计算机科学
COMPUTER SCIENCE

边云协同计算中成本感知的物联网数据处理方法

王晨华, 侯守璐, 刘秀磊

引用本文

王晨华, 侯守璐, 刘秀磊.边云协同计算中成本感知的物联网数据处理方法[J]. 计算机科学, 2022, 49(11A): 211000101-7.

WANG Chen-hua, HOU Shou-lu, LIU Xiu-lei. Cost-aware IoT Data Processing in Edge-Cloud Collaborative Computing [J]. Computer Science, 2022, 49(11A): 211000101-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000101

边云协同计算中成本感知的物联网数据处理方法

王晨华[1] 侯守璐[1] 刘秀磊[1,2]
1 北京信息科技大学数据与科学情报分析研究所 北京 100101
2 北京材料基因工程高精尖创新中心(北京信息科技大学) 北京 100101
(w1760014@163.com)

摘 要 随着物联网终端设备联网产生大量计算密集型的任务,文中提出了一种针对边云协同计算中成本优化的大数据处理方法。首先,所提算法考虑网络传输带宽约束及计算资源约束,联合优化带宽资源、计算资源分配以及动态卸载策略。其次,应用 MapReduce 框架,建立边云协同计算模型,通过 Lyapunov 优化理论将目标公式拆分成 4 个子问题,分别进行优化求解。大量对比实验结果表明,通过合理利用边缘和云的力量,在保证系统稳定性的前提下,所提算法可以有效地提高云计算的数据处理效率,降低服务供应商的数据处理开销,同时,该算法可降低任务成本总开销及提高性价比(队列长度与运营成本的比值),在物联网数据处理过程中,应用边云协同计算方法对降低成本开销并提高性价比具有重要意义。

关键词:物联网;边云协同;任务卸载;运营成本;Lyapunov 优化理论
中图法分类号 TP312

Cost-aware IoT Data Processing in Edge-Cloud Collaborative Computing

WANG Chen-hua[1], HOU Shou-lu[1] and LIU Xiu-lei[1,2]
1 Institute of Data and Scientific Information Analysis, Beijing Information Science and Technology University, Beijing 100101, China
2 Beijing Material Genetic Engineering High-precision Innovation Center, Beijing Information Science and Technology University, Beijing 100101, China

Abstract With the networking of Internet of Things(IoT) terminal devices, a large number of computation-intensive tasks appear. This paper proposes a cost-optimized big data processing method in the edge-cloud collaborative computing environment. Firstly, the proposed algorithm considers the constraints of network transmission bandwidth and computer resources, jointly optimizes bandwidth resources, and calculates resource distribution and dynamic offloading strategies. Secondly, based on the MapReduce framework, it establishes an edge-cloud collaborative computing model. According to Lyapunov optimization theory, it splits the target formula into four subproblems which can be solved separately. Comparative experiments results indicate that using the power of the edge rationally, the data processing efficiency of cloud computing can be improved and the expense of service providers can be reduced. At the same time, the algorithm can improve the cost performance(the ratio of queue length to operating cost). In processing IoT data, is of great significance to reduce operating costs by utilizing edge-cloud collaborative computing methods.

Keywords Internet of things, Edge-cloud collaboration, Task offloading, Operating cost, Lyapunov optimization theory

随着 5G 网络、物联网、人工智能等信息技术的蓬勃发展,物联网终端设备产生的数据每天都在爆炸式地增长,如现实生活中的车联网和智慧城市等。2016 年—2021 年,思科在全球云指数中指出接入互联网的设备数量从 171 亿增长到 271 亿[1],同时,根据 MaChina Research 的预测,未来物联网设备相互连接的数量将在 2025 年增长到 270 亿[2]。

目前,物联网终端设备上运行的应用服务复杂且计算密集,但由于相应的计算、存储能力有限,所有任务请求无法都在本地进行处理,这对数据网络传输提出了更高的需求,数据处理速度与未来网络的算法实用性也需进一步提升。为解决以上问题,国内外研究者提出将服务请求从物联网终端设备转移到云服务器,以充分利用其更高的计算能力与更大的储存容量。但由于云服务器多位于远程,远离终端设备,且终端设备上产生的数据量与日俱增,将服务请求调度到云服务

基金项目:促进高校分类发展-重点研究培育项目(2121YJPY225,2121YJPY226);科研机构创新能力建设-数据科学与情报分析研究所;促进高校内涵发展-面向边缘计算的创新科研平台建设项目(2020KYNH105)
This work was supported by the Promoting the Classified Development of Colleges and Universities-Key Research and Cultivation Project (2121YJPY225,2121YJPY226), Construction of Innovation Capability of Scientific Research Institutions-Institute of Data Science and Information Analysis and Promoting the Connotation Development of Colleges and Universities-Construction of an Innovative Scientific Research Platform for Edge Computing(2020KYNH105).
通信作者:刘秀磊(xiuleiliu@hotmail.com)

器上会加剧网络传输负担。之后研究者提出将计算能力有限的边缘服务器放在网络边缘，作为物联网终端设备与云服务器之间的桥梁[3]，提供存储、计算、网络等资源，以减少网络传输和多级转发带来的时延损耗，减少请求响应的时间，提升电池的续航能力，保证数据的安全性与隐私性[4-5]。因此，边云协同计算将是未来物联网数据处理的重要发展趋势。

然而，边云协同计算面临海量数据处理的高时延、高能耗和高成本的问题，已不能满足对网络资源的协同管理与实时响应[6]。Ma 等[7]提出通过近似雅可比交替方向乘子法(ADMM)算法，结合边云协同计算中涉及的时延、成本、能耗等多因素分析，建立以降低网络运营成本，提高用户服务质量为目标的模型算法。Wu 等[8]提出以分布式任务卸载算法为基础，对云边协同系统数据的资源分配与卸载策略进行优化，从而降低时延、能耗、时间复杂度，最大化地进行网络资源的分配与任务卸载。Su 等[9]提出了一种边云协同计算的框架，通过将云计算能力扩展到边缘服务器，进行资源部署与任务调度优化，高效利用边云协同数据处理的各自优势，减少系统数据处理的时延，从而提升边云协同系统的服务质量。

以上研究并未充分考虑服务供应商的实际需求。在保证用户合理需求的情况下，如何以最小成本获取最高价值是边云协同计算的真正意义。为此，本文从服务供应商的角度，提出了一种基于 Lyapunov 优化理论的边云协同计算，感知物联网数据的分布式实时优化方法，在保证系统稳定的前提下，通过优化边缘服务器与云服务器传输过程中的带宽资源分配以及任务计算过程中的资源调度，可降低任务成本总开销，且本文提出的方法可以合理地选择数据处理位置、确定边缘服务器与云服务器的活动数量与 CPU 频率，以减少服务器的存储资源空闲和浪费。

1 系统模型

本研究使用 MapReduce 计算模型，其中边缘服务器将部分数据片段合理地分配至云服务器，数据处理后产生临时结果的过程被称为 Map 操作，将产生的多个临时结果数据在多个云节点执行聚合操作，这个过程被称为 Reduce 操作。本文算法主要适用于计算密集型应用，通过物联网终端设备将数据收集到边缘服务器，将数据计算、存储、分割为一系列数据片段，再将一些未处理和已处理的数据片段分发到云服务器上进行计算、存储与分析。本文系统模型由 I 个分散的边缘节点(如物联网网关[10])与 J 个云节点共同组成，系统模型中定义 $I=\{1,2,3,\cdots,I\}$ 为边缘服务器索引集合，$J=\{1,2,3,\cdots,J\}$ 为云服务器索引集合，由于物联网设备产生的任务能近距离传输到边缘服务器上，因此不考虑物联网设备到边缘服务器之间的传输费用。

本文将物联网终端设备产生的海量数据中的一部分卸载在边缘服务器上进行计算处理，另一部分上传至云服务器上进行存储、计算，由于边缘服务器和云服务器、云服务器和云服务器之间的带宽相对有限，导致数据传输产生受限，因此，本文考虑的是一个离散的时间段系统，即时间槽长度为 τ，每个时刻 $t\in\{1,2,3,\cdots,T\}$，物联网终端设备在 t 时刻产生到达边缘节点 i 的数据量记为 $A_i(t)$，存在有限的最大值 $A_i^{\max}$，且满足 $0\leqslant A_i(t)\leqslant A_i^{\max}$，$\forall i\in I$，$\forall t$，系统模型中具体符号的含义如表 1 所列。

表 1 符号具体含义

Table 1 Specific meanings of symbols

符号	含义
I	边缘服务器的索引集合
J	云服务器的索引集合
τ	时间槽长度
$k_{ij}(t)$	t 时刻将 1GB 数据从边缘服务器 i 传输到云服务器 j 的链路价格
$\eta_{jk}(t)$	t 时刻将 1GB 数据从云服务器 j 迁移到云服务器 k 的链路价格
$a_{ij}(t)$	t 时刻边缘服务器 i 卸载到云服务器 j 的数据量
$o_{ij}(t)$	t 时刻边缘服务器 i 卸载到云服务器 j 的计算结果量
$r_{jk}(t)$	t 时刻云服务器 j 迁移到云服务器 k 的数据量
$\phi_i(t),\phi_j(t)$	t 时刻边缘和云服务器的单位能耗价格
$n_i(t),n_j(t)$	t 时刻边缘和云服务器的活动数量
$f_i(t),f_j(t)$	t 时刻边缘和云服务器的 CPU 频率
$G_i(t)$	t 时刻边缘服务器 i 原始数据量的队列积压
$Z_i(t)$	t 时刻边缘服务器 i 计算结果量的队列积压
$R_j(t)$	t 时刻云服务器 j 数据量的队列积压

1.1 通信模型

物联网终端设备联网后会产生海量数据，然而所有数据均在边缘服务器或全部上传至云服务器进行计算、存储，导致服务器空间被不合理占用，因此，本文提出由物联网终端设备、边缘服务器和云服务器组成的系统模型，在边缘计算场景下，基于 MapReduce 模型的物联网大数据任务计算卸载策略，利用 Lyapunov 优化理论建立稳定的队列模型，获得以节约网络带宽资源、提高任务计算效率、减少任务成本为目标的最优卸载策略。本文中提到的最小成本主要包括传输成本和计算成本，传输成本中 t 时刻边缘服务器上传数据到云服务器所产生的链路费用如式(1)所示：

$$B(t)=\sum_{i\in I}\sum_{j\in J}k_{ij}(t)[a_{ij}(t)+o_{ij}(t)] \tag{1}$$

其中，$k_{ij}(t)$ 表示在 t 时刻将 1GB 数据从边缘服务器 i 传输到云服务器 j 的链路价格，$a_{ij}(t)$，$o_{ij}(t)$ 分别表示在 t 时刻边缘服务器 i 上传到云服务器 j 上的卸载数据量和计算结果量。

当计算任务在边缘服务器执行时，服务器的功耗由其 CPU 工作频率决定，单个服务器 CPU 周期频率为 $f_i(t)$，并将处理器执行单位比特任务量所需的 CPU 运行周期定义为任务处理密度 ρ，在 t 时刻边缘服务器 i 的活动数量为 $n_i(t)$，边缘服务器 i 维护了一个积压队列 $G_i(t)$ 列(以字节为单位)，用以暂时存储等待处理的原始数据，边缘服务器队列积压的数据量随时间的动态变化如式(2)所示：

$$G_i(t+1)=\max\left[G_i(t)-\sum_{j\in J}a_{ij}(t)-n_i(t)\frac{f_i(t)}{\rho},0\right]+A_i(t) \tag{2}$$

其中，$A_i(t)$ 表示物联网设备 t 时刻卸载到边缘服务器 i 的待处理数据量。

边缘服务器 i 维护了一个积压队列 $Z_i(t)$，用来存储其并行顺序执行数据的计算结果量，其积压队列更新如式(3)所示：

$$Z_i(t+1)=\max[Z_i(t)-\sum_{j\in J}o_{ij}(t),0]+\alpha\frac{n_i(t)f_i(t)}{\rho} \tag{3}$$

其中，α 表示边缘服务器上处理结果与其原始数据的大小比值，$\alpha\in(0,1)$。

传输成本中，在 t 时刻云服务器之间迁移数据所产生的费用如式(4)所示：

$$M(t)=\sum_{j\in J}\sum_{k\in J\setminus j}\eta_{jk}(t)r_{jk}(t) \tag{4}$$

其中，$\eta_{jk}(t)$表示在 t 时刻将 1GB 数据从云服务器 j 迁移到云服务器 k 的链路价格，$r_{jk}(t)$表示在 t 时刻从云服务器 j 迁移到云服务器 k 的数据量。

当计算任务在云服务器执行时，服务器的功耗由 CPU 工作频率决定，在 t 时刻单个服务器的 CPU 周期频率为 $f_j(t)$，t 时刻云服务器 j 的活动数量为 $n_j(t)$，本研究考虑在云服务器上引用 MapReduce 编程模型的框架，执行多节点 Reduce 聚合操作，产生的队列积压数据随着时间的动态变化如式(5)所示：

$$R_j(t+1)=\max\left[R_j(t)-\sum_{k\in J\setminus j}r_{jk}(t)-\frac{n_j(t)f_j(t)}{\rho},0\right]+\sum_{i\in I}a_{ij}(t)+\sum_{i\in I}o_{ij}(t)+\sum_{k\in J\setminus j}r_{kj}(t) \tag{5}$$

其中，$r_{kj}(t)$表示在 t 时刻云服务器 k 迁移到云服务器 j 的数据结果量。

1.2 计算模型

在网络传输过程中，物联网设备会产生海量数据，服务器的硬件存储、计算能力、使用寿命已不能满足当前用户的需求。为节约网络带宽资源，提高任务计算效率，降低任务成本开销，在 t 时刻边缘服务器 i 和云服务器 j 产生的计算能耗费用分别为 $L(t)$和 $E(t)$，计算式如(6)、式(7)所示：

$$L(t)=\sum_{i\in I}\phi_i(t)P_i(t)\tau \tag{6}$$

$$E(t)=\sum_{j\in J}\phi_j(t)P_j(t)\tau \tag{7}$$

其中，$\phi_i(t)$和 $\phi_j(t)$分别表示在 t 时刻边缘服务器 i 和云服务器 j 的单位能耗价格，$P_i(t)\tau$ 和 $P_j(t)\tau$ 分别表示在 t 时刻边缘服务器 i 和云服务器 j 产生的总能耗。

在 t 时刻边缘服务器 i 和云服务器 j 产生的功耗分别为 $P_i(t)$，$P_j(t)$，计算式如式(8)、式(9)所示：

$$P_i(t)=n_i(t)(hf_i^2(t)+p_{\min}) \tag{8}$$

$$P_j(t)=n_j(t)(h(f_j(t)-f_j^{\min})^2+p_{\min}) \tag{9}$$

其中，h 表示耗电系数。

为保证边云协同计算平台的稳定性，边缘服务器和云服务器的时间队列大小之和必须有界，计算式如式(10)所示：

$$\bar{Q}\triangleq\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\{E[\sum_{i\in I}[G_i(t)+Z_i(t)]]+E[\sum_{j\in J}R_j(t)]\}<\infty \tag{10}$$

本文研究的边云协同计算平台在 t 时刻处理物联网终端设备大数据所产生的总任务成本开销，具体如式(11)所示：

$$C(t)=M(t)+B(t)+L(t)+E(t) \tag{11}$$

本文以最小化任务成本开销为目标，将边云协同计算平台的目标式(11)进行相应的转化，$X(t)=\{a_{ij}(t),o_{ij}(t),n_i(t),f_i(t),r_{jk}(t),n_j(t),f_j(t),\forall i,j,k\}$表示在 t 时刻需要优化的所有决策变量，具体如式(12)所示：

$$\begin{aligned}&\min_X\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E\{C(t)\}\\&\text{s.t. C1}:0\leqslant a_{ij}(t)+o_{ij}(t)\leqslant W_{ij}(t),\forall i,j,t\\&\quad\text{C2}:0\leqslant r_{jk}(t)\leqslant W_{jk}(t),\forall j,t\\&\quad\text{C3}:0\leqslant n_i(t)\leqslant n_i^{\max},0\leqslant f_i(t)\leqslant f_i^{\max},\forall i,t\\&\quad\text{C4}:0\leqslant n_j(t)\leqslant n_j^{\max},0\leqslant f_j(t)\leqslant f_j^{\max},\forall j,t\\&\quad\text{C5}:\bar{Q}<\infty\end{aligned} \tag{12}$$

其中，约束 C1 是为确保本系统平台在 t 时刻边缘服务器传输到云服务器的卸载数据量与计算结果量不超过链路带宽；约束 C2 是为确保本系统平台在 t 时刻云服务器之间的迁移数据量不超过链路带宽；约束 C3 和 C4 分别是为确保边缘服务器和云服务器的活动数量符合实际系统平台设计的最大数据量及 CPU 频率符合实际系统设置的最大 CPU 频率；约束 C5 是为确保边云协同系统中的所有队列的稳定性。

2 算法设计与实现

基于对本系统的设计模型进行详细分析，发现以上公式的队列是相互耦合的，难以直接求解，因此，我们需要考虑在系统稳定的情况下，将系统平台的目标问题解耦成 4 个子问题进行求解，分别为边缘节点与云节点的数据卸载与结果返回、云节点之间的数据迁移调度、边缘服务器与云服务器的计算资源管理。同时，应用 Lyapunov 优化理论、漂移加罚(Lyapunov drift-plus-penalty)函数分析边云协同服务器的数据积压队列，使用迭代式 MapReduce 模型在云服务器上执行多节点聚合操作，从而获得提高任务处理效率、减少队列等待时间的低开销多节点任务卸载策略。

2.1 基于 Lyapunov 优化理论分析

基于 Lyapunov 优化理论分析是一种经典的处理多节点任务卸载、多目标优化的方法，其主要通过控制平台系统的稳定性来实现性能的优化[11]，具有较强的理论指导意义，Lyapunov 方法是在现代控制理论中研究非线性系统稳定性的主要方法，如式(16a)－式(16d)所示，由于物联网数据中影响系统稳定性的因素较多，且存在多种隐性因素，因此使用 Lyapunov 方法研究该系统的稳定性具有实际意义。Lyapunov 函数具体如下：

$$L(\Theta(t))\triangleq\frac{1}{2}\sum_{i\in I}[G_i^2(t)+Z_i^2(t)]+\frac{1}{2}\sum_{j\in J}R_j^2(t) \tag{13}$$

其中，Lyapunov 函数对边缘服务器和云服务器端队列的平方求和，该函数值越大，则表示边云协同系统中等待队列越长，单时隙条件下 Lyapunov 漂移加罚函数 $\Delta(\Theta(t))$计算式具体如下：

$$\Delta(\Theta(t))=E\{L(\Theta(t+1))-L(\Theta(t))|\Theta(t)\} \tag{14}$$

在 Lyapunov 优化框架下，将边云协同系统中的队列长度和任务成本开销相结合，具体如下：

$$\min_X\Delta(\Theta(t))+VE\{C(\Theta(t))|\Theta(t)\} \tag{15}$$

其中，参数 V 可以看作边云协同系统中队列长度和任务成本开销之间的权衡参数，在实际应用中，服务供应商或用户可根据自己的需求确定。

本文为减少队列等待长度与最小化任务成本开销，结合式(15)的上界，不直接求解最小值，Neely 等[12]已证明此方法的性能与最优解的差距是有限的，且该方法可以任意地接近最优解，其中任务成本总开销的上界具体如下：

$$\begin{aligned}&\Delta(\Theta(t))+VE\{C(t)|\Theta(t)\}\leqslant B+\sum_{i\in I}G_i(t)A_i(t)+\\&\sum_{i\in I}\sum_{j\in J}\{[Vk_{ij}(t)-G_i(t)+R_j(t)]a_{ij}(t)+[R_j(t)-\\&Z_i(t)+Vk_{ij}(t)]o_{ij}(t)\}+\end{aligned} \tag{16a}$$

$$\sum_{i\in I}[V\phi_i(t)(hf_i^2(t)+P_{\min})\tau+Z_i(t)\alpha\frac{f_i(t)}{\rho}-G_i(t)\frac{f_i(t)}{\rho}]n_i(t)+ \tag{16b}$$

$$\sum_{j\in J}\sum_{k\in J\setminus j}[V\eta_{jk}(t)+R_k(t)-R_j(t)]r_{jk}(t)+ \tag{16c}$$

$$\sum_{j\in J}[V\phi_j(t)(h(f_j(t)-f_j^{\min})^2+P_{\min})\tau-R_j(t)\frac{f_j(t)}{\rho}]n_j(t) \tag{16d}$$

式(16a)中 B 的表达式如下,详细证明过程请参考文献[12]。

$$B=\frac{1}{2}\sum_{i\in I}\{(A_i^{\max})^2+(\sum_{j\in J}W_{ij}^{\max}+\frac{n_i^{\max}f_i^{\max}}{\rho})^2+(\sum_{j\in J}o_{ij}^{\max})^2+(\alpha\frac{n_i^{\max}f_i^{\max}}{\rho})^2\}+\frac{1}{2}\sum_{j\in J}\{(\sum_{k\in J\setminus j}W_{jk}^{\max}+n_j^{\max}\frac{f_j^{\max}}{\rho})^2+(2\sum_{i\in I}W_{ij}^{\max}+\sum_{k\in J\setminus j}W_{kj}^{\max})^2\}$$

本文所提的边云协同平台计算方法将式(16)拆分成式(16a)—式(16d)的 4 个子问题:边缘节点上传到云节点的数据卸载 $a_{ij}(t)$与结果返回 $o_{ij}(t)$、云节点之间的数据迁移调度 $r_{jk}(t)$、以及边缘服务器与云服务器上活动数量 $n_i(t)$和 $n_j(t)$以及 CPU 频率 $f_i(t)$和 $f_j(t)$的计算资源管理,这些子问题可以并行求解。

2.2 边缘节点与云节点的数据卸载与结果返回

在式(16a)中,不同边缘节点与云节点之间的每一对连接互不影响且并行运行,它们通过网络带宽传输边缘节点的任务卸载量、任务计算结果量到云节点上,进行相应的计算和存储等,因此求解以下子问题可获得在 t 时刻边缘节点最优传输的卸载量和计算结果量,具体如下:

$$\min_{a_{ij}(t),o_{ij}(t)}[R_j(t)+Vk_{ij}(t)-G_i(t)]a_{ij}(t)+[R_j(t)-Z_i(t)+Vk_{ij}(t)]o_{ij}(t)$$
$$\text{s.t. C1} \tag{17}$$

(1)在 t 时刻,如果 $R_j(t)+Vk_{ij}(t)-G_i(t)\geqslant 0$ 和 $R_j(t)-Z_i(t)+Vk_{ij}(t)\geqslant 0$,那么 $a_{ij}(t)$和 $o_{ij}(t)$的最优解具体如式(18a)所示:

$$\begin{cases}a_{ij}(t)=0\\o_{ij}(t)=0\end{cases} \tag{18a}$$

(2)如果队列 $G_i(t)<Z_i(t)$,$W_{ij}(t)$代表边边缘节点与云节点之间传输的最大带宽,那么 $a_{ij}(t)$和 $o_{ij}(t)$的最优解具体如下:

$$\begin{cases}a_{ij}(t)=0\\o_{ij}(t)=W_{ij}(t)\end{cases} \tag{18b}$$

(3)如果 $G_i(t)>Z_i(t)$,那么 $a_{ij}(t)$和 $o_{ij}(t)$的最优解具体如下:

$$\begin{cases}a_{ij}(t)=W_{ij}(t)\\o_{ij}(t)=0\end{cases} \tag{18c}$$

从本节公式中可以看出,子问题边缘节点与云节点之间的资源分配,可以通过比较决策变量的权重大小,即 $G_i(t)$和 $Z_i(t)$,来决定边缘节点传输到云节点的任务卸载量 $a_{ij}(t)$和计算结果量 $o_{ij}(t)$的大小。

2.3 边缘服务器的计算资源管理

由于边缘服务器之间的计算运行是互不影响并独立运行,因此,每个边缘服务器储存空间相对独立。通过联合式(16b)和本系统平台在 t 时刻边缘服务器的最优活动数量与 CPU 频率可通过以下子问题求解。

$$\min_{n_i(t),f_i(t)}[V\phi_i(t)(hf_i^2(t)+P_{\min})\tau+Z_i(t)\alpha\frac{f_i(t)}{\rho}-G_i(t)\frac{f_i(t)}{\rho}]n_i(t)$$
$$\text{s.t. C3} \tag{19}$$

为方便求解计算,对式(19)的中括号部分参数进行具体定义:$y(f_i(t))=V\phi_i(t)(hf_i^2(t)+P_{\min})\tau+Z_i(t)\alpha\frac{f_i(t)}{\rho}-G_i(t)\frac{f_i(t)}{\rho}$,子问题求解 $n_i(t)$的取值情况,具体如下:

$$n_i(t)=\begin{cases}n_i^{\max}, & \text{如果 } y(f_i(t))\leqslant 0\\n_i^{\min}, & \text{其他情况}\end{cases} \tag{20}$$

通过上式 $n_i(t)$的取值情况,对其 $f_i(t)$进行子问题求解取值,具体如下:

$$\min_{f_i(t)}\{n_i^{\max}\cdot y(f_i(t))\}\quad \text{s.t. } y(f_i(t))\leqslant 0$$
或 $$\min_{f_i(t)}\{n_i^{\min}\cdot y(f_i(t))\}\quad \text{s.t. } y(f_i(t))>0 \tag{21}$$

通过将 $f_i(t)$的约束条件设置在$[f_i^{\min},f_i^{\max}]$内,利用二次函数性质,可获得二次函数顶点的纵坐标值为 $\beta=-\frac{\alpha Z_i(t)-G_i(t)}{2\rho V\phi_i(t)h\tau}$,最后,通过对式(21)进行求解分析,可得到式(19)全局的最优解。

$$\min_{n_i(t),f_i(t)}\begin{Bmatrix}n_i^{\min}y(f_i^{\min}),n_i^{\min}y(\beta),n_i^{\min}y(f_i^{\max})\\n_i^{\max}y(f_i^{\min}),n_i^{\max}y(\beta),n_i^{\max}y(f_i^{\max})\end{Bmatrix} \tag{22}$$

本节子问题首先利用线性规划求解出边缘服务器的活动数量 $n_i(t)$,再利用二次函数性质求解出边缘服务器 CPU 的频率 $f_i(t)$,从而减少边缘服务器上的资源浪费。

通过对边缘节点和云节点的数据卸载量及结果返回、边缘服务器计算资源管理两个子问题求解分析,并对队列式(2)、式(3)的动态性进行比较,证明本系统的算法 1 可以在系统稳定的情况下,获得边缘服务器的最优卸载策略,具体步骤如算法 1 所示。

算法 1 边缘服务器的资源分配算法

输入:两个队列 $G_i(t)$、$Z_i(t)$的大小,边缘服务器的活动数量与 CPU 频率范围$\lceil n_i^{\min},n_i^{\max}\rceil$,$\lceil f_i^{\min},f_i^{\max}\rceil$,以及权重参数 V

输出:数据卸载量 $a_{ij}(t)$、计算结果量 $o_{ij}(t)$、活动数量 $n_i(t)$和 CPU 频率 $f_i(t)$

1. 初始化:$G_i(t)=0$,$Z_i(t)=0$
2. 每个边缘节点 i 与云节点 j($i\in I$,$j\in J$)
3. 如果 $R_j(t)+Vk_{ij}(t)-G_i(t)\geqslant 0$ 和 $R_j(t)-Z_i(t)+Vk_{ij}(t)\geqslant 0$
4. 边缘服务器数据卸载量 $a_{ij}(t)=0$,计算结果量 $o_{ij}(t)=0$
5. 如果 $G_i(t)<Z_i(t)$
6. 边缘服务器的卸载量 $a_{ij}(t)$为最大带宽
7. 如果 $G_i(t)>Z_i(t)$
8. 边缘服务器计算结果量 $o_{ij}(t)$为最大带宽
9. 根据式(19)计算 $y(f_i(t))$在连续变量下的固定点 β
10. 在频率$[f_i^{\min},f_i^{\max}]$中,根据式(22)设置边缘服务器的活动数量 $n_i(t)$和 CPU 频率 $f_i(t)$

2.4 云节点之间的数据迁移调度

在式(16c)中,不同云节点之间的卸载传输之间互不影响且并行运行,其中多个云节点用于聚合数据计算最终结果量,因此,通过求解以下子问题可以获得在 t 时刻云节点的最优卸载量,具体如下:

$$\min_{r_{jk}(t)}[V\eta_{jk}(t)+R_k(t)-R_j(t)]r_{jk}(t)$$
$$\text{s.t. C2} \tag{23}$$

式(23)是一个线性规划求解问题,其最优解具体如下:

$$r_{jk}(t)=\begin{cases}W_{jk}(t), & \text{如果 } V\eta_{jk}(t)+R_k(t)-R_j(t)<0\\0, & \text{其他情况}\end{cases} \tag{24}$$

由上式可知,$V\eta_{jk}(t)+R_k(t)-R_j(t)$为权重大小,用于决定云节点之间的卸载量 $r_{jk}(t)$的大小,系统在云服务器中选择了多个节点用于执行最终结果聚合操作。

2.5 云服务器的计算资源管理

云服务器运行情况与上述边缘服务器基本相同，通过联合公式(16d)和系统平台在 t 时刻云服务器的最优活动数量与 CPU 频率，对以下子问题求解，具体如下：

$$\min_{n_j(t),f_j(t)} [V\phi_j(t)(h(f_j(t)-f_j^{\min})^2+P_{\min})\tau- R_j(t)\frac{f_j(t)}{\rho}]n_j(t)$$

$$\text{s.t. C4} \tag{25}$$

为简化求解过程，对式(25)中的中括号部分内容进行具体的定义：$y(f_j(t))=V\phi_j(t)(h(f_j(t)-f_j^{\min})^2+P_{\min})\tau-R_j(t)\frac{f_j(t)}{\rho}$，子问题求解中的 $n_j(t)$ 的取值情况具体如下：

$$n_j(t)=\begin{cases} n_j^{\max}, & \text{如果 } y(f_j(t))\leqslant 0 \\ n_j^{\min}, & \text{其他情况} \end{cases} \tag{26}$$

通过式(26)中 $n_j(t)$ 的取值情况，再对其 $f_j(t)$ 进行子问题求解取值，具体如下：

$$\min_{f_j(t)}\{n_j^{\max}\cdot y(f_j(t))\} \quad \text{s.t. } y(f_j(t))\leqslant 0$$

或 $$\min_{f_j(t)}\{n_j^{\min}\cdot y(f_j(t))\} \quad \text{s.t. } y(f_j(t))>0 \tag{27}$$

通过将 $f_j(t)$ 的约束条件设置为 $[f_j^{\min},f_j^{\max}]$ 内，利用二次函数性质，能够确定二次函数顶点的纵坐标值为 $\mu=\frac{2V\phi_j(t)hf_j^{\min}+R_j(t)/\rho}{2V\phi_j(t)h\tau}$，最后，通过对式(27)进行求解分析，可以获得式(25)全局的最优解，具体如下：

$$\min_{n_j(t),f_j(t)}\left\{\begin{matrix} n_j^{\min}y(f_j^{\min}), n_j^{\min}y(\mu), n_j^{\min}y(f_j^{\max}) \\ n_j^{\max}y(f_j^{\min}), n_j^{\max}y(\mu), n_j^{\max}y(f_j^{\max}) \end{matrix}\right\} \tag{28}$$

本节子问题首先利用线性规划求解出云服务器的活动数量 $n_j(t)$，再利用二次函数性质求得云服务器 CPU 的频率 $f_j(t)$，从而减少云服务器上的资源浪费。

通过对云节点之间迁移调度、云服务器计算资源管理两个子问题求解分析，并对队列式(5)的动态性进行比较，证明本系统的算法 2 可以在系统稳定的情况下，获得边缘服务器的最优卸载策略，具体步骤如算法 2 所示。

算法 2 云服务器的资源分配算法

输入：一个队列 $R_j(t)$ 的大小，云服务器的活动数量与 CPU 频率范围 $[n_j^{\min},n_j^{\max}]$，$[f_j^{\min},f_j^{\max}]$，以及权重参数 V

输出：迁移量 $r_{jk}(t)$、活动数量 $n_j(t)$ 和 CPU 频率 $f_j(t)$

1. 初始化：$R_j(t)=0$
2. 每个云节点 j，k($j\in j$，$k\in J\backslash j$)
3. 如果 $V\eta_{jk}(t)+R_k(t)-R_j(t)<0$
4. 云服务器迁移量 $r_{jk}(t)$ 为最大带宽
5. 否则
6. 云服务器的迁移量 $r_{jk}(t)=0$
7. 根据式(25)计算 $y(f_j(t))$ 在连续变量下的固定点 μ
8. 在频率 $[f_j^{\min},f_j^{\max}]$ 中，根据式(28)设置云服务器的活动数量 $n_j(t)$ 和 CPU 频率 $f_j(t)$

2.6 性能分析

本文分析了边云协同平台系统的任务成本开销和队列长度等性能，并就算法进行了有效证明，从而为服务供应商在运营成本开销和服务质量方面提供了参考价值。有学者提出令向量 $\boldsymbol{\lambda}=(\lambda_i)$，$\forall i\in I$，集合了所有边缘服务器上时间平均数据的达到率，其中 $\boldsymbol{\lambda}_i=E\{I_i(t)\}$ 向量定义了边云协同计算平台的容量区域，即时间平均数据的达到率闭包集 $\boldsymbol{\lambda}$[13-14]，此到达率的闭包集内存在合理的边缘服务器和云服务器优化数据卸载、计算结果迁移以及资源分配算法以确保式(10)成立。

定理 1 如果存在一个常数 $\varepsilon>0$ 满足 $\boldsymbol{\lambda}+\varepsilon\in\Lambda$，本文提出的方法具体提供以下性能的保证：

$$\bar{Q}\triangleq\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\{E[\sum_{i\in I}[G_i(t)+Z_i(t)]]+E[\sum_{j\in J}R_j(t)]\}< \frac{B+VE^{\max}}{\varepsilon} \tag{29}$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E[C(t)]\leqslant E^*+\frac{B}{V} \tag{30}$$

其中，E^* 和 $E^{\max}$ 分别表示运营成本开销的最大值和最优值，B 值详见式(16)。定理 1 中的常数 ε 表示平均时间数据的达到率向量 $\boldsymbol{\lambda}$ 与容量区域 Λ 的边界之间的距离。定理 1 的结果表明，本文提出的保证边云协同计算平台稳定的算法，需要 $\boldsymbol{\lambda}$ 在 Λ 范围内，可以把本系统的时间平均队列积压的限定设置为一个常数，时间平均队列积压与 $\boldsymbol{\lambda}$ 和 Λ 的边界距离呈反比，且 Λ 不随时间变化，数据达到率仍维持稳定，假设 Λ 和参数 ε 存在，由此可证明定理 1 中的折衷性和渐近最优性，表明本系统边云协同计算平台上的运营成本开销和队列长度之间存在[O(1/V)，O(V)]上的折衷，参数 V 的值用于微调运营成本开销和性能之间的权重，并由服务供应商根据实际场景应用的需求进行确定。

3 仿真实验与分析

本文基于 MATLAB2016b 仿真平台模拟了一个边云协同计算平台系统，此系统由 16 个边缘节点和 4 个云节点组成，假设边缘服务器的数据达到量 $A_i(t)$ 服从正态分布，系统中允许服务器的活动数量范围在 20 到 500 之间，服务器的 CPU 频率范围是两组离散的数组，即边缘服务器和云服务器的频率分别为[2，2.5，3，3.5，4]和[6，6.5，7，7.5，8] GHz；服务器的单位能耗价格在[15.12，66.27]($/MWh)内分布[15-16]；最后，本系统每个数据点的模拟实验总时间 $T=3000$，每个时间段 $\tau=60$；具体仿真实验参数如表 2 所列。

表 2 仿真实验参数

Table 2 Simulation experiment parameters

参数	参数取值
系统带宽/Gbps	1～20
边缘服务器 CPU 频率 $f_i(t)$/GHz	2.0～4.0
云服务器的 CPU 频率 $f_j(t)$/GHz	6.0～8.0
服务器最小运行功率 $P_{\min}$/W	120
链路传输价格 $k_{ij}(t)$，$\eta_{jk}(t)$/($/GB)	0.025～0.085
单位能耗价格 $\phi_i(t)$，$\phi_j(t)$/($/MWh)	15.12～66.27

3.1 边云协同中参数 V 对算法性能的影响

根据 2.6 节性能分析可知，参数 V 对本系统运营成本开销存在较大影响，如图 1 所示，随着参数 V 的增大，时间平均队列长度呈增长趋势，且时间平均运营成本开销逐渐减少，由此验证了定理 1 成立，即在已知运营成本开销预算的情况下，通过调整参数 V，可达到服务供应商的应用要求。

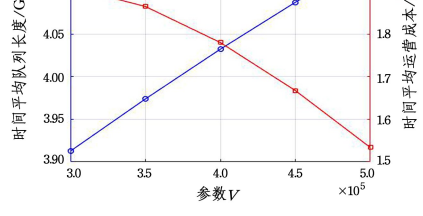

图 1 参数 V 对队列长度和成本开销的影响

Fig. 1 Effect of parameter V on queue length and cost overhead

3.2 边云协同中不同算法的性能对比

考虑在相同边云协同场景下[17-18]，将所提算法与对比实验算法1、算法2进行对比分析，其中对比实验算法1使用最大边缘服务器的活动数量与CPU频率；对比实验算法2使用最大云服务器的活动数量与CPU频率，对比实验算法1和2的其他配置与本文所提算法保持一致。图2—图4分别描述了对比方法的时间平均性价比、任务成本开销与队列长度随着数据到达量的变化趋势，所提方法均在参数 $V=5\times10^3$ 时。

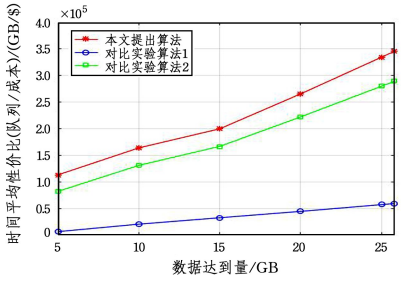

图2 性价比的比较

Fig. 2 Comparison of cost performance

如图2所示，随着数据达到量的增加，时间平均性价比也呈现出递增的趋势。从图中可以明显看出，本文提出的算法明显优于其他两种算法，且时间平均性价比提高约30%，原因主要是所提算法以最小化任务成本开销为目标，合理考虑到边缘服务器与云服务器的活动数量、CPU频率，减少了资源的浪费，从而节省了空闲服务器运转所需的功耗成本开销。

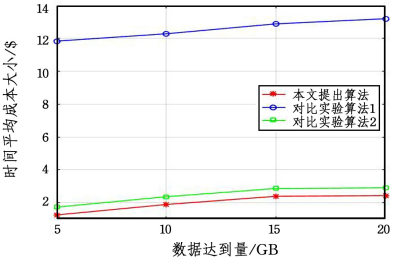

图3 任务成本开销的比较

Fig. 3 Comparison of task cost overhead

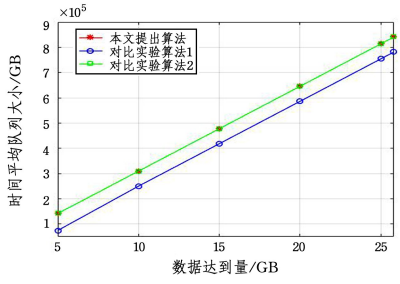

图4 队列长度的比较

Fig. 4 Comparison of queue length

如图3、图4所示，随着数据到达量的增加，3种不同算法的时间平均成本逐渐增加直至达到平稳趋势，时间平均队列大小呈现出递增的趋势。通过考虑边缘服务器与云服务器之间动态合理地分发任务量、计算能耗，从而减少了传输能耗和计算能耗的成本开销，如图3所示，对比算法1明显优于其他两种算法，曲线达到平稳时数据到达量至最大值，尽管对比算法2与本文算法曲线趋势相同，但本文算法的时间平均成本开销最小，降低了约87%。但图4中显示，尽管3种不同算法数据到达量曲线趋势相同，在保证系统内所有队列的稳定性下，本文所提算法的时间平均队列效果较对比算法1稍差，主要原因是本文算法中服务器活动数量、CPU频率是根据数据到达量进行分配，并非选用最大的服务器活动数量与CPU频率，进而导致算法的时间平均队列值大于对比算法1。

结束语 本文研究了边云协同计算中成本感知的物联网数据处理方法，为边云协同计算平台提供了一种新的分布式优化方法，首先，本文方法应用MapReduce模型建立边云协同计算模型，隐藏边云协同模式中的复杂管理功能，屏蔽边云协同环境中动态、异构、层次化的各类资源；其次，基于Lyapunov优化理论将目标公式拆分成4个子问题，并进行最优求解分析；最后，仿真实验结果表明本文方法在保证系统队列的稳定性的情况下，成本开销降低了约87%，性价比提高了约30%。

参 考 文 献

[1] DING T, CAO J N, YANG L, et al. Edge Computing: Applications, State-of-the-Art and Challenges [J]. ZTE Technology, 2019, 25(3): 2-7.

[2] LIANG J B, TIAN F S, JIANG C, et al. Survey on task offloading techniques for mobile edge computing with multi-device and multi-servers in the Internet of Things[J]. Computer Science, 2021, 48(1): 16-25.

[3] SHI W S, CAO J, ZHANG Q, et al. Edge computing: Vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[4] LEE J. A view of cloud computing[J]. International Journal of Networked and Distributed Computing, 2013, 1(1): 2-8.

[5] SHI W, DUSTDAR S. The Promise of Edge Computing[J]. Computer, 2016, 49(5): 78-81.

[6] TU Y P, CHEN H M, YAN L J. Offloading decision problems for edge computing in IoT systems: modeling, solution and classification [J]. Small Microcomputer System, 2021, 42(10): 2145-2152.

[7] MA L, LIU M, LI C, et al. A cloud-edge collaborative computing task scheduling algorithm for 6G edge networks [J]. Journal of Beijing University of Posts and Telecommunications, 2020, 43(6): 66-73.

[8] WU X W, LIAO J X. Game-based resource allocation and task offloading scheme in collaborative cloud-edge computing system [J/OL]. https://doi.org/10.16182/j.issn1004731x.joss.21-0077.

[9] SU M F, WANG G J, LI R F. Resource deployment with prediction and task scheduling optimization in edge-cloud collaborative computing [J/OL]. http://kns.cnki.net/kcms/detail/11.1777.tp.20210316.1150.004.html.

[10] CZIVA R, PEZAROS D P. Container network functions: bringing NFV to the network edge[J]. IEEE Communications Magazine, 2017, 55(6): 24-31.

[11] DING X Q, XUE J B. System resource allocation strategy based on Lyapunov optimization in edge computing[J]. Microelectronics and Computer, 2020, 37(2): 63-68.

[12] NEELY M J. Stochastic network optimization with application to communication and queueing systems[J]. Synthesis Lectures

on Communication Networks,2010,3(1):1-211.

[13] XIAO W,BAO W,ZHU X,et al. Cost-Aware Big Data Processing Across GeoDistributed Datacenters[J]. IEEE Transactions on Parallel and Distributed Systems,2017,28(11):3114-3127.

[14] ZHOU Z,LIU F,ZOU R,et al. Carbon-aware online control of geo-distributed cloud services[J]. IEEE Transactions on Parallel and Distributed Systems,2015,27(9):2506-2519.

[15] HOU S L. Resource management in fog-assisted cloud computing for Internet of Things[D]. Beijing: Beijing University of Posts and Telecommunications,2020.

[16] CHEN L,XU Y,LU Z,et al. IoT Microservice Deployment in Edge-cloud Hybrid Environment Using Reinforcement Learning[J]. IEEE Internet of Things Journal,2020,8(16):12610-12622.

[17] HUANG M,LIU W,WANG T,et al. A cloud-MEC collaborative task offloading scheme with service orchestration[J]. IEEE Internet of Things Journal,2019,7(7):5792-5805.

[18] BONADIO A,CHITI F,FANTACCI R. Performance Analysis of an Edge Computing SaaS System for Mobile Users[J]. IEEE Transactions on Vehicular Technology,2019,69(2):2049-2057.

WANG Chen-hua, born in 1994, postgraduate. Her main research interests include Internet of things and edge computing.

LIU Xiu-lei, born in 1981, Ph. D, is a member of China Computer Federation. His main research interests include ontology matching, semantic sensor, knowledge graph, semantic Web and semantic search.