Software Engineering and Applications 软件工程与应用, 2019, 8(6), 295-302
Published Online December 2019 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2019.86036

Hans汉斯

A Scheduling Algorithm for Minimum Total Delay Time in Mobile Edge Computing

Yuzhou Wen

School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou Zhejiang
Email: yuzhouwen95@163.com

Received: Nov. 7th, 2019; accepted: Nov. 22nd, 2019; published: Nov. 29th, 2019

Abstract

As an emerging architecture, mobile edge computing extends cloud computing services to the edge of the network close to users through mobile edge computing servers, meeting the needs of applications that require real-time control and real-time data analysis. However, due to the limited computing power of the mobile edge computing server, the delay time of the task is long. In order to improve the status quo, this paper proposes a scheduling algorithm with minimum total delay time. The server determines the optimal order of task calculation to minimize the total lag time. In addition, this paper also proposes an incentive mechanism that allows users to submit tasks with reasonable computational effort and expected completion time, while reducing the number and amount of submitted tasks when the server computing resources are insufficient. The results show that the proposed algorithm's performance is close to the traditional scheduling algorithms, and increases 17% to 200% in total delay time and average delay time.

Keywords

Mobile Edge Computing, Scheduling Algorithm, Delay Time, Incentive Mechanism

一种移动边缘计算中最小总滞后时间的调度算法

温雨舟

浙江理工大学信息学院，浙江 杭州
Email: yuzhouwen95@163.com

收稿日期：2019年11月7日；录用日期：2019年11月22日；发布日期：2019年11月29日

摘 要

移动边缘计算作为一个新兴架构，将云计算服务通过移动边缘计算服务器扩展到靠近用户的网络边缘，

文章引用: 温雨舟. 一种移动边缘计算中最小总滞后时间的调度算法[J]. 软件工程与应用, 2019, 8(6): 295-302.
DOI: 10.12677/sea.2019.86036

满足了需要实时控制和即时数据分析的应用需求。然而，由于移动边缘计算服务器的计算能力有限，导致任务的滞后时间较长。为了改善现状，本文提出了一种最小总滞后时间的调度算法，服务器确定任务计算的最优顺序以最小化总滞后时间。此外，本文还提出了一种激励机制，使得用户提交具有合理的计算量和预期完成时间的任务，同时在服务器计算资源不足时减少提交任务的数量和计算量。结果表明，该算法在接近传统调度算法性能的同时，在总滞后时间和平均滞后时间上提高了17%到200%。

关键词

移动边缘计算，调度算法，滞后时间，激励机制

1. 引言

近年来，随着信息和通信技术的不断发展，移动设备的数量呈现爆发式增长，移动数据流量也显著提高。智能设备和新应用的到来，出现了许多新需求，现有的移动云计算(MCC)框架已经不能满足其低时延、高带宽、隐私安全保障等需求[1] [2] [3]。移动边缘计算(MEC)作为一个新兴架构，将云计算服务通过移动边缘计算服务器扩展到靠近用户的网络边缘，解决了当前云计算框架高时延、低带宽、隐私安全威胁等问题，逐渐成为了当前的研究热点之一[4] [5] [6] [7]。

随着人工智能技术的发展，出现了许多需要实时控制和即时数据分析且应用于移动边缘计算框架的新技术，如智能电网[8]、自动驾驶[9]和智慧医疗[10]等。然而，传统的移动边缘计算调度方法由于边缘计算服务器的计算能力有限，导致需要任务的滞后时间较长，严重影响移动边缘计算框架的正常运行。此外，传统的调度方法没有设置有效的激励机制，使得用户在服务器的计算资源不足时仍大量提交任务或者向服务器提交非必要的任务，导致大量的任务堆积，严重影响计算卸载服务的实时性。

对于边缘计算计算卸载调度问题，很多学者已经进行了研究。

文献[11]提出了一个基于深度增强学习的区块链增强型移动边缘计算计算卸载方法，考虑了区块链数据挖掘任务和数据处理任务，有效地避免了无用的探索并在不降低性能的情况下加快了收敛速度；文献[12]提出了一种移动边缘计算中具有数据缓存的计算卸载方法，避免了多个移动用户将重复的计算任务卸载到网络边缘，减少了移动终端的总体执行延迟；文献[13]提出了一种用户非对称边缘计算中基于学习驱动的计算卸载方法，利用任务执行时间与边缘服务器的关联性，在提高任务卸载速率的同时获得较低的处理延迟。然而这些工作都没有考虑到移动边缘计算服务器计算性能有限的情况，如果服务器遇到较大的计算量或较多的任务量，将会导致设计的调度算法无法正常运行。

文献[14]设计了一个基于联合和价格的计算卸载方法，采用博弈论设计了合理的激励机制，保证了计算卸载的效率；文献[15]提出了一个软件定义超密集网络中采用移动边缘计算的任务卸载方法，最大程度地减少了延迟，同时节约了用户设备的电池寿命；文献[16]设计了一个异构网络中用于物联网的能量感知计算卸载方法，提出了一个迭代解决方案框架，制定了传输功率分配策略和计算分流方案。然而这些工作没有考虑到移动边缘计算框架下用户在服务器的计算资源不足时仍大量提交任务或者向服务器提交非必要的任务，导致大量的任务堆积，严重影响计算卸载服务的实时性。

针对以上问题，本文提出了一种移动边缘计算中基于动态规划的最小化总滞后时间计算卸载调度策

略。在该策略下，用户在向边缘计算服务器提交任务的同时提交该任务期望的完成时间，边缘计算服务器根据用户提交的数据量和期望的完成时间对任务进行调度，在多项式时间内确定任务计算的最优顺序。与此同时，本文还提出了一种激励机制，使得用户提交具有合理的计算量和预期完成时间的任务，同时在服务器计算资源不足时减少提交任务的数量和计算量。实验表明，该算法在接近传统调度算法性能的同时，在总滞后时间和平均滞后时间上提高了17%到200%。

2. 问题描述

本文考虑如图1所示的移动边缘计算框架，有 n 个用户和1个移动边缘计算服务器。每个用户将一个计算卸载任务提交给移动边缘服务器，服务器在计算结束后通过高带宽链路将计算结果反馈给用户。

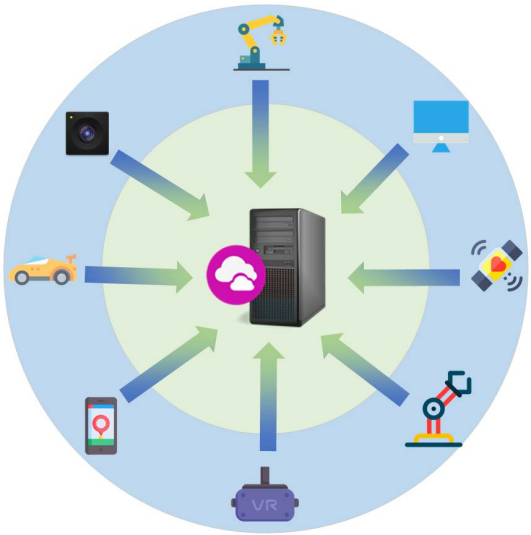

Figure 1. Framework of mobile edge computing
图 1. 移动边缘计算框架

用户可以用集合 $U=\{u_0,u_1,u_2,\cdots,u_n\}$ 表示，其提交的计算卸载任务为 $T=(P,D)=\{t_0,t_1,t_2,\cdots,t_n\}$，其中 $P=\{p_0,p_1,p_2,\cdots,p_n\}$ 表示该任务所需要的计算时间，$D=\{d_0,d_1,d_2,\cdots,d_n\}$ 表示该任务期望的完成时间。设任务的滞后时间为 $L=\{l_0,l_1,l_2,\cdots,l_n\}$，如果该任务在 d_i 之前完成，则该任务的滞后时间 $l_i=0$；如果该任务在 d_i 之后的时刻 f_i 完成，则该任务的滞后时间 $l_i=f_i-d_i$。任务的开始时间为 $S=\{s_0,s_1,s_2,\cdots,s_n\}$。

移动边缘计算服务器用 M 表示，其部署在靠近用户的网络边缘，与用户之间的时延足够低且链路带宽足够高，用户提交任务给服务器和服务器返回结果给用户的传输时间可忽略不计。但是，移动边缘计算服务器的计算性能有限，一次只能运行一个计算卸载任务，因此需要确定任务计算的最优顺序以最小化总滞后时间。

综上，本文的优化目标为在移动边缘计算服务器只能同时计算一个任务且任务计算开始后不可中断的条件下最小化总滞后时间：

$$\begin{aligned} \min \quad & \sum_{i=0}^{n} l_i \\ \text{s.t.} \quad & f_i \le \sum_{i=0}^{n} p_i, \forall i \\ & s_i + p_i = f_i, \forall i \end{aligned} \tag{1}$$

3. 调度算法

本文提出的基于动态规划的最小总滞后时间调度算法需要确定计算任务$\{t_0,t_1,t_2,\cdots,t_n\}$的最优顺序以获得最小总滞后时间。用$k$表示这些任务中具有最大计算时间的任务。因此对于一个$\delta(0\le\delta\le n-k)$存在一个最优调度[17]，该最优调度可以看作以下三个任务子集的串联：

1) 某种顺序的任务排列：$\{t_0,t_1,\cdots,t_{k-1},t_{k+1},\cdots,t_{k+\delta}\}$；

2) 任务t_k；

3) 某种顺序的任务排列：$\{t_{k+\delta+1},t_{k+\delta+2},\cdots,t_n\}$；

其中，任务t_k的完成时间$f_k(\delta)$为：

$$f_k(\delta)=\sum_{j\le k|\delta}p_j \tag{2}$$

显然，为了整个调度最优，需要让第一个和第三个任务子集的内部也最优。因此本文设计了一个基于动态规划的调度算法，在确定任务集合的任务子集的最后顺序之后再确定任务集合的最优顺序。设子集$T(j,n,k)$表示任务$\{t_j,t_{j+1},\cdots,t_{n-1},t_n\}$中计算时间小于等于任务$k$计算时间$p_k$的所有任务，其中任务$k$不是这个子集的一部分，即使$j\le k\le n$时也不是。$V(T(j,n,k),\gamma)$表示该子集在时间$\gamma$开始的一个最优顺序下该子集的总滞后时间。

综上所述，基于动态规划的最小总滞后时间调度算法描述如下。

初始条件：

$$V(\varnothing,\gamma)=0 \tag{3}$$

$$V(\{j\},\gamma)=\max(0,\gamma+p_j-d_j) \tag{4}$$

该初始条件表示如果子集在任意时间γ为空时，总滞后时间为0。如果子集只有一个任务时，总滞后时间为该任务的滞后时间。

递归关系：

$$\begin{aligned}V(T(j,n,k),t)=&\min(V(j,k'+\delta,k'),\gamma)+\max(0,f_{k'}(\delta)-d_{k'})\\&+V(T(k'+\delta+1,n,k'),f_{k'}(\delta))\end{aligned} \tag{5}$$

其中，k'为该子集中计算时间最大的任务，即：

$$p_{k'}=\max(p_{j'}\mid j'\in T(j,n,k)) \tag{6}$$

开始调度时的初始方程为：

$$V(\{t_0,t_1,\cdots,t_n\},0) \tag{7}$$

该算法的最坏情况下的时间复杂度可确定如下。在时间λ内最多有$O(n^3)$个任务子集$T(j,n,k)$和$\sum p_j$个时间点，因此本算法最多需要解$O(n^3\sum p_j)$个递归方程。求解每个递归方程的时间复杂度为$O(n)$，因此本算法的计算复杂度以$O(n^4\sum p_j)$为界，显然为n的多项式。由于边缘计算的用户规模较小，因此该算法的时间复杂度为$O(n^4)$。

4. 激励机制

由于边缘服务器的计算能力有限，如果服务器中当前等待的任务较多，之后提交的任务会获得非常大的滞后时间，严重影响移动边缘计算框架的正常运行。因此，需要根据服务器当前等待的任务数量设置合理的激励机制使得用户在服务器等待任务较多时尽量减少任务的提交或者减少任务的数据量。

此外，用户提交的任务具有较长的计算时间将会占去较多的服务器计算资源，提交任务的预期完成时间与计算时间接近将会消耗服务器最近的计算资源，该资源是最为宝贵的。因此，为了使得用户提交较少数据量的任务且提交合理的任务预期完成时间，也需要设置合理的激励机制。

设当前的时间片在服务器中正在等待的任务所需要的计算时间为 P，P 的值为所有用户的共同知识，且在每个时间片来临时更新。P 值越大，用户提交同样计算量和预期完成时间的任务将会付出更大的费用。

综上，在服务器对计算卸载任务 t_i 收取的费用 C_i 为：

$$C_i = \alpha p_i \cdot \beta^{d_i - p_i} \cdot \mathrm{e}^{\gamma P} \tag{8}$$

其中 $\alpha \geq 1$，$0 < \beta \leq 1$，$\gamma > 0$。

由于用户需要支付的费用与服务器的空闲计算资源、任务的计算量和任务的预期完成时间紧密相关，因此 C_i 的收取可以激励用户提交合理的计算卸载任务，保证了移动边缘计算框架的正常运行。

5. 实验与性能分析

本文将提出的最小化总滞后时间算法(MTD)与先入先出调度算法(FIFO)、随机排列调度算法(RANDOM)和分支定界调度算法(BBM)在总滞后时间、平均滞后时间和调度顺序计算时间三个方面进行比较。其中，总滞后时间和平均滞后时间分别表示边缘计算应用中用户任务的总时延和平均时延，边缘计算应用只有拥有合理的时延才能够正常运行。调度顺序计算时间表示算法的性能，调度算法只有在任务到达时快速确定调度顺序才能保证该算法在实际应用中的可靠性。

5.1. 实验设置

实验中边缘计算服务器为一台具有 8 核 4.0 GHz、32 GB 内存、1 TB 硬盘的移动边缘计算服务器。服务器每执行一次调度任务，每种算法执行 100 次，求平均值作为最终实验结果。用户的数量设置为 5~25 个，符合移动边缘计算小规模用户的适用场景。其中，每个用户提交一个具有不同的计算量和期望完成时间的计算卸载任务。

5.2. 结果分析

仿真实验时分别设置任务的计算量为 100~500，任务的期望完成时间为计算时间加上 0~100。

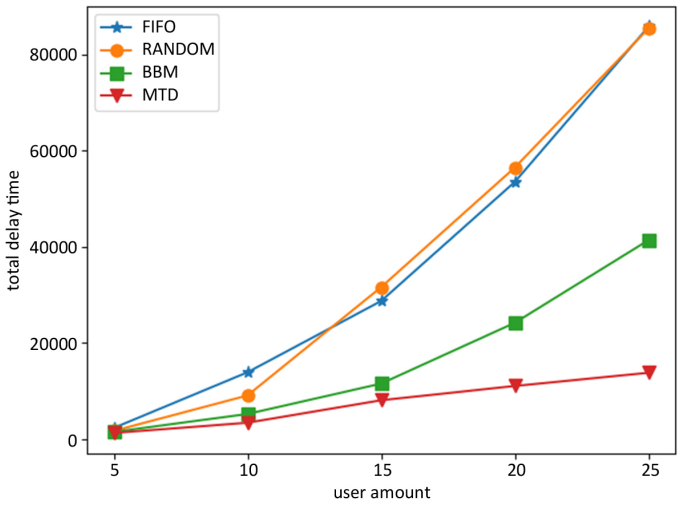

Figure 2. Comparison of total delay time
图 2. 总滞后时间比较

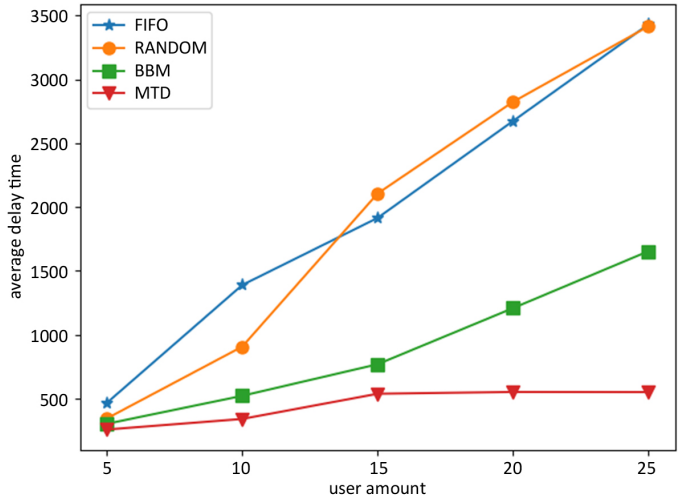

Figure 3. Comparison of average delay time
图 3. 平均滞后时间比较

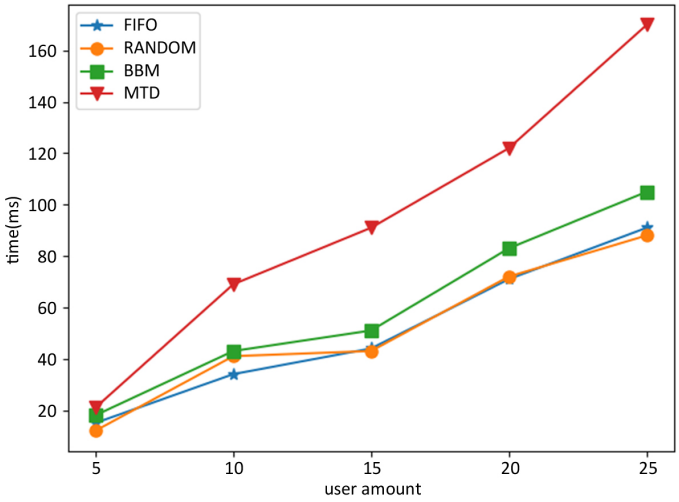

Figure 4. Comparison of scheduling order calculation time
图 4. 调度顺序计算时间比较

如图 2 所示，MTD 算法的总滞后时间最小，BBM 算法次之，但总滞后时间比 MTD 算法增加了 17%到 200%，FIFO 算法和 RANDOM 算法表现最差，总滞后时间较 BBM 算法增加了 53%到 160%。此外，MTD 算法在用户数量增加的情况下在总滞后时间上相对其他三个算法的优势越明显。

如图 3 所示，MTD 算法的平均滞后时间同样最小，BBM 算法次之，但平均滞后时间比 MTD 算法增加了 17%到 200%，FIFO 算法和 RANDOM 算法同样表现最差，平均滞后时间较 BBM 算法增加了 14%到 173%。与总滞后时间相同，MTD 算法在用户数量增加的情况下在平均滞后时间上相对其他三个算法的优势越明显。

如图 4 所示，MTD 算法在调度顺序计算时间上高于 BBM 算法、RANDOM 算法和 FIFO 算法。相比 BBM 算法，MTD 算法的调度顺序计算时间增加值在 50 ms 之内，相比效果更差的 FIFO 和 RANDOM 算法，MTD 算法的调度顺序计算时间增加值在 100 ms 之内。此外，由于 MTD 算法的计算复杂度为 $O\left(n^4\right)$，任务数量越大，调度顺序计算时间将增加的越明显。然而，移动边缘计算的适用场景为小规模用户，用户数量较小，MTD 算法的调度顺序计算时间并不会影响移动边缘计算框架的正常运行。

6. 结语

本文分析了传统移动边缘计算调度算法，发现现有算法没有考虑到服务器计算性能有限的情况，导致实际应用中任务滞后时间较长，因此提出了一种基于动态规划的最小总滞后时间的调度算法。该算法中用户在提交计算卸载任务的同时提交该任务的预期完成时间，服务器根据该任务的数据量和预期完成时间计算出任务计算的最优顺序以最小化总滞后时间。与此同时，本文还提出了一种激励机制，使得用户提交具有合理的计算量和预期完成时间的任务，同时在服务器计算资源不足时减少提交任务的数量和计算量。实验表明，该算法在性能接近传统调度算法的前提下，相比传统调度算法在总滞后时间和平均滞后时间上提高了 17%到 200%，具有非常重要的实际意义，可以用于工业物联网、智能家居、智慧医疗等应用。

在未来的工作中，将继续优化和验证本算法的性能，使得该算法能够适用于用户更多、计算任务更密集的移动边缘计算场景。

参考文献

[1] Pan, J. and Mcelhannon, J. (2017) Future Edge Cloud and Edge Computing for Internet of Things Applications. IEEE *Internet of Things Journal*, **5**, 439-449. https://doi.org/10.1109/JIOT.2017.2767608

[2] Shirazi, S. (2017) The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog from a Security and Resilience Perspective. *IEEE Journal on Selected Areas in Communications*, **35**, 2586-2595. https://doi.org/10.1109/JSAC.2017.2760478

[3] Filip, I., Postoaca, A., Stochitoiu, R., *et al*. (2019) Data Capsule: Representation of Heterogeneous Data in Cloud-Edge Computing. *IEEE Access*, **7**, 49558-49567. https://doi.org/10.1109/ACCESS.2019.2910584

[4] Wang, S., Zhang, X., Zhang, Y., *et al*. (2017) A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications. *IEEE Access*, **5**, 6757-6779. https://doi.org/10.1109/ACCESS.2017.2685434

[5] Mach, P. and Becvar, Z. (2017) Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys & Tutorials*, **19**, 1628-1656. https://doi.org/10.1109/COMST.2017.2682318

[6] Mao, Y., You, C., Zhang, J., *et al*. (2017) A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, **19**, 2322-2358. https://doi.org/10.1109/COMST.2017.2745201

[7] Abbas, N., Zhang, Y., Taherkordi, A., *et al*. (2017) Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*, **5**, 450-465. https://doi.org/10.1109/JIOT.2017.2750180

[8] Chen, S., Wen, H., Wu, J., *et al*. (2019) Internet of Things Based Smart Grids Supported by Intelligent Edge Computing. *IEEE Access*, **7**, 74089-74102. https://doi.org/10.1109/ACCESS.2019.2920488

[9] Feng, J., Liu, Z., Wu, C., *et al*. (2017) AVE: Autonomous Vehicular Edge Computing Framework with ACO-Based Scheduling. *IEEE Transactions on Vehicular Technology*, **66**, 10660-10675. https://doi.org/10.1109/TVT.2017.2714704

[10] Krishnan, P.R., Durga, P. and Srihari, R.E. (2018) IoT Based Smart Edge for Global Health: Remote Monitoring with Severity Detection and Alerts Transmission. *IEEE Internet of Things Journal*, **6**, 2449-2462. https://doi.org/10.1109/JIOT.2018.2870068

[11] Qiu, X., Chen, W., Hong, Z., *et al*. (2019) Online Deep Reinforcement Learning for Computation Offloading in Blockchain-Empowered Mobile Edge Computing. *IEEE Transactions on Vehicular Technology*, **68**, 8050-8062. https://doi.org/10.1109/TVT.2019.2924015

[12] Yu, S., Langar, R., Fu, X., *et al*. (2018) Computation Offloading with Data Caching Enhancement for Mobile Edge Computing. *IEEE Transactions on Vehicular Technology*, **67**, 11098-11112. https://doi.org/10.1109/TVT.2018.2869144

[13] Hu, M., Zhuang, L., Wu, D., *et al*. (2019) Learning Driven Computation Offloading for Asymmetrically Informed Edge Computing. *IEEE Transactions on Parallel and Distributed Systems*, **30**, 1802-1815. https://doi.org/10.1109/TPDS.2019.2893925

[14] Zhang, T. (2017) Data Offloading in Mobile Edge Computing: A Coalition and Pricing Based Approach. *IEEE Access*, **6**, 2760-2767. https://doi.org/10.1109/ACCESS.2017.2785265

[15] Chen, M. and Hao, Y. (2018) Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE Journal on Selected Areas in Communications*, **36**, 587-597. https://doi.org/10.1109/JSAC.2018.2815360

[16] Li, S., Tao, Y., Qin, X., *et al*. (2019) Energy-Aware Mobile Edge Computation Offloading for IoT over Heterogenous Networks. *IEEE Access*, **7**, 13092-13105. https://doi.org/10.1109/ACCESS.2019.2893118

[17] Pinedo, M. (2000) Scheduling: Theory, Algorithms, and Systems. 2th Edition, Prentice Hall Inc., Englewood Cliffs.