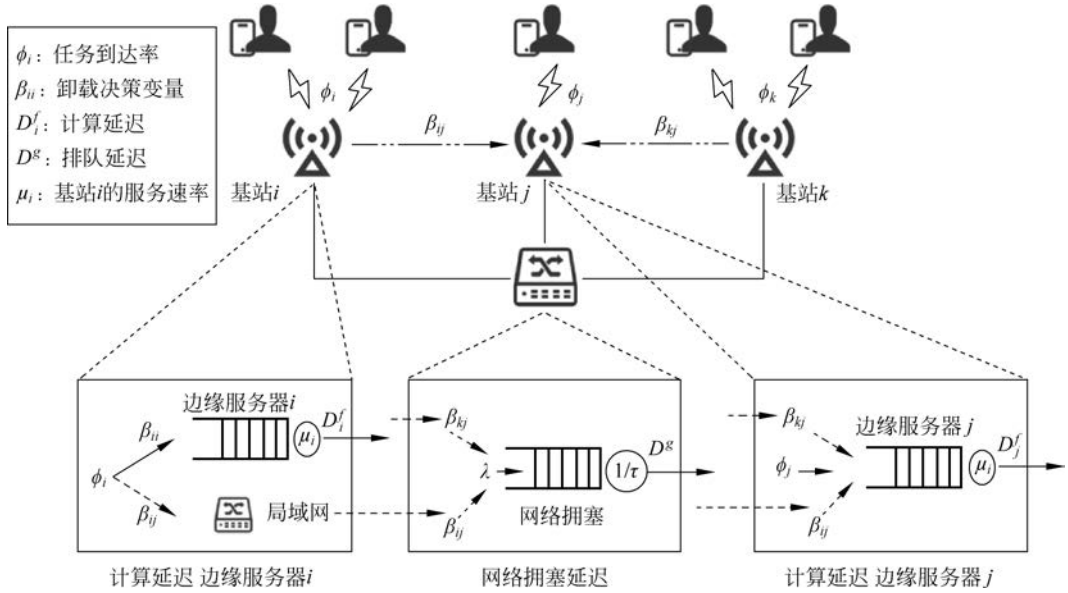


图 3-4 任务卸载模型示意图

计算延迟的刻画参考了排队论中的 M/M/1 模型,简而言之就是,时间等于服务速率与任务到达速率差的倒数。因此,小基站 i 的计算延迟可以表示为

$$D_i^{f,t}(\beta^t) = \frac{1}{\mu_i - \omega_i^t(\beta^t)} \quad (3-4)$$

式中: μ_i 为小基站 i 的服务速率; $\omega_i^t(\beta^t)$ 为到来的总任务负载; β^t 为上述连续决策变量的集合。

拥塞延迟取决于局域网内部小基站之间的通信量,同样以 $M/M/1$ 刻画。

$$D^{g,t}(\beta^t) = \frac{\tau}{1 - \tau\lambda^t(\beta^t)}, \quad \lambda^t < \frac{1}{\tau} \quad (3-5)$$

式中: τ 为局域网无拥塞情况下发送和接收 s 比特数据所需的时间; $\lambda^t(\beta^t)$ 为小基站之间的传输数据量。

与传输能耗不同的是,小基站的计算能耗可以用只与计算任务量相关的线性模型来表示

$$E_i^{c,t}(\beta^t) = \kappa \cdot \omega_i^t(\beta^t) \quad (3-6)$$

式中: κ 为处理单位任务量时的能耗。

4. 问题建模

以上分析了在整个计算卸载过程中的延迟和能耗的构成部分,系统的总延迟和总能耗即为相关部分的线性和。具体来说,系统总延迟定义为传输延迟、计算延迟、拥塞延迟之和,系统总能耗定义为传输能耗与计算能耗之和,可以分别表达为式(3-7)与式(3-8)。

$$D_i^t(\beta^t) = \sum_{j \in \mathcal{N}} \beta_{ij}^t D_j^{f,t}(\beta^t) + \lambda_i^t D^{g,t}(\beta^t) + D_i^{u,t} \quad (3-7)$$

$$E_i^t(\beta^t) = E_i^{tx,t} + E_i^{c,t}(\beta^t) = E_i^{tx,t} + \kappa \cdot \omega_i^t(\beta^t) \quad (3-8)$$

在本实例中,以在给定能量约束下最小化系统长期延迟为优化目标,刻画为优化问题 P3_1。

$$\min_{\beta^0, \dots, \beta^{T-1}} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^N E\{D_i^t(\beta^t)\} \quad (3-9)$$

$$\text{s. t. C1: } \frac{1}{T} \sum_{t=0}^{T-1} E\{E_i^t(\beta^t)\} \leq \bar{E}_i, \quad \forall i \in \mathcal{N}$$

$$\text{C2: } E_i^t(\beta^t) \leq E_{\max}, \quad \forall i \in \mathcal{N}, \forall t$$

$$\text{C3: } D_i^t(\beta^t) \leq D_{\max}, \quad \forall i \in \mathcal{N}, \forall t$$

$$\text{C4: } \beta^t \in B^t, \forall t$$

可以看出,虽然整个优化问题是以最小化延迟为目标的,但是约束条件中的前两个都与能耗紧密相关。其中约束条件 C1 为基站由于某种激励机制而具有一个预先确定的长期能耗约束, C2 为每个时隙内基站的能耗约束。这就是典型的将能耗作为约束条件之一的计算卸载问题。

求得上述问题的最优解需要知道每个时隙的任务到达情况(离线信息),而这几乎是不可能的。此外,长期能量约束和跨时隙的卸载决策存在耦合关系,即当前时隙消耗了较多的能量会导致将来可用能量的减少。因此为了解决上述问题,需要一种在线优化方法,在无法预见未来的情况下有效地执行任务卸载决策。

3.6 基于边缘云选择的计算卸载

在实际边缘计算环境中,一个用户周围会存在多个可用的边缘服务节点,势必会引起边缘云选择。然而多个边缘云之间存在负载均衡问题,因此多服务器边缘计算系统进行计算卸载策略优化面临更大的挑战。多边缘云计算系统有着显著的优越性,具体体现在以下几

方面。

(1) 在为大量卸载用户提供服务时,可以通过云间协同减轻某节点负担。

(2) 每个用户可以选择将其任务卸载到更优上行链路信道条件的边缘服务器,从而节省传输能量消耗。

(3) 协作资源分配可以帮助减轻用户之间的干扰和资源竞争的影响,当多个用户同时卸载其任务时,可以提高边缘云卸载收益。

3.6.1 边缘云选择的必要性及待解决问题

如图 3-5 所示,边缘计算中的边缘云选择问题主要从以下两点出发。

(1) 移动性,在边缘计算环境中,不论用户节点还是服务节点都可能处于移动环境,因此需要根据地理位置合理选择适合的边缘服务节点。

(2) 竞争性,由于边缘计算环境中不论是计算资源还是通信资源都不及传统中心式计算充足,因此需要把有限的资源高效地分配到最需要的地方,即“好钢用在刀刃上”。

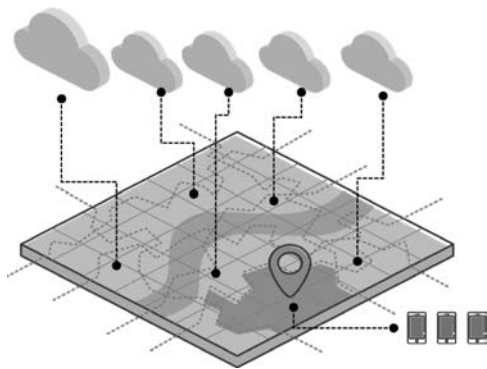


图 3-5 基于位置的边缘云选择示意图

在边缘云选择中,需要确定智能移动设备如何在环境约束下选择最佳的边缘云并将任务卸载到所选的边缘云(常见的环境约束有移动性约束和接入方式约束等)。此外,还需考虑如何合理分配边缘云资源,使边缘节点和用户的系统性能最佳(如总任务执行成本最低)。

3.6.2 环境约束下的多小区机制中的边缘云选择

考虑一个如图 3-6 所示的多小区超密集网络,其中每个小基站都配备一个边缘计算服务器,以为移动用户提供计算卸载服务,最大限度地提高用户的卸载收益。

1. 小区选择约束

在该实例中,为了保证小区通信资源和计算资源最大限度地被利用,规定一个用户只能将其任务卸载到一个边缘云上(经过选择后),称之为选择约束,可以具体表示为

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{N}} x_{us}^j \leq 1, \quad \forall u \in \mathcal{U} \quad (3-10)$$

式中: x_{us}^j 为二元决策变量,表示用户 u 是否通过子信道 j 将任务上传到基站 s 。

由于选择了正交频分复用接入方式,小区内的干扰得到缓解,但是小区间的干扰仍然存在,因此用户 u 和基站 s 在子信道 j 上的信号与干扰加噪声比可表示为



图 3-6 多小区机制中的边缘云选择

$$\gamma_{us}^j = \frac{p_u h_{us}^j}{\sum_{k \in \mathcal{U} \cup \mathcal{S}} x_{ks}^j p_k h_{ks}^j + \sigma^2}, \quad \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N} \quad (3-11)$$

数据传输速率可表示为

$$R_{us} = W \log_2(1 + \gamma_{us}) \quad (3-12)$$

2. 延迟和能耗

在本实例中,卸载延迟主要取决于传输延迟和计算延迟,二者分别表示为

$$t_{\text{up}}^u = \sum_{s \in \mathcal{S}} \frac{x_{us} d_u}{R_{us}}, \quad \forall u \in \mathcal{U} \quad (3-13)$$

$$t_{\text{exe}}^u = \sum_{s \in \mathcal{S}} \frac{x_{us} c_u}{f_{us}}, \quad \forall u \in \mathcal{U} \quad (3-14)$$

式中: t_{up}^u 为传输时延; t_{exe}^u 为计算延迟; d_u 为传输数据量; c_u 为任务工作量; f_{us} 为基站 s 分配给用户 u 的处理能力。考虑到卸载延迟等于传输延迟加上计算延迟,通过合并同类项可简化为

$$t_u = t_{\text{up}}^u + t_{\text{exe}}^u = \sum_{s \in \mathcal{S}} x_{us} \left(\frac{d_u}{R_{us}} + \frac{c_u}{f_{us}} \right), \quad \forall u \in \mathcal{U} \quad (3-15)$$

由于计算过程在边缘计算节点而不是本地用户设备,因而考虑用户的能耗时计算能耗可以省略,但是若考虑整个边缘计算系统的能耗则该项往往不可省略。在该例中,卸载能耗即为传输能耗,且采用了最为常见的能耗定义为时间乘以功率的模式。

$$E_u = p_u t_{\text{up}}^u = p_u d_u \sum_{s \in \mathcal{S}} \frac{x_{us}}{R_{us}}, \quad \forall u \in \mathcal{U} \quad (3-16)$$

3. 用户卸载增益

既然以用户为考虑对象,用户进行计算卸载一定是“有利可图”的,因此该例中以进行计算卸载带来的能耗和时间减少量的加权和作为评价用户卸载增益 J_u 的标准。

$$J_u = \left(\beta_u^t \frac{t_u^l - t_u}{t_u^l} + \beta_u^e \frac{E_u^l - E_u}{E_u^l} \right) \sum_{s \in \mathcal{S}} x_{us}, \quad \forall u \in \mathcal{U} \quad (3-17)$$

式中: t_u^l 和 E_u^l 分别为本地计算延迟和本地计算能耗; β_u^t 和 β_u^e 分别是能量和时间的权重值,可以代表不同的用户偏好。

4. 问题建模

该实例中,边缘云选择问题可以被刻画为优化问题 P3_2。

$$\max_{X, P, F} J(X, P, F) \quad (3-18)$$

$$\begin{aligned}
& \text{s. t. } x_{us}^j \in \{0,1\}, \quad \forall u \in \mathcal{U}, s \in \mathcal{S}, j \in \mathcal{N} \\
& \text{C1: } \sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{N}} x_{us}^j \leq 1, \quad \forall u \in \mathcal{U} \\
& \text{C2: } \sum_{u \in \mathcal{U}} x_{us}^j \leq 1, \quad \forall s \in \mathcal{S}, j \in \mathcal{N} \\
& \text{C3: } 0 < p_u \leq P_u, \quad \forall u \in \mathcal{U}_{\text{off}} \\
& \text{C4: } f_{us} > 0, \quad \forall u \in \mathcal{U}_s, s \in \mathcal{S} \\
& \text{C5: } \sum_{u \in \mathcal{U}} f_{us} \leq f_s, \quad \forall s \in \mathcal{S}
\end{aligned}$$

在给定卸载决策 X 、上行功率 P 和计算资源分配策略 F 的情况下,将系统的总效用定义为所有用户卸载效用的加权和,优化目标是最大化系统效用 $J(X, P, F)$ 。

前两个约束条件 C1、C2 表明每个任务可以在本地执行,也可以通过一个子信道卸载到一个边缘节点。约束条件 C3 表明任一边缘节点通过某一子信道只能服务单个用户。约束条件 C4 阐明了用户发射功率的可行区间。最后两个约束条件 C4、C5 说明每个边缘节点必须向与其关联的用户分配一个大于 0 的计算资源,并且分配给所有关联用户的总计算资源不得超过节点的计算能力。

以上优化问题是一个混合整数的非线性规划问题,通常需要指数级时间复杂度才能找到最优解。由于大量的变量与用户数量、边缘服务器数量和子带宽数量是线性关系,因此要对问题进行解耦,设计一种低复杂度的次优方案,在实现实用部署的同时保证算法的竞争力。

3.7 可分负载应用的计算迁移和卸载

在边缘计算场景中,可分负载应用是一类可以将数据或任务根据某种规则任意划分为多个部分,而每个部分可以在不同节点进行独立处理的应用。以常见的人脸识别和 AR 应用为例,如图 3-7 所示,其应用内部由多个独立子任务构成,使得实现部分卸载成为可能。

如图 3-8 所示,可分负载应用卸载的研究重点实质上是任务(或组件)之间的依赖关系。任务和组件的依赖关系一般可分为三种:顺序依赖、并行依赖和通用依赖。可以看出顺序依赖(串行)和并行依赖都较为简单,现实生活中多为复杂的通用依赖。

一般来说,任务(或组件)的依赖关系可由有向无环图(DAG)表示,其中:顶点代表任务(或组件);边代表依赖性;权重代表每个过程的输入和输出。所需资源量可以在顶点边标出。

3.7.1 数据划分型可分负载计算迁移

根据现有文献,对于这类应用程序,需要处理的数据量是预先知道的。可以利用并行性在本地端处理部分数据,其余部分在云端或边缘端并行处理,即所谓的并行依赖。

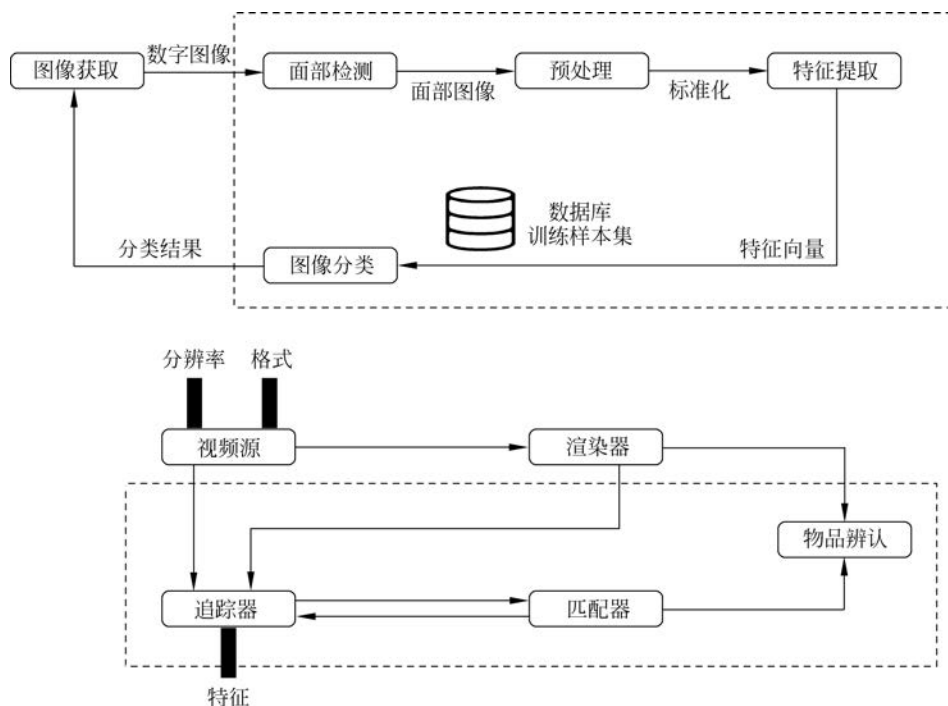


图 3-7 人脸识别应用和 AR 应用示意图

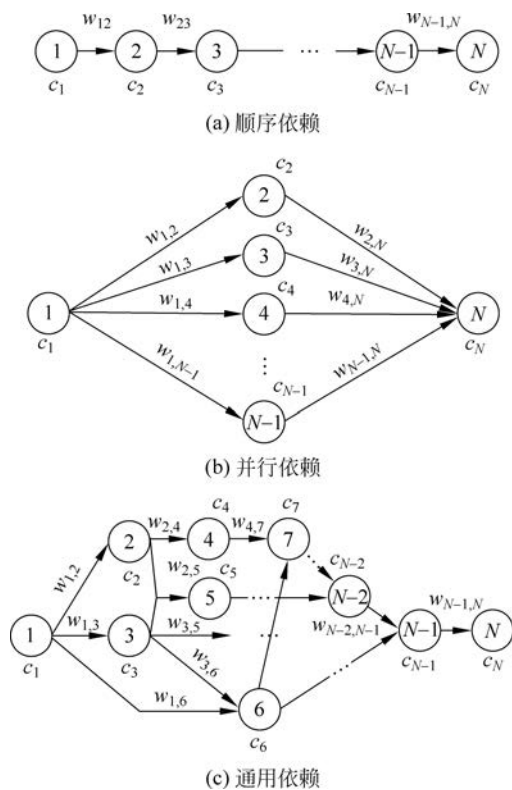


图 3-8 任务依赖模型

如图 3-9 所示,可以将数据划分成不同大小的数据块。在本实例中应用程序被抽象为具有两个参数的概要文件,即 $(I, L_{\max}(\cdot))$,其中 I 和 $L_{\max}$ 分别为计算输入的数据量和应用程序相关的延迟需求。假设数据可以划分成任何大小的子集,则该问题的最优解可以作为现实卸载策略的性能上界。

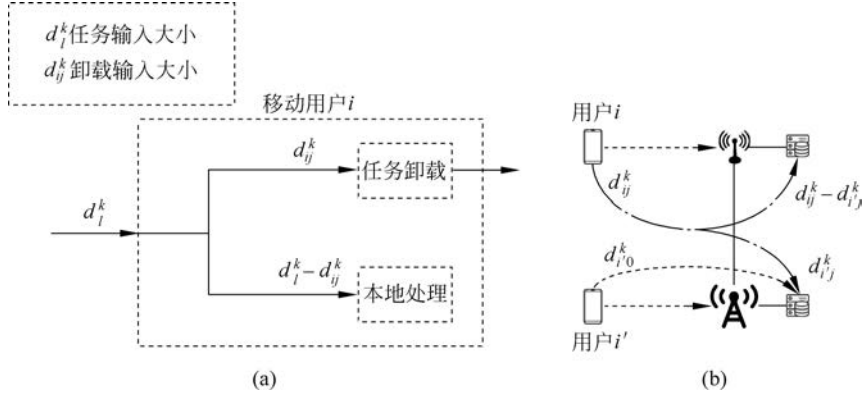


图 3-9 数据划分型可分负载计算卸载示意图

1. 本地计算模型

由于 I 为计算输入的数据量,所以本地执行应用所需 CPU 周期数为

$$C = \alpha I \quad (3-19)$$

式中: α 取决于应用程序的性质,如其计算复杂性。假设 λ 为本地执行的比特数与总输入数据比特数的比值,则本地计算所需要的时间为

$$t_l = \frac{\alpha \lambda I}{f_l} \quad (3-20)$$

式中: f_l 为本地设备的计算能力,根据能耗等于时间和功率的乘积,可以得到

$$P = k f_l^3 \quad (3-21)$$

$$E_l = \alpha \lambda I k f_l^2 \quad (3-22)$$

式中: P 为功率,由芯片架构和芯片处理能力决定。

2. 卸载计算模型

卸载计算的时间由上下行传输时间和计算时间构成,而卸载计算的能耗是由上下行阶段本地设备的能耗之和构成的。假设信道为频率平坦快瑞利衰落信道,则上传的数据量为

$$(1 - \lambda) I \quad (3-23)$$

3. 问题建模

在问题刻画过程中值得注意的是,由于本书采取边缘节点和本地并行执行的模式,因而任务完成时间取两者之间的最大值,可以表示为

$$L(f_l, P_l, \lambda) = \max\{t_l, t_c\} \quad (3-24)$$

而能量消耗等于本地计算能耗与卸载能耗之和,可表示为

$$E(f_l, P_l, \lambda) = E_l + E_c \quad (3-25)$$

(1) 能量最小化问题。为了延长电池寿命,在保证应用程序的等待时间要求的同时,最小化智能移动设备的总体能耗非常必要,模型可以刻画为能量最小化问题 P3_3。

$$\min E(f_l, P_l, \lambda) \quad (3-26)$$

$$\text{s. t. C1: } L(f_l, P_l, \lambda) \leq L_{\max}$$

$$\text{C2: } 0 \leq \lambda \leq 1$$

$$\text{C3: } 0 \leq P_l \leq P_{l\max}$$

$$\text{C4: } 0 \leq f_l \leq f_{l\max}$$

在能量最小化问题 P3_3 中,约束条件 C1 表示延迟约束,C2 表示 λ 的范围,C3 和 C4 分别是无线电接口和 CPU 施加的最大发射功率和计算速度限制。由于 $E(f_l, P_l, \lambda)$ 和 $L(f_l, P_l, \lambda)$ 都是非凸的,因此,问题 P3_3 是一个具有挑战性的非凸问题。值得注意的是,在 DVS 技术下,智能移动设备可以根据优化结果调整其计算速度。

(2) 延迟最小化问题。在移动智能设备对能量消耗有严格要求而应用程序对延迟敏感的情况下,最好使用有限的能量来尽可能缩短执行应用程序的延迟时间,最小化延迟问题刻画为优化问题 P3_4。

$$\min L(f_l, P_l, \lambda) \quad (3-27)$$

$$\text{s. t. C5: } E(f_l, P_l, \lambda) \leq E_{\max}$$

$$\text{C2, C3, C4.}$$

在延迟最小化问题 P3_4 中,约束条件 C5 表示能量约束,同样,问题 P3_4 也是一个具有挑战性的非凸问题。此外,由于目标函数不具有光滑性,问题 P3_4 也是一个非光滑性的优化问题。

由于两个问题都是非凸优化问题,因此首先对其进行可行性分析,再将非凸问题分别转化为凸问题进行求解,最后针对多边缘云环境进行相应改进。

3.7.2 应用划分型可分负载计算卸载

通常,应用程序由几个可分割且逻辑上独立的任务组成。例如,增强现实应用程序由 5 个关键任务组成,即视频源、跟踪器、映射器、目标识别器和渲染器。在这些任务中,跟踪器、映射器和目标识别器是计算密集型任务,可以凭借边缘计算节点进行计算。通过卸载,可以减少本地设备的能耗和应用程序处理时间。应用划分型可分负载计算卸载主要考虑可分割的应用程序,例如病毒扫描应用程序和图形压缩应用程序等。

可以将这样的应用程序划分为 K 个逻辑独立的任务,由于这种类型的应用程序预先知道要处理的数据量,因此可以将每个逻辑上独立的任务任意划分为几个块以支持并行性。

其问题基本刻画与上文类似,不再赘述。值得注意的是,在可分负载应用的计算卸载中,一般为了简便问题刻画和求解过程,大部分可分负载应用计算迁移问题所采用的依赖模型都是并行模型、顺序模型及二者的简单组合,很少涉及复杂的通用依赖模型,因此这一部分的研究还有待深入和探索。

3.8 可充电环境下的计算卸载

3.8.1 可充电环境下的计算卸载场景

除了一般的计算能力和资源约束之外,边缘计算环境中最常见的问题就是不论本地设备还是边缘节点都受到有限电池容量的限制。针对电池容量限制,一方面可以使用大容量

电池,但是受制于现有移动设备的尺寸及硬件成本,这种方案在大多数场景中并不可行。因此,过去常被用于无线可充电通信网络的能量收集技术已经成为边缘计算场景中的重要技术,通过能量收集可以捕获周围环境中的可再生能源,例如太阳辐射、风能和人体运动能量等,这有助于通信网络的自我维持和长久运行。

可充电环境下的计算卸载具有以下优点。

- (1) 获取的能量通常是绿色能源,因此使边缘计算成为绿色计算模式。
- (2) 为移动设备提供持续计算能力。
- (3) 富余的能量带来了更为强劲的处理能力。

然而,在可充电环境中需要考虑以下问题。

(1) 移动设备消耗的能量在每个时刻都是不同的,因此,如何在计算过程中将电池能量保持在稳定水平是一个有待解决的难题。

(2) 系统随机获取与时间相关的信息。基于这些因素,长期优化系统性能具有挑战性。

(3) 双远近问题(doubly near-far problem),即更接近接入点或能量源的用户比其他用户享有更好的信道条件,这意味着更近的用户通常可以更好地进行能量收集和传递。

本节就针对双远近问题和稳定性问题展开讨论。

3.8.2 针对双远近问题的可充电协作卸载

本实例采用时分多址(time division multiple access, TDMA)机制的充电卸载协议,设计了一个无线可充电的 MEC 框架以完成近远两类用户的计算密集型任务卸载,并提出了两种传输模式下的卸载方案:直接传输(direct transmission, DT)模式与中继传输(relay transmission, RT)模式,如图 3-10 所示。

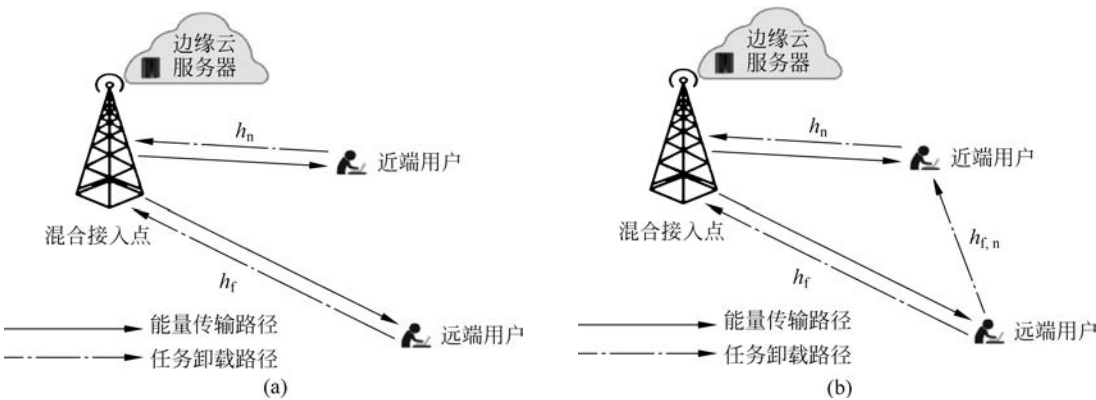


图 3-10 直接传输模式与中继传输模式

1. 直接传输模式

从用户公平性角度考虑,最大化在所有用户中能量效率最小的用户的能量效率。根据 max-min 标准,通过最大化每个用户的个体能效来达到用户公平,其中能量效率被定义为用户吞吐量与其收获能量的比率。

在直接传输模式中,如图 3-11 所示,混合接入点(hybrid access point, HAP)通过与边缘云的连接,从用户公平性的角度为处于两个不同距离的移动设备提供能量。远端用户在 τ_f 将任务卸载到 HAP,近端用户在 τ_n 将任务卸载到 HAP。

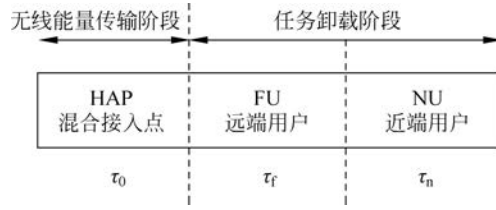


图 3-11 直接传输模式

直接传输模式中以最大化模式中每个用户的个体能效为目标,可以刻画为优化问题 P3_5。

$$\max_{r_0, \tau_i} \min_{i \in \{j, n\}} \eta_i^{\text{DT}}(\tau_0, \tau_i) \quad (3-28)$$

s. t.

$$\text{C1: } \sum_j \tau_j = 1, 0 \leq \tau_j \leq 1, j \in \{0, f, n\}$$

$$\text{C2: } R_i^{\text{DT}} \geq r_i^{\text{DT}}, i \in \{f, n\}$$

$$\text{C3: } E_i \leq C_i, i \in \{f, n\}$$

约束条件 C2 为最小速率,要满足服务质量的要求,C3 为最大收集能量,不应超过电池容量。在优化问题 P3_5 中,可以看出当固定计费时间时,减少具有较高能量有效性的用户的上载时间,将增加具有较低能量有效性的其他用户的上载时间。获得用户最佳能量有效性的有效方法是,每个用户都获得相同的能量有效性,因此可以直接求得优化问题 P3_5 的最优值。

2. 中继传输模式

从系统高效角度,最大化所有用户的能量效率。采取用户协作机制,近端用户(near user, NU)先作为中继节点,转发远端用户(far user, FU)的任务,再卸载自己的任务转发到边缘云。通过优化时间资源分配,来最大化系统的总能效。



图 3-12 中继传输模式

在中继传输模式中,如图 3-12 所示,与边缘云连接的混合接入点(hybrid access point, HAP)在 τ_0 为两个不同距离的移动设备提供能量。为了消除双近远效应,近端用户作为中继使用其收获的能量帮助远端用户将其信息转发给混合接入点。远端用户在 τ_f 将数据发送到近端用户,近端用户在 $\frac{1}{2}\tau_n$ 将远端用户的任务转发到混合接入点,再在剩余时间内将自己的计算任务卸载到混合接入点。

中继传输模式中以最大化系统能效为目标,可以刻画为优化问题 P3_6。

$$\max_{\tau} \eta^{\text{RT}}(\tau) \quad (3-29)$$

$$\text{s. t. } \text{D1: } \sum_j \tau_j = 1, 0 \leq \tau_j \leq 1, j \in \{0, f, n\}$$

$$D2: R_{n,HAP,l}^{RT} \geq r_{n,HAP,l}^{RT}, l=1,2$$

$$D3: E_i \leq C_i, i \in \{f,n\}$$

在优化问题 P3_6 中, D1 为时隙 τ_0 和 τ_i 的范围条件, D2 为最小速率应满足 QoS 的要求, D3 为确保所采集的最大能量不超过电池容量。优化变量 τ_0 、 τ_f 和 τ_n 的可行区域由约束 D1~D3 给出。可以通过约束 D1 减少变量的数量, 同时约束 D3 可以简化为与变量无关的恒定约束。因此优化问题 P3_6 可以转化为优化问题 P3_7。

$$\begin{aligned} \max_{\tau_0, \tau_n} \quad & \eta^{RT}(\tau_0, \tau_n) \\ \text{s. t.} \quad & D1: 0 \leq \tau_0, \tau_n \leq 1 \\ & D2: R_{n,HAP,l}^{RT} \geq r_{n,HAP,l}^{RT}, l=1,2 \\ & D3: E_i \leq C_i, i \in \{f,n\} \end{aligned} \quad (3-30)$$

通过以上讨论可以知道, 提出的能量效率最大化问题 P3_7 是凹凸分式规划问题, 属于特殊的非线性分式规划领域, 具有与凸优化理论共享的重要属性。通过使用 Dinkelbach 的方法, 可以将凹凸分式规划转化为凸优化问题, 并借助凸优化理论中的经典方法进行求解。

本章小结

计算卸载是边缘计算的关键技术, 也是边缘计算中关注的理论焦点。本章介绍了进行计算卸载的必要性, 计算卸载的概念、分类及步骤, 并以几个具体的应用场景为例, 介绍了边缘计算卸载模型和问题刻画。后续章节会针对具体的应用场景展开讨论并更详细的介绍。

习题

- (1) 简述计算卸载的分类及其分类标准。
- (2) 简述边缘计算进行计算卸载的意义。
- (3) 选取一篇关于边缘计算计算卸载的文献, 对其如何刻画计算卸载过程进行描述。
- (4) 尝试寻找并讨论不同应用场景下的计算卸载技术(如车联边缘计算、边缘智能环境等)。

参考文献

- [1] Kumar S, Tyagi M, Khanna A, et al. A survey of mobile computation offloading: applications, approaches and challenges[C]//International Conference on Advances in Computing and Communication Engineering (ICACCE), Paris, France, 2018, pp. 51-58.
- [2] Kumar K, Lu Y H. Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? [J]. Computer, 2010, 43(4): 51-56.
- [3] Mao Y, You C, Zhang J, et al. A Survey on Mobile Edge Computing: The Communication Perspective [J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.
- [4] Mach P, Becvar Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading [J]. IEEE Communications Surveys & Tutorials, 2017, PP(3): 1-1.
- [5] Habak K, Ammar M, Harras K A, et al. Femto Clouds: Leveraging mobile devices to provide cloud service at the edge[C]//IEEE International Conference on Cloud Computing (ICCC2015). New York,

- USA, 2015, pp. 9-16.
- [6] 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望[J]. 计算机研究与发展, 2019, 56(1): 73-93.
 - [7] 李子姝, 谢人超, 孙礼, 等. 移动边缘计算综述[J]. 电信科学, 2018, 34(1): 87-101.
 - [8] 谢人超, 廉晓飞, 贾庆民, 等. 移动边缘计算卸载技术综述[J]. 通信学报, 2018, 39(11): 138-155.
 - [9] Chen L, Zhou S, Xu J. Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networking, IEEE/ACM Transactions on Networks[J]. 2018, 26(4): 1619-1632.
 - [10] Mukherjee M, Kumar V, Kumar S, et al. Computation offloading strategy in heterogeneous fog computing with energy and delay constraints [C]//2020 IEEE International Conference on Communications (ICC2020), Dublin, Ireland, 2020, pp. 1-5.
 - [11] Pan Y, Chen M, Yang Z, et al. Energy-Efficient NOMA-Based Mobile Edge Computing Offloading [J]. IEEE Communications Letters, 2019, 23(2): 310-313.
 - [12] Meng Z, Xu H, Huang L, et al. Achieving energy efficiency through dynamic computing offloading in mobile edge-clouds[C]//2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Chengdu, China, 2018, 175-183.
 - [13] Tran T X, Pompili D. Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(1): 856-868.
 - [14] Huang L, Feng X, Zhang L, et al. Multi-Server Multi-User Multi-Task Computation Offloading for Mobile Edge Computing Networks[J]. Sensors, 2019, 19(6).
 - [15] Huang X, Zhang W, Yang J, et al. Market-based dynamic resource allocation in Mobile Edge Computing systems with multi-server and multi-user[J]. Computer Communications, 2021, 165: 43-52.
 - [16] Wang Y, Sheng M, Wang X, et al. Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling[J]. IEEE Transactions on Communications, 2016, 64(10): 4268-4282.
 - [17] Wu H, Wolter K. Software aging in mobile devices: Partial computation offloading as a solution[C]//IEEE International Symposium on Software Reliability Engineering Workshops, Gaithersburg, USA, 2016, pp. 125-131.
 - [18] Wang J F, Lv T J, Huang P M, et al. Mobility-Aware Partial Computation Offloading in Vehicular Networks: A Deep Reinforcement Learning Based Scheme[J]. 中国通信, 2020, 17(10): 31-49.
 - [19] Zhou S, Jadoon W. The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment[J]. Computer Networks, 2020, 178: 107334-107360.
 - [20] Dai Y, Xu D, Maharjan S, et al. Joint Computation Offloading and User Association in Multi-Task Mobile Edge Computing [J]. IEEE Transactions on Vehicular Technology, 2018, 67(12): 12313-12325.
 - [21] Ji L, Guo S. Energy-Efficient Cooperative Resource Allocation in Wireless Powered Mobile Edge Computing[J]. IEEE Internet of Things Journal, 2018, 6(3): 4744-4754.
 - [22] Wang F, Xu J, Wang X, et al. Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems[J]. IEEE Transactions on Wireless Communications, 2018, 17(3): 1784-1797.
 - [23] Mao Y, Zhang J, Letaief K B. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605.
 - [24] Bi S, Ying J, Zhang. Computation Rate Maximization for Wireless Powered Mobile-Edge Computing with Binary Computation Offloading[J]. IEEE Transactions on Wireless Communications, 2018, 17(6): 4177-4190.