



C-V2X 边缘服务器的动态负载均衡算法研究

林 峰^a, 段建岚^b, 李传伟^b, 蒋建春^c

(重庆邮电大学 a. 电子信息与网络工程研究院; b. 通信与信息工程学院; c. 自动化学院, 重庆 400065)

摘 要: 为更好地解决蜂窝车联网与移动边缘计算融合应用场景下边缘服务器资源负载分配不均、资源利用率较低等问题, 提出一种动态负载均衡算法。通过监测边缘服务器的实时运行状态动态更新负载指标权重, 准确评估边缘服务器的实际负载状态, 并结合边缘服务器集群的负载率阈值、均值和标准差, 实现任务的合理分配。实验结果表明, 与传统随机轮询算法和最小流量均衡算法相比, 该算法能够更好地提升边缘服务器集群的负载均衡度, 缩短任务完成时间。

关键词: 蜂窝车联网; 边缘计算; 负载均衡; 动态调整; 集群

开放科学(资源服务)标志码(OSID):



中文引用格式: 林峰, 段建岚, 李传伟, 等. C-V2X 边缘服务器的动态负载均衡算法研究[J]. 计算机工程, 2020, 46(12): 201-206, 221.

英文引用格式: LING Feng, DUAN Jianlan, LI Chuanwei, et al. Research on dynamic load balancing algorithm for C-V2X edge server[J]. Computer Engineering, 2020, 46(12): 201-206, 221.

Research on Dynamic Load Balancing Algorithm for C-V2X Edge Server

LING Feng^a, DUAN Jianlan^b, LI Chuanwei^b, JIANG Jianchun^c

(a. Institute of Electronic Information and Network Engineering; b. College of Communication and Information Engineering; c. College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

[Abstract] To smooth the load imbalancing across the edge servers and improve resource utilization in the scenarios of Cellular Vehicle to Everything (C-V2X) integrated with Mobile Edge Computing (MEC), this paper proposes a dynamic load balancing algorithm. The algorithm monitors the real-time running status of the edge servers to adjust the weight of each load index dynamically, so that the actual load status of edge servers can be accurately evaluated. Then the algorithm considers the threshold, mean value and the standard deviation of loads of the edge server cluster to perform reasonable task allocation. Experimental results show that compared with the traditional random polling algorithm and minimum traffic balancing algorithm, the proposed algorithm can better smooth the load imbalancing in the edge server cluster, and reduce the time spent for tasks.

[Key words] Cellular Vehicle to Everything (C-V2X); edge computing; load balancing; dynamic adjustment; cluster

DOI: 10.19678/j.issn.1000-3428.0056602

0 概述

随着车联网技术的快速发展, 蜂窝车联网 (Cellular Vehicle to Everything, C-V2X) 与移动边缘计算 (Mobile Edge Computing, MEC) 技术不断融合, C-V2X 可大幅度降低未来自动驾驶和车联网部署成本, 而 MEC 将计算存储与业务服务能力向网络边缘迁移^[1], 从而实现“人-车-路-云”协同交互。然而, 在道路上由于不同区域的车辆分布不均衡, 因此导致不

同区域 MEC 服务器的数据访问量各不相同。负载均衡的目的是将大量并发任务合理调度到各个服务器节点上进行计算, 从而避免服务器出现负载倾斜问题, 进而提高集群性能。在现有的负载均衡算法研究中, 根据调度策略的不同, 主要可以分为静态负载均衡算法和动态负载均衡算法两类^[2]。静态负载均衡算法是依据既定的策略来调度任务, 而不考虑当前服务器节点的负载状况, 如轮询算法、加权轮询算法等。动态负载均衡算法是以当前服务器节点负载状况为

基金项目: 国家科技重大专项 (2018ZX03001023-006); 重庆市教育委员会科学技术研究项目 (KJQN201801611)。

作者简介: 林 峰 (1978—), 男, 高级工程师、硕士, 主研方向为车联网、智能终端系统; 段建岚、李传伟, 硕士研究生; 蒋建春, 教授、博士。

收稿日期: 2019-11-14 修回日期: 2019-12-28 E-mail: linfeng@cqupt.edu.cn

参考,合理调度任务,如最小连接法、加权最小连接法等^[3]。在实际应用中,相较于静态负载均衡算法,动态负载均衡算法通常可以更好地反映服务器状态,因此其在任务调度方面具有更好的效果。为此,本文提出一种动态负载均衡算法,其充分考虑各边缘服务器自身负载状态,动态更新负载指标权重,并根据服务器性能合理分配任务。

1 相关工作

在云计算系统中,文献[4]提出一种基于最小流量的自适应调度算法,通过网络流量状态参数判定服务器负载状态并设置服务器管理阈值,从而将新请求调度到低负载的服务器节点上。该算法虽然可在一定程度上减轻服务器节点负载,但由于其只考虑了网络流量,未考虑服务器 CPU 利用率、磁盘读写速率等其他性能指标,因此不能全面反映服务器节点的实际负载状态。文献[5]将鲸鱼算法与人工蜂群算法相结合,先将鲸鱼个体对应资源负载调度方案,利用鲸鱼算法对其进行种群初始化,并引入淘汰机制和竞争机制提高算法局部搜索能力,再将鲸鱼个体模拟成蜂群个体,利用人工蜂群算法对其进行优化提高算法全局搜索能力。文献[6]研究发现蚁群算法在任务调度方面具有收敛速度慢及资源调度不均衡等问题,并针对这些问题进行改进,通过计算负载指标权重加快算法收敛速度,提高虚拟机的负载均衡度。

在 Web 服务器集群系统中,文献[7]提出一种改进型轮询负载均衡策略,利用系统状态信息优化调度结果,根据循环列表及上一次选定服务器的指针进行调度。文献[8]研究基于神经网络反馈机制的负载均衡算法,利用神经网络的反馈机制获取负载状态,并将其作为加权最小连接数算法的输入得到最佳可分配负载权值,从而实现任务的合理分配。文献[9]研究一种基于残差的模糊自适应算法,具有较高的算法准确性。文献[10]在萤火虫算法的基础上引入混沌算法,解决了萤火虫算法收敛速度慢、易陷入局部最优的问题。

在软件自定义网络中,文献[11]研究基于链路实时状态的负载均衡策略,通过链路监测数据来选择低负载链路,从而解决链路拥塞问题。文献[12]针对节点负载信息获取困难、流量导向方式复杂等问题,提出一种负载均衡机制,利用 SNMP 协议和 OpenFlow 协议监测服务器实时状态,通过加权计算选出最优服务器,并采用最优转发路径算法进行流量调度,从而提高集群效率。文献[13]基于遗传算法和蚁群算法提出融合算法,提高了软件自定义网络的综合性能。

在边缘计算网络中,文献[14]研究了无共享、随机共享和最小负载共享 3 种负载均衡方案,通过比较发现最小负载共享方案最适用于服务器之间的协作,可实现集群的负载均衡。文献[15]提出一种数据中心协作的资源共享方案,规定每个数据中心都使用缓冲区存储服务请求以供本地执行,当缓冲区已满时将请求迁移至相邻数据中心,如果该数据中心的当前队列长度低于给定阈值,则接受该请求。通过该方式可以有效缓解数据中心的阻塞状态及减少任务执行时延。文献[16]针对边缘计算复杂多变的环境,提出一种基于深度强化学习的智能资源分配方案,可自适应分配计算资源,减少平均服务时间。文献[17]提出一种边缘计算任务调度模型,将等待时间最小化问题转换为整体规划问题,并通过动态规划实现最优调度。文献[18]提出一种改进型混沌蝙蝠群算法,在蝙蝠算法的基础上引入混沌因素和二阶振荡机制来加快动态参数更新,从而提升算法收敛速度,并通过深度学习快速预测调度结果。

2 C-V2X 边缘服务器负载均衡算法

对于 C-V2X 边缘服务器负载均衡的研究,多数算法着重于探究任务分配策略,而忽视了服务器的任务分配时间以及如何准确判断服务器运行状态等问题。这些问题对后续任务分配具有较大影响,容易导致集群负载均衡度及资源利用率低等问题,而车联网环境中存在边缘服务器负载差异大、负载状态随时间变化、集群计算资源分配不均衡等问题,因此本文提出一种基于动态负载指标权重的负载均衡算法,其充分考虑服务器当前负载状态,并动态调整负载指标权重,从而提升集群负载均衡度。

2.1 算法应用场景

由于从现有基站和终端上无法直接获得 C-V2X 安全应用所需的实时计算、存储、控制、加速等资源且不具备高效计算能力,而从云平台上获取这些资源并进行计算所需要的时延会急剧增加,无法达到 V2X 应用的时延和带宽要求,因此将 V2X 应用在靠近车辆的位置部署,使得 V2X 应用中产生的巨大流量和网络负担可在本地卸载,从而减少信息传输时延及网络带宽消耗。

图 1 显示了 C-V2X 与 MEC 融合应用场景,考虑道路在自由流状态下具有不间断交通,沿路设有演进型基站(eNodeB),每个 eNodeB 为其传输范围内的车辆提供无线接入服务,且每辆车在其范围内只能与一个 eNodeB 通信,eNodeB 之间通过 X2 接口通信。由于每个 eNodeB 的无线电功率不同以及无线环境多样,因此这些 eNodeB 的无线覆盖区域可能不同^[19],并且每个 eNodeB 都配备一个资源有限

的 MEC 服务器为卸载的任务提供计算服务^[20],每个 MEC 服务器都可与区域内的调度中心通信,MEC 服务器 id 与其对应的 eNodeB id 相匹配。调度中心负责周期性记录各个服务器负载状态,并为发起任务转移请求的服务器制定任务转移策略。

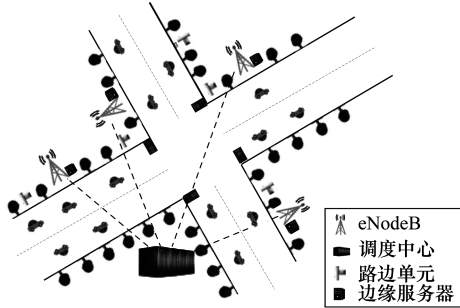


图1 C-V2X与MEC融合应用场景

Fig. 1 Integration application scenario of C-V2X and MEC

2.2 相关参数定义

MEC 服务器负载均衡的首要问题是评估当前服务器状态,而评估指标的不同会对负载状态的判断结果产生较大影响。本文算法充分考虑 MEC 服务器运行状态,综合处理器利用率、磁盘读写速率、内存使用率、带宽占用率 4 个方面对服务器状态进行评估,分别反映了服务器 CPU 的繁忙程度、数据操作的吞吐量、系统运行状态及接收数据量。

考虑道路范围内由 n 个 MEC 服务器组成一个集群,记为 $S = (S_1, S_2, \dots, S_n)$ 。第 i 个服务器为 S_i ,其性能为 $C^{S_i} = (C_{cpu}^{S_i}, C_{i/o}^{S_i}, C_{mem}^{S_i}, C_{band}^{S_i})$,其中 $C_{cpu}^{S_i}$ 、 $C_{i/o}^{S_i}$ 、 $C_{mem}^{S_i}$ 、 $C_{band}^{S_i}$ 分别表示服务器 i 最大能承载的处理器利用率、磁盘读写速率、内存使用率、带宽占用率,通常情况下 $C_{cpu}^{S_i}$ 、 $C_{i/o}^{S_i}$ 、 $C_{mem}^{S_i}$ 、 $C_{band}^{S_i}$ 为定值。

服务器 i 的负载状态为 $L^{S_i} = (L_{cpu}^{S_i}, L_{i/o}^{S_i}, L_{mem}^{S_i}, L_{band}^{S_i})$,其中 $L_{cpu}^{S_i}$ 、 $L_{i/o}^{S_i}$ 、 $L_{mem}^{S_i}$ 、 $L_{band}^{S_i}$ 分别表示服务器 i 的当前处理器利用率、磁盘读写速率、内存使用率和带宽占用率。

服务器 i 的负载率为:

$$F_{S_i}^{load} = \alpha_{cpu} R_{cpu}^{S_i} + \alpha_{i/o} R_{i/o}^{S_i} + \alpha_{mem} R_{mem}^{S_i} + \alpha_{band} R_{band}^{S_i} \quad (1)$$

$$\alpha_{cpu} + \alpha_{i/o} + \alpha_{mem} + \alpha_{band} = 1 \quad (2)$$

其中: α_{cpu} 、 $\alpha_{i/o}$ 、 α_{mem} 、 α_{band} 分别表示处理器利用率、磁盘读写速率、内存使用率、带宽占用率各指标所占的权重,其值越高表示对应资源所占比例越大,依赖性越强; $R_{cpu}^{S_i}$ 、 $R_{i/o}^{S_i}$ 、 $R_{mem}^{S_i}$ 、 $R_{band}^{S_i}$ 分别表示服务器 i 的处理器利用率、磁盘读写速率、内存使用率、带宽占用率 4 个指标的占比,计算方法如下:

$$R_{cpu}^{S_i} = L_{cpu}^{S_i} / C_{cpu}^{S_i} \quad (3)$$

$$R_{i/o}^{S_i} = L_{i/o}^{S_i} / C_{i/o}^{S_i} \quad (4)$$

$$R_{mem}^{S_i} = L_{mem}^{S_i} / C_{mem}^{S_i} \quad (5)$$

$$R_{band}^{S_i} = L_{band}^{S_i} / C_{band}^{S_i} \quad (6)$$

2.3 负载指标权值计算

传统负载均衡算法通常将负载指标权值设为定值,然而由于各个服务器性能存在差异且运行中各个指标的依赖情况不同,因此导致传统负载均衡算法无法准确反映服务器节点的负载状态^[21]。例如,服务器的 CPU 利用率已接近峰值,而其他指标均处于较低水平,该情况在传统负载均衡算法中并未达到高负载的状态,但实际情况中却已处于高负载的状态,此时若继续为其分配任务,则会严重影响服务器运行性能。又由于融合场景下车辆的移动性会导致边缘服务器负载状态随时间改变,因此本文提出一种动态更新负载指标权值的方法对其进行优化,从而更加准确地判断服务器实际负载状态。

由于在一个 eNodeB 覆盖范围内,车辆数量不会出现跃变情况,即短时间内车辆数量突然降低或升高,因此与 eNodeB 匹配的 MEC 服务器负载情况呈缓慢变化趋势,同时考虑到较高的负载指标应获取较高的权值,通过对比负载指标均值,判断负载指标权重是否需要提高或降低。当前负载指标均值的计算公式如下:

$$\bar{L}_{param}^{S_i} = \frac{\sum_{i=1}^n L_{param}^{S_i}}{n} \quad (7)$$

当 $L_{param}^{S_i} \geq \bar{L}_{param}^{S_i}$ 时,说明服务器当前负载状态对该负载指标的依赖性加强,此时应提高该指标对应权重。当 $L_{param}^{S_i} < \bar{L}_{param}^{S_i}$ 时,说明服务器当前负载状态对该负载指标的依赖性降低,此时应减少该指标对应权重。为更新权重,本文算法引入一个权重修正变量 ε 并更新权重 $W_{param a'}^{S_i}$:

$$\varepsilon = \left| 1 - \frac{L_{param}^{S_i}}{\bar{L}_{param}^{S_i}} \right| = \left| 1 - \frac{n L_{param}^{S_i}}{\sum_{i=1}^n L_{param}^{S_i}} \right| \quad (8)$$

$$W_{param a'}^{S_i} = \begin{cases} W_{param a}^{S_i} + \varepsilon, & L_{param}^{S_i} \geq \bar{L}_{param}^{S_i} \\ W_{param a}^{S_i} - \varepsilon, & L_{param}^{S_i} < \bar{L}_{param}^{S_i} \end{cases} \quad (9)$$

本文考虑处理器利用率、磁盘读写速率、内存使用率、带宽占用率 4 个负载指标,得到:

$$W_{sum'}^{S_i} = W_{cpu'}^{S_i} + W_{i/o'}^{S_i} + W_{mem'}^{S_i} + W_{band'}^{S_i} \quad (10)$$

结合式(2)、式(7)~式(10)得到负载指标的新权值为:

$$\alpha_{param'}^{S_i} = \frac{W_{param a'}^{S_i}}{W_{sum'}^{S_i}} \quad (11)$$

2.4 服务器负载判断

根据服务器 i 的负载率,本文考虑设置高负载率的阈值 \bar{F}_{high}^{load} 与低负载率的阈值 \bar{F}_{low}^{load} , \bar{F}_{high}^{load} 和 \bar{F}_{low}^{load} 的值可根据服务器实际情况设置。设 G_{S_i} 表示服务器 i 的负载状态:

$$Gs_i = \begin{cases} 0, & F_{S_i}^{\text{load}} \leq \bar{F}_{\text{low}}^{\text{load}} \\ 1, & \bar{F}_{\text{low}}^{\text{load}} < F_{S_i}^{\text{load}} < \bar{F}_{\text{high}}^{\text{load}} \\ 2, & F_{S_i}^{\text{load}} \geq \bar{F}_{\text{high}}^{\text{load}} \end{cases} \quad (12)$$

其中:当 $F_{S_i}^{\text{load}} \geq \bar{F}_{\text{high}}^{\text{load}}$ 时, $Gs_i = 2$ 说明服务器处于高负载状态,此时若有计算任务接入,应将计算任务转移至其他低负载的服务器上;当 $\bar{F}_{\text{low}}^{\text{load}} < F_{S_i}^{\text{load}} < \bar{F}_{\text{high}}^{\text{load}}$ 时, $Gs_i = 1$ 说明当前服务器处于正常负载状态,此时可考虑接收转移的任务;当 $F_{S_i}^{\text{load}} \leq \bar{F}_{\text{low}}^{\text{load}}$ 时, $Gs_i = 0$ 说明当前服务器处于低负载状态,此时可优先考虑接收转移的计算任务。

2.5 任务转移

当服务器处于高负载状态时,服务器将发起任务转移请求,调度中心接收请求后会该服务器排队中的任务转移至低负载状态的服务器上,从而提升整个集群的负载均衡度。

设 $F_{\text{centre}} = \{F_{S_1}^{\text{load}}, F_{S_2}^{\text{load}}, \dots, F_{S_n}^{\text{load}}\}$ 为调度中心接收到的服务器负载率集合, \bar{F}_{centre} 表示集群负载率均值,计算公式为:

$$\bar{F}_{\text{centre}} = \frac{1}{n} \sum_{i=1}^n F_{S_i}^{\text{load}} \quad (13)$$

设 σ_F 为集群负载率标准差,负载率标准差体现了集群负载率的变化情况,负载率标准差越低,负载率变化越小,反之越大,其计算公式为:

$$\sigma_F = \sqrt{\frac{1}{n} \sum_{i=1}^n (F_{S_i}^{\text{load}} - \bar{F}_{\text{centre}})^2} \quad (14)$$

由式(14)可知,若要提升集群整体负载均衡度,则需降低 σ_F 值。因此,合理迁移高负载状态服务器任务队列中的任务将成为关键。

假设待分配的任务数为 n ,集群的服务器数为 m ,结合服务器当前负载状态,考虑到负载率越低的服务器具有越高的分配概率,且可接收迁移的任务量也越多。设 P_{ij} 表示任务 i 分配到服务器 j 上的概率,计算公式为:

$$P_{ij} = \begin{cases} \frac{\bar{F}_{\text{centre}}}{F_{S_j}^{\text{load}} \sum_{k \in \text{set}} \frac{\bar{F}_{\text{centre}}}{F_{S_k}^{\text{load}}}}, & j \in \text{set} \\ 0, & j \notin \text{set} \end{cases} \quad (15)$$

其中, set 表示服务器集群中满足 $Gs_i \neq 2$ 时的服务器集合。

根据以上计算可知,当有任务需要进行迁移时,调度中心都会对比各个服务器负载状态并计算迁移概率,然后用轮盘赌法决定任务迁移到哪个服务器上,概率值即赌盘的扇区面积,概率值越大,

扇区面积越大,被选中的几率越大。因此,通过合理分配迁移任务可提升整个服务器集群的负载均衡度。

2.6 算法流程

本文算法主要分为:1)本地服务器计算各负载指标权值、服务器性能和服务器负载率,并将结果周期性上传至调度中心;2)调度中心收集各服务器信息并存表,当服务器发起任务迁移请求时,调度中心计算任务迁移概率并用轮盘赌法决定任务迁移到哪个服务器上。若无服务器可接收迁移任务,即集群中的服务器都处于高负载状态,则抛弃任务。

动态负载均衡算法流程如图2所示,具体步骤如下:

步骤1 对各个 MEC 服务器进行参数初始化,包括服务器性能参数、指标权值设置等。

步骤2 各个 MEC 服务器按照一定时间间隔 T ,动态更新指标权值、计算服务器性能和负载率以及判断服务负载状态并将相关信息上传至调度中心。

步骤3 调度中心接收各个 MEC 服务器上传的信息,并对这些信息进行存表、更新等操作。

步骤4 若调度中心接收到服务器的任务迁移请求,则对服务器进行分类,选出可接收迁移任务的服务器,并计算各个服务器的转移概率。

步骤5 调度中心根据转移概率设计任务迁移方案,eNodeB接收调度中心调度安排,将任务通过X2接口传输至eNodeB。当任务计算完成后,将计算结果进行回传。

步骤6 重复步骤4和步骤5,等待下一个周期,从步骤2开始执行并进行周期性循环。

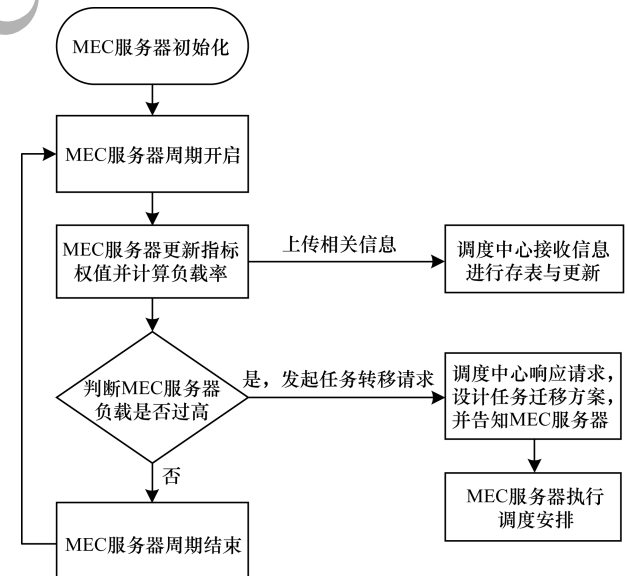


图2 动态负载均衡算法流程

Fig. 2 Procedure of dynamic load balancing algorithm

3 实验验证

为验证本文算法性能,选择传统随机轮询算法和文献[4]最小流量均衡算法做对比实验。实验环境使用 Cloudsim 仿真软件进行模拟,通过 Datacenter 类、Vm 类和 Cloudlet 类设置数据中心、模拟服务器和产生计算任务,利用 Origin9 对仿真结果进行图形化处理。

3.1 参数设置

通过 Cloudsim 仿真软件进行模拟,设置虚拟机参数和任务参数如表 1、表 2 所示。

表 1 虚拟机参数设置

Table 1 Virtual machine parameter setting

参数名	参数值
CPU 数量	4
处理器计算能力/MIPS	400
内存使用量/GB	2
带宽占用率/(Mb · s ⁻¹)	300
磁盘容量/GB	2

表 2 任务参数设置

Table 2 Task parameter setting

参数名	参数值
任务大小/MB	20 ~ 200
任务输出大小/MB	15 ~ 150
任务长度/MI	1 000 ~ 5 000

3.2 评价指标

在实验过程中,为充分模拟现实环境,本文将从以下两方面分析算法性能:

1) 将负载率均值和负载率标准差作为参考指标,其中:负载率均值体现了整个集群的负载状态,其值越低,说明集群整体负载越低、负载状态越好;负载率标准差体现了整个集群的负载稳定性,其值越低,说明集群负载越均衡、稳定性越强。

2) 将任务完成时间作为算法性能指标,任务完成时间越短,算法性能越好。

3.3 结果分析

本文在同等条件下,对随机轮询算法、最小流量均衡算法以及本文算法进行仿真验证对比,仿真结果如图 3 ~ 图 5 所示。由图 3 可以看出,本文动态负载均衡算法的负载率均值最低,其次是随机轮询算法,负载率均值最大为最小流量均衡算法。本文算法和最小流量均衡算法均在任务数达到 300 后,触发大规模任务调度,从而使负载率均值上升速度变缓,但本文算法负载率均值上升速度约为最小流量均衡算法的一半,表明其对任务的分配更加合理。由图 4 可以看出,本文算法的负载率标准差比另外两种算法都低,且在任务数达到 250 后基本呈稳定状态,表明其可以更好地提升集群负载均衡度。由图 5 可以看出,在任务数较少时,3 种算法的任务完

成时间差距不大,集群负载基本都处于低负载状态。随着任务数的增加,3 种算法在任务完成时间上的差距越来越大,表明其对集群负载的影响各不相同,且本文算法的任务完成时间一直处于最少状态,相较另外两种算法能更好地缩短用户完成任务的时间,从而提高用户体验。

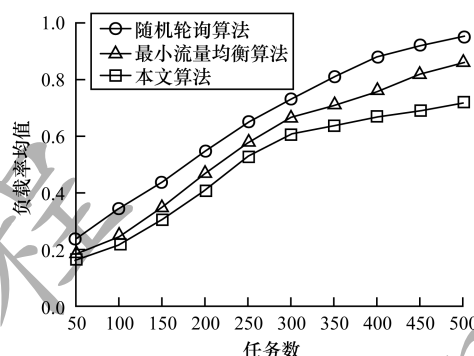


图 3 负载率均值对比

Fig. 3 Comparison of mean value of load ratio

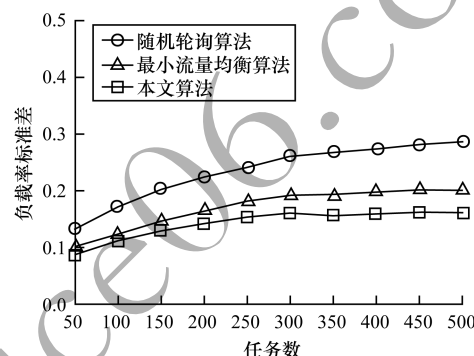


图 4 负载率标准差对比

Fig. 4 Comparison of standard deviation of load ratio

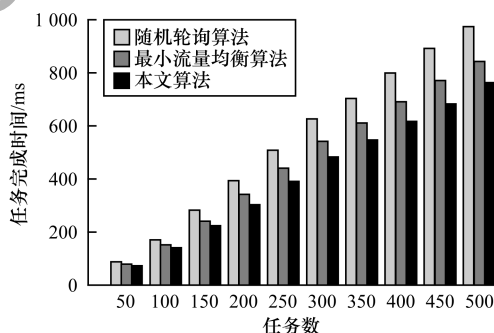


图 5 任务完成时间对比

Fig. 5 Comparison of task completion time

通过上述实验对比可知,本文算法相较随机轮询算法与最小流量均衡算法能够更好地适应 C-V2X 与 MEC 融合应用场景,提升边缘服务器集群负载均衡度,减少任务完成时间。其主要原因:相较随机轮询算法与最小流量均衡算法,本文算法考虑了处理器利用率、磁盘读写速率、内存使用率、带宽占用率这些对服务器负载有较大影响的因素,通过对所

有因素进行加权可以更准确地反映出当前服务器的负载状态。此外,本文算法同时考虑融合环境下任务数量与类型会随时间变化,导致边缘服务器负载对不同影响因素的依赖程度随时间而变化,因此本文通过边缘服务器负载状态合理判断边缘服务器的任务转移时间,从而提升集群负载均衡度。

4 结束语

在车联网环境中为减少端到端通信时延并提供更加可靠的服务,引入了边缘计算技术,然而在C-V2X与MEC融合应用场景中车辆分布不均容易导致边缘服务器负载不均、资源利用率低等问题。为此,本文提出一种动态负载均衡算法,通过充分考虑边缘服务器自身运行状态,动态调整指标权重并合理分配计算任务,从而提升集群负载均衡度。实验结果表明,与传统随机轮询算法和最小流量均衡算法相比,本文算法可以更好地分配计算资源,提高资源利用率,缩短任务完成时间。后续将对集群调度中心的任务分配机制进行研究,进一步减少任务分配时间并提升用户体验。

参考文献

- [1] LI Zishu, XIE Renchao, SUN Li, et al. A survey of mobile edge computing [J]. Telecommunications Science, 2018, 34(1): 87-10. (in Chinese)
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): 87-10.
- [2] ASLAM S, SHAH M A. Load balancing algorithms in cloud computing: a survey of modern techniques [C]// Proceedings of 2015 National Software Engineering Conference. Washington D. C., USA: IEEE Press, 2015: 30-35.
- [3] YIN Youlei. Research and application of MQTT server load balancing technology [D]. Shanghai: Shanghai Normal University, 2018. (in Chinese)
尹友磊. MQTT 服务器负载均衡技术的研究与应用[D]. 上海: 上海师范大学, 2018.
- [4] KANG Lu, TING Xing. Application of adaptive load balancing algorithm based on minimum traffic in cloud computing architecture [C]// Proceedings of 2015 International Conference on Logistics, Informatics and Service Sciences. Washington D. C., USA: IEEE Press, 2015: 1-8.
- [5] GENG Qiang, HUANG Xueqin. Research on an adaptive load balancing algorithm based on cloud computing [J]. Bulletin of Science and Technology, 2019, 35(8): 129-133. (in Chinese)
耿强, 黄雪琴. 云计算下的一种自适应负载均衡算法的研究[J]. 科技通报, 2019, 35(8): 129-133.
- [6] QUAN Li, FU Ming. Research on task scheduling optimization strategy in cloud computing [J]. Computer Engineering, 2018, 44(8): 14-18. (in Chinese)
全力, 傅明. 云计算中任务调度优化策略的研究[J]. 计算机工程, 2018, 44(8): 14-18.
- [7] XU Zongyu, WANG Xingxuan. A modified round-robin load-balancing algorithm for cluster-based Web servers [C]// Proceedings of the 33rd Chinese Control Conference. Washington D. C., USA: IEEE Press, 2014: 28-33.
- [8] WANG Peng, HUANG Hongqiong. An improved load balancing algorithm based on neural network feedback mechanism [J]. Modern Computer (Professional Edition), 2018, 612(12): 5-11. (in Chinese)
王鹏, 黄洪琼. 基于神经网络反馈机制的改进型负载均衡算法[J]. 现代计算机(专业版), 2018, 612(12): 5-11.
- [9] TENG Honglei, WANG Yuegang, YANG Bo, et al. An improved algorithm for nonlinear target tracking based on residual fuzzy matching [J]. Piezoelectrics & Acousto-optics, 2019, 41(2): 311-317, 322. (in Chinese)
滕红磊, 王跃钢, 杨波, 等. 一种残差模糊匹配的非线性目标跟踪改进算法[J]. 压电与声光, 2019, 41(2): 311-317, 322.
- [10] YANG Dan, LI Chaofeng, YANG Jian. Computing resource scheduling based on improving chaos firefly algorithm [J]. Computer Engineering, 2015, 41(2): 17-20, 25. (in Chinese)
杨单, 李超峰, 杨健. 基于改进混沌萤火虫算法的云计算资源调度[J]. 计算机工程, 2015, 41(2): 17-20, 25.
- [11] CAO Bin, LIU Xie, SUN Qi. Dynamically adaptive load balancing strategy under the software defined network structure [J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2015, 27(4): 460-465, 548. (in Chinese)
曹宾, 刘颢, 孙奇. 软件定义网络架构下的动态自适应负载均衡策略研究[J]. 重庆邮电大学学报(自然科学版), 2015, 27(4): 460-465, 548.
- [12] YU Tianfang, RUI Lanlan, QIU Xuesong. Research on SDN-based load balancing technology of server cluster [J]. Journal of Electronics and Information Technology, 2018, 40(12): 3028-3035. (in Chinese)
于天放, 芮兰兰, 邱雪松. 基于软件定义网络的服务器集群负载均衡技术研究[J]. 电子与信息学报, 2018, 40(12): 3028-3035.
- [13] XUE H, KIM K T, YOUN H Y. Dynamic load balancing of software-defined networking based on genetic-ant colony optimization [J]. Sensors, 2019, 19(2): 311-318.
- [14] LIU L, CHAN S, HAN G, et al. Performance modelling of representative load sharing schemes for clustered servers in multi-access edge computing [J]. IEEE Internet of Things Journal, 2019, 6(3): 4880-4888.
- [15] BERALDI R, MTIBAA A, ALNUWEIRI H. Cooperative load balancing scheme for edge computing resources [C]// Proceedings of the 2nd International Conference on Fog and Mobile Edge Computing. Washington D. C., USA: IEEE Press, 2017: 94-100.
- [16] WANG Jiadai, ZHAO Lei, LIU Jiajia et al. Smart resource allocation for mobile edge computing: a deep reinforcement learning approach [EB/OL]. [2019-10-10]. <https://www.10.1109/TETC.2019.2902661>.

(上接第 206 页)

- [17] LUO Jie, DENG Xiaoheng, ZHANG Honggang, et al. Ultra-low latency service provision in edge computing [C] // Proceedings of 2018 IEEE International Conference on Communications. Washington D. C., USA: IEEE Press, 2018: 1-6.
- [18] JIAN Chengfeng, CHEN Jiawei, PING Jing, et al. An improved chaotic bat swarm scheduling learning model on edge computing [J]. IEEE Access, 2019, 7: 58602-58610.
- [19] AGIWAL M, ROY A, SAXENA N. Next generation 5G wireless networks; a comprehensive survey [J]. IEEE Communications Surveys and Tutorials, 2016, 18(3): 1617-1655.
- [20] MAO Yuyi, YOU Changsheng, ZHANG Jun, et al. A survey on mobile edge computing: the communication perspective [J]. IEEE Communications Surveys and Tutorials, 2017, 19(4): 2322-2358.
- [21] YI Chang, ZHANG Xiuguo, CAO Wei. Dynamic weight based load balancing for microservice cluster [C] // Proceedings of the 2nd International Conference on Computer Science and Application Engineering. Washington D. C., USA: ACM Press, 2018: 2-8.

编辑 陆燕菲