阳柳, 林晓勇, 毛雅淇. 基于边缘计算的基站端计算资源调度策略研究[J]. 智能计算机与应用,2024,14(11):1-8. DOI:10.20169/j.issn.2095-2163.241101

基于边缘计算的基站端计算资源调度策略研究

阳 柳[1], 林晓勇[1,2], 毛雅淇[3]

(1 南京邮电大学 通信与信息工程学院, 南京 210003; 2 南京邮电大学 通达学院, 江苏 扬州 225127;
3 西北工业大学 电子信息学院, 西安 710072)

摘 要: 传统的移动边缘计算是将计算任务卸载到移动边缘服务器,特殊场景下如边缘网络带宽受限、基站覆盖能力变弱、甚至灾害或战争情况下导致基站与边缘服务器通信中断,复杂计算任务无法提交到边缘甚至云端,基站上行传输强行被切断。结合移动自组织网络以及雾计算技术,给出基站端计算技术,基站通过下行传输将计算任务转发给小区内的移动智能终端,利用资源调度策略实现完成任务计算。资源调度策略中,针对基站端计算网络下的终端管理,提出了一种通过衡量并发传输时延的可获得最大计算力的聚簇算法,利用终端可用计算力与簇总计算力的关系,比例分配计算任务和带宽,最大效能地提高频谱利用率,从而快速完成原有计算任务。仿真结果表明,本文算法在基站端计算网络中能有效减少系统时延,高效完成计算任务。

关键词: 可用计算力; 基站端计算; 聚簇

中图分类号: TP391 **文献标志码**: A **文章编号**: 2095-2163(2024)11-0001-08

Research on resource scheduling strategy of base station terminal computing based on edge computing

YANG Liu[1], LIN Xiaoyong[1,2], MAO Yaqi[3]

(1 School of Communications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China; 2 College of Tongda, Nanjing University of Posts and Telecommunications, Yangzhou 225127, Jiangsu, China; 3 School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: The traditional mobile edge computing offloads computing tasks to mobile edge servers. In special scenarios such as limited edge network bandwidth, weakened base station coverage, and even disaster or war situations, communication between base stations and edge servers is interrupted. Complex computing tasks cannot be submitted to the edge or even the cloud, and the uplink transmission of base stations is forcibly cut off. Combining mobile ad hoc networks and fog computing technology, a base station terminal computing technology is proposed. The base station forwards computing tasks to mobile intelligent terminals in the community through downlink transmission, and uses resource scheduling strategies to complete task calculations. In the resource scheduling strategy, a clustering algorithm is proposed for terminal management in the base station computing network, which can obtain the maximum computation by measuring the concurrent transmission delay. By utilizing the relationship between the available computation of the terminal and the total computation of the cluster, computing tasks and bandwidth are proportionally allocated, maximizing the efficiency of spectrum utilization and quickly completing the original computing tasks. The simulation results show that the algorithm proposed in this paper can effectively reduce system latency and efficiently complete computational tasks in the base station computing network.

Key words: available computation; base station terminal computing; clustering

0 引 言

目前基于云计算(Cloud Computing, CC)[1]、边缘计算(Edge Computing, EC)[2]和雾计算(Fog Computing, FC)[3-5]的研究较为成熟,伴随着物联网(Internet of Things, IoT)演进,万物互联推动了边缘

基金项目: 国家自然科学基金(61801240)。

作者简介: 阳 柳(1999—),女,硕士研究生,主要研究方向:边缘计算; 毛雅淇(2000—),女,博士研究生,主要研究方向:无线通信。

通信作者: 林晓勇(1974—),男,博士,副教授,主要研究方向:信息与通信。Email:njuptlxy@163.com。

收稿日期: 2023-06-06

计算技术的发展,5G 基站技术加速边缘计算受到学界关注[2]。移动边缘计算(Mobile Edge Computing,MEC)将服务器下沉至数据源一侧为用户提供计算和网络等资源,以完成用户的各种业务请求[6-8],其具有更低的时延和能耗。当各种突发情况下导致边缘服务器损坏,基站所接受的计算任务将无法通过云计算和边缘计算进行处理。基于此,提出基站端计算(Base station Terminal Computing,BTC)技术,基站端计算是基站通过下行传输将计算任务转发给小区内的移动智能终端,终端节点利用移动自组织网络(Mobile Ad hoc Network,MANET)互相连接,利用聚簇对终端节点进行管理,选举簇首与基站进行任务传递,利用终端节点的可用计算力组建计算网络,实现计算力共享,并合理分配计算任务和信道带宽,最终实现基站端计算资源调度。BTC 能完成边缘服务器损坏下基站需要解决的计算任务。

典型的聚簇算法有最小 ID 算法(Least IDentity algorithm,LID)[9]、最大连接度算法[10]、最低移动性算法和加权聚簇算法(Weighted Clustering Algorithm,WCA)[11-16],这些典型算法主要考虑簇的生存时间以及簇首选举的频繁性。目前,大多数文献通过自适应的加权聚簇算法实现簇首的选举,文献[17-18]基于自适应按需加权聚簇算法(Adaptive Weighted Clustering Algorithm,A-WCA),综合考虑了无线节点的移动性、节点度、剩余能量等因素来进行簇首的选举。文献[19]考虑节点与邻居的距离之和、节点度的偏差和节点的剩余能量,以此作为簇首选举的因素。文献[20]通过等角度聚簇算法(Equal Angle Clustering algorithm,EAC),通过基站到簇首的通信半径,得到系统时延最低时的最佳簇首位置,以此选定簇首。

综上所述,已有的聚簇方案大多都未考虑簇内节点计算力与系统带宽分配的耦合。目前虽有很多聚簇方案采用加权聚簇的方法来保证簇的生存时间,但都未考虑可用计算力对簇和系统的贡献。基于此,本文重点研究在聚簇实现节点管理的情况下,基于终端可用计算力的比例计算任务和带宽分配算法,实现资源分配,最大效能地提高频谱利用率。本文的主要贡献如下:

(1)在 BTC 中,基于簇中计算力最大化选举簇首,提出基站端计算中最大计算力聚簇算法(Maximum Computations Clustering algorithm,MCC)。

(2)在 BTC 中,提出基于终端可用计算力比例分配算法,利用该算法对计算任务和带宽的分配,有效地提升了传输速率,降低系统时延。

本文的其余部分的结构如下:第 1 节详细陈述了基站端计算的系统模型。第 2 节提出了基于簇中计算力最大化的聚簇算法和资源分配的方法。第 3 节给出了仿真结果和分析。最后,第 4 节总结全文。

1 系统模型

1.1 系统建模

系统模型如图 1 所示。BTC 的应用场景设想在一些特殊情况下,比如在战争情况下、MEC 服务器被摧毁,或在自然灾害下、基站性能弱覆盖(弱基站),只能接收发送任务,无法参与任务计算,为了保证系统内计算任务得以解决,基站可利用小区内终端节点的可用计算力完成计算任务,维持整个系统的稳定。基站覆盖范围内终端节点可用集合 $G=\{1,2,\cdots,Num\}$ 表示。研究可知,BTC 过程包括 3 个阶段。第一阶段为节点聚簇并选举簇首,第二阶段为基站到簇首的计算任务传输,第三阶段为簇首根据比例分配算法给簇内节点下发计算任务和分配信道带宽,并完成任务计算。簇内节点可用集合 $K=\{1,2,\cdots,N\}$ 表示,簇内通信采用正交频分复用(Coded Orthogonal Frequency Division Multiplexing,COFDM)技术,簇首可并行发送计算任务给簇内节点,簇内节点之间不通信。

BTC 中各节点基于 MANET 消息协议维持路由表,可得邻居节点的位置信息和可用计算力;节点可与基站保持基本的连接,基站可以得到各节点的基本信息,定义为 $I=\{id,f,(x,y),will\}$,其中 id 为节点的编号,f 为用户终端可用计算力,(x,y) 为当前时刻的地理位置,$will$ 为内节点的通信意愿。基站端计算利用 MANET 网络与用户进行连接,建立基本的相互关系,再通过 MCC 算法实现终端节点的聚簇管理,形成多个小型的计算网络中心,完成基站下发的计算任务。

1.2 问题分析

BTC 第一阶段在计算任务传输前已完成,不存在时延消耗;BTC 第二阶段的传输时延取决于簇首的信道质量,基站到各簇首的传输时延为 $t_{1BS,cu}$,计算过程如下所示:

$$t_{1BS,cu}=\frac{C}{R_{cu}}=\frac{C}{B\log_2\left(1+\frac{P_{BS,cu}d_{BS,cu}^{(-l)}}{n_oB}\right)} \quad (1)$$

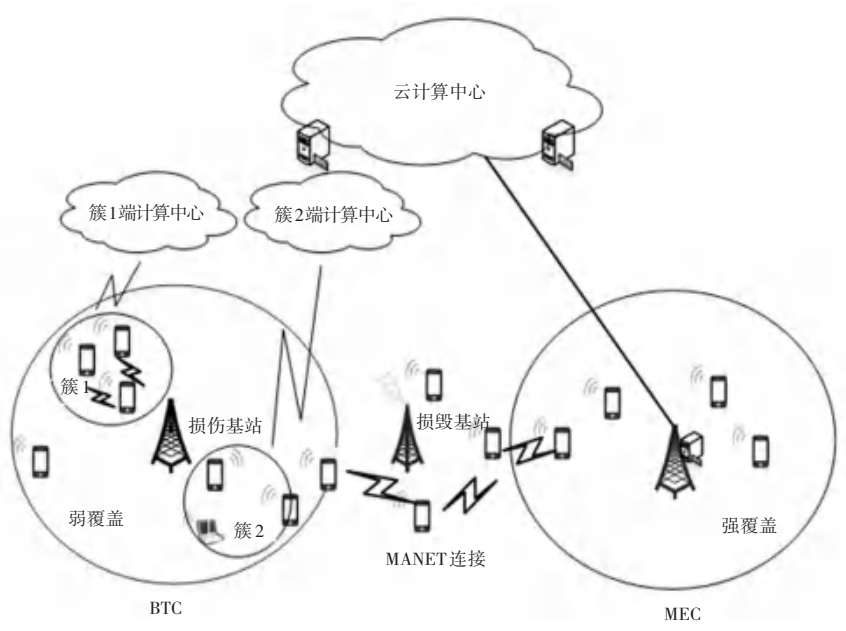

图 1　系统模型

Fig. 1　Model of the system

其中，C 表示基站传输的计算任务大小；B 表示基站到簇首的传输带宽；R_{cu} 表示基站到簇首的传输速率；$P_{BS,cu}$ 表示基站的发射功率；$d_{BS,cu}$ 表示簇首到基站的物理距离；l 表示路径损耗因子；n_0 表示噪声功率谱密度。

BTC 第三阶段时延包括簇首到簇内节点的传输时延和簇内节点计算时延，第三阶段传输时延取决于信道质量最差的节点，计算时延取决于计算力最小的节点。

簇首到簇内节点 i 的传输时延为 $t_{2cu,i}$，具体公式如下：

$$t_{2cu,i}=\frac{C_i}{R_{cu,i}}=\frac{C_i}{B_i\log_2\left(1+\dfrac{p_{cu,i}d_{cu,i}^{(-l)}}{n_oB_i}\right)} \tag{2}$$

其中，$R_{cu,i}$ 表示簇首到簇内节点 i 的传输速率；C_i 表示簇内节点 i 收到的计算任务；B_i 表示簇内节点 i 所分配的信道带宽；$p_{cu,i}$ 表示簇首到簇内节点 i 节点的发射功率；$d_{cu,i}$ 表示簇首到簇内节点 i 的物理距离。

簇内节点 i 完成计算任务所需的计算时延为 $t_{com,i}$，推得的公式为：

$$t_{com,i}=\frac{C_i\phi}{f_i} \tag{3}$$

其中，ϕ 表示 CPU 计算 1 bit 数据所需转数，f_i 表示簇内节点 i 的可用计算力。

节点完成计算任务后产生的计算结果很小甚至不用传输回基站，因此不考虑上行时延。完成计算任务所需时延为下行传输时延和计算时延。

第二阶段传输过程，第三阶段传输过程和第三阶段计算过程为串行过程，各簇之间与簇内之间的通信为并发过程，因此整个系统完成计算任务所需时延为第二阶段和第三阶段时延和的最大值，用 T 表示，如下所示：

$$T=\max(t_{1BS,cu}+t_{2cu,i}+t_{c,i})=\max\left\{\frac{C}{B\log_2\left(1+\dfrac{P_{BS}d_{BS,cu}^{(-l)}}{n_oB}\right)}+\frac{C_i}{B_i\log_2\left(1+\dfrac{p_{cu,i}d_{cu,i}^{(-l)}}{n_oB_i}\right)}+\frac{C_i\phi}{f_i}\right\} \tag{4}$$

基于上述分析，系统总时延与所选簇有很大关系，因此本文以系统时延为优化目标来选举簇首，保证所选簇能最快完成计算任务。系统模型见下式：

$$\min_{C_i,B_i}T=\min\max\left\{\frac{C}{B\log_2\left(1+\dfrac{P_{BS}d_{BS,cu}^{(-l)}}{n_oB}\right)}+\frac{C_i}{B_i\log_2\left(1+\dfrac{p_{cu,i}d_{cu,i}^{(-l)}}{n_oB_i}\right)}+\frac{C_i\phi}{f_i}\right\}$$

$$\text{C1}:\sum_{i=1}^{N}C_i=C;$$

$$\text{C2}:\sum_{i=1}^{N}B_i\leqslant B \tag{5}$$

其中，C1 为计算任务分配约束，保证完成所有计算任务；C2 为带宽分配约束，要求所分配的带宽和要小于系统总带宽。

基于上述分析，要保证簇首选举后，簇内节点能在最快时间内完成计算任务，为了简便上述问题，本文通过讨论单个簇的形成介绍基站端计算。

2 资源调度策略

计算任务完成所需时延包括传输时延和计算时延，不同的聚簇结果，影响簇中总计算力，从而影响计算时延，簇中节点所分配的计算任务大小和带宽影响着传输时延。因此资源调度策略的目的是通过结合节点计算力、计算任务和信道带宽分配实现系统最小时延。

资源调度策略流程如图 2 所示。BTC 中节点根据自身通信意愿选择是否加入 MANET 网络，基于 MCC 算法选择簇首，并将计算任务传输给该节点，簇首通过比例分配算法给各簇内节点分配带宽和计算任务，最终实现最低时延花销。

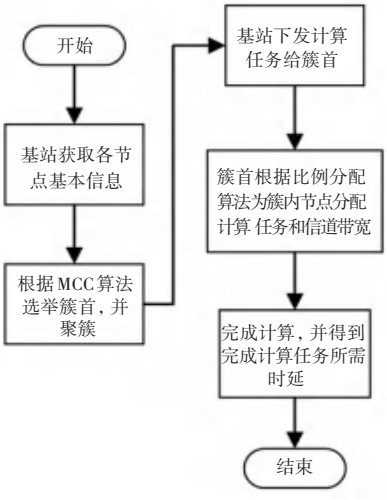

图 2 资源调度策略流程图

Fig. 2 Flowchart of resource scheduling strategy

2.1 MCC 算法

传统的簇首选举算法分割了簇首与整个簇之间的关系，仅考虑簇首选举的单一条件进行聚簇，就无法保证所聚簇的整体性能。本文通过衡量不同节点作为簇首时簇的整体性能，以此选定最佳簇，并选举簇首。

2.1.1 节点入簇

由于节点存在自私性，并不是所有节点都愿意加入 BTC 网络参与计算，节点入簇的条件如下式所示：

$$\begin{cases} will(i)=1,\ i\in G & (l) \\ d_{cu,i}\leqslant r,\quad i\in G & (m) \end{cases} \tag{6}$$

节点意愿满足条件(l)表示愿意加入 BTC 网络，即聚簇时自愿入簇，反之则不愿入簇；条件(m)表示节点与簇首的物理距离满足最大通信距离要求，只有满足上述 2 个条件节点才能入簇。

2.1.2 簇首选举

系统总时延主要由计算时延决定，因此在选举簇首时需要保证计算时延最小化，这里用到的公式为：

$$\begin{aligned} &\frac{C_1}{f_1}=\frac{C_2}{f_2}=\cdots=\frac{C_i}{f_i}=\cdots=\frac{C_N}{f_N}=\tau && (a)\\ &\Rightarrow t_c=\tau\phi && (b)\\ &\Rightarrow C_i=f_i\tau && (c)\\ &\Rightarrow C=\sum_{i=1}^{N}f_i\tau=F\tau && (d)\\ &\Rightarrow \tau=\frac{C}{F} && (e)\\ &\Rightarrow C_i=\tau f_i && (f) \end{aligned} \tag{7}$$

步骤(a)表示通过计算任务分配，使各节点计算时延相同，节约计算时延成本，设 τ 为时延因子；步骤(b)表示了时延因子与计算时延的关系；步骤(c)表示根据节点可用计算力进行计算任务分配；步骤(d)表示计算任务 C 与簇内合力 F 之间的关系；步骤(e)表示时延因子 τ 在固定计算任务 C 要求下，与簇内可用计算力和 F 有关。因此计算任务分配方案如步骤(f)所示。

根据式(7)的步骤(e)可知，实现计算时延最小化即要保证簇内计算合力最大化。

由于各节点的连接度以及与各邻居节点的距离存在差异，因此不同节点作为簇首时，完成同一任务所需时延也有差异。为得到最小时延，本文提出的基于最大化簇中计算力的簇首选举的聚簇算法，算法伪代码具体如下。

算法 1 MCC 算法

输入 r, Num;

输出 A, FF

1. 初始化相关参数，基本信息 I;
2. for $i = 1$: Num do
3. if will (i) == 1 then
4. 保存节点信息，集合 $M=\{1,2,\cdots,m\}$ 表示;
5. end if
6. end for

7. for $i = 1:m$ do

8. for $j = 1:m$ do

9. $D(i, j) \leftarrow M$ 中各节点之间的距离；

10. if $D(i, j) \leqslant r$ then

11. 保存节点信息，集合 $M1 = \{1,2,\cdots,m1\}$ 表示；

12. end if

13. end for

14. end for

15. for $i = 1:m1$ do

16. $F(i) \leftarrow$ 根据式(7)中步骤(d)得到各节点作为簇首时的总计算力；

17. end for

18. $K = \{1,2,\cdots,N\} \leftarrow$ 得到使总计算力最大簇内节点分布；

19. $FF = \{f_1, f_2, \cdots, f_N\} \leftarrow$ 得到簇中节点的可用计算力集合

2.2 计算资源分配算法

2.2.1 计算任务分配

由于计算时延远远大于传输时延，为了确保系统时延最小，首先需要保证簇内计算时延最小。根据式(3)可知，计算时延与簇内节点的可用计算力和其所需处理的计算任务大小有关。

由于计算过程是并发处理，最终计算时延值取簇中计算时延最大值，可通过计算任务分配，使簇内节点同时完成计算任务，即此时计算时延最小。设定各节点分配到的计算任务为 C_i。分析步骤见式(7)。

2.2.2 带宽分配

由式(2)可知，传输时延与簇内节点的计算任务分配和传输速率有关。传输速率与信道带宽和与簇首的距离有关。可得 BTC 第三阶段传输时延与信道带宽，计算任务与距离簇首的距离的函数关系，计算公式如下：

$$t_{2cu,i} = H(C_i, B_i) \tag{8}$$

根据式(7)中的步骤(f)可知每个节点分配的计算任务 C_i，则可将式(8)转换为式(9)，如下所示：

$$t_{2cu,i} = H(B_i) \tag{9}$$

传输时延与带宽的关系如图 3 可知。簇内节点所分配的带宽用传输时延近似为反比例函数，则式(9)可转换为式(10)，如下所示：

$$t_{2cu,i} = \frac{\alpha C_i}{B_i} \tag{10}$$

其中，α 为比例因子。

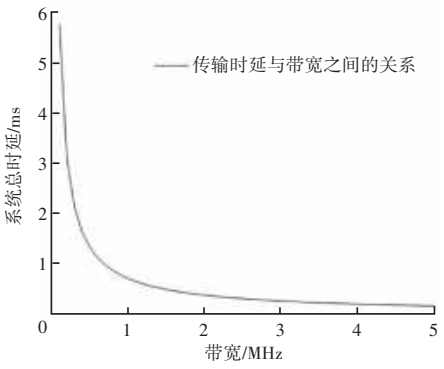

图 3　传输时延与带宽的关系

Fig. 3　Relationship between transmission delay and bandwidth

由 2.2.1 节分析方法可设 BTC 第三阶段中簇首到所有的簇内节点的传输时延相同，分析步骤可用如下公式来描述：

$$\frac{C_1}{B_1} = \frac{C_2}{B_2} = \cdots = \frac{C_i}{B_i} = \cdots = \frac{C_N}{B_N} \approx \beta \tag{g}$$

$$\Rightarrow C_i \approx B_i\beta \tag{h}$$

$$\Rightarrow C \approx B\beta \tag{i}$$

$$\Rightarrow \beta \approx \frac{C}{B} \tag{j}$$

$$\Rightarrow B_i \approx \frac{C_i}{\beta} = \frac{C_i B}{C} = \frac{f_i B}{F} \tag{k}$$

(11)

步骤(g)表示通过信道带宽分配，簇首到各簇内节点的传输时延相同，节约传输成本，其中 β 为传输时延因子；步骤(h)表示计算任务与信道带宽之间的近似关系；步骤(i)得到总带宽与总传输时延的近似关系；步骤(j)求得传输因子与总带宽和总传输时延的近似关系；因此节点信道带宽分配与可用计算力的关系如步骤(k)所示。

由上述分析可知，计算任务和信道带宽皆可通过比例分配算法进行分配，具体过程见算法 2。

算法 2　比例分配算法

输入　B, C, FF

输出　$C1, B1$

1. 由算法 1 得到聚簇详情，得到簇内节点的可用计算力 FF；

2. for $i = 1$ to N do

3. 计算簇内节点的总计算力 F；

4. end for

5. $C1 = \{C_1, C_2, \cdots, C_N\} \leftarrow$ 根据式(7)中的步骤(e)和(f)得到簇内节点的计算任务分配；

6. $B1 = \{B_1, B_2, \cdots, B_N\} \leftarrow$ 根据式(11)中的步骤(j)和(k)得到簇内节点的信道带宽分配

3 仿真结果及分析

本节详细分析了不同簇首选举算法下不同计算任务大小、小区内节点数和小区覆盖面积对系统总时延的影响。

为了评估本文算法的性能分别与文献[9]中的最小 ID 簇首选举算法(LID)、文献[16]中的自适应加权聚簇选举算法(A-WCA)和文献[19]中的等角度聚簇算法(EAC)进行对比。

3.1 仿真环境建立

本文所采用的仿真参数见表 1。

表 1 系统仿真参数设置

Table 1 System simulation parameter settings

仿真参数	变量	仿真数值
簇首发射功率	$P_{cu,i}$/ dbm	20
基站发射功率	$P_{BS,cu}$/ dbm	30
热噪声功率	n_0 /dbm	-144
路径损耗因子	l	4
本地计算能力	f_i /GHz	0.01~2.00
处理 1 bit 任务所需的 CPU 周期	ϕ	1 500
计算任务大小	C /bit	10^3~10^4
受损边缘服务器可提供的最大信道带宽总和	B /MHz	10
基站若覆盖范围	R /m	100~260
用户数	Num	100~300

3.2 仿真分析

LID 算法能最快聚簇,但该算法没有考虑整体簇的性能,因此聚簇结果并不稳定,簇中计算力无法得到保证;A-WCA 算法通过综合多个因素选举簇首,在一定程度上保证了簇的性能,但仅考虑簇首权重作为聚簇标准,使该算法最终花费的时延往往不是最小值;EAC 算法通过等角度聚簇,利用基站到簇首的通信半径,得到系统时延最低的最佳簇首位置,但无法保证簇首最大实现连接度,簇中计算力受影响。MCC 算法利用簇中计算力最大化选举簇首,保证了整个簇的系统计算性能,再利用比例分配算法进行资源分配,能最大程度地缩小系统时延。

图 4 表示为用户数为 150,小区覆盖范围为 150 m,计算任务与带宽比例分配下,不同算法下时延随计算任务的变化趋势。随着计算任务的增大,各种算法时延都有明显的增加。由于 LID 算法簇首选举随机,簇中计算合力不稳定,所以即使计算任务较小,时延也可能大于计算任务较大时花费的时延,该算法整体性能不稳定。A-WCA 算法和 EAC 算法通过不同考虑的角度选举簇首,2 个算法性能相近。相比其他几种算法,本文算法能够获得最低的时延。

图 5 表示为用户数为 150,小区覆盖范围为 150 m,计算任务与带宽平均分配下,不同算法下时延随计算任务的变化趋势。随着计算任务的增大,各种算法时延都有明显的增加。计算任务与带宽比例分配方案时延远远小于平均分配。在平均分配下,计算时延主要取决于簇内节点个数,个数越多每个节点分配到的计算任务越少,因此各算法性能均不稳定。LID 算法由于无法保证每次迭代所聚簇的计算力的稳定性,因此存在计算任务越小、计算时延却越大的情况。

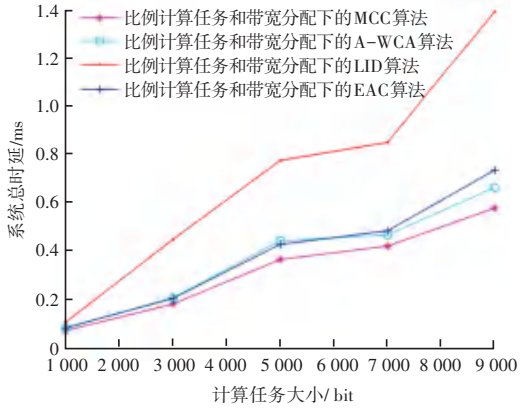

图 4 系统总时延与计算任务的关系

Fig. 4 Relationship between total system delay and computing tasks

图 6 表示为计算任务大小为 10^4 bit,小区覆盖范围为 150 m,计算任务与带宽比例分配下,不同算法下时延随小区覆盖用户数的变化趋势。随着用户数的增加,各簇首的邻居节点也增多,系统计算合力也变大,因此各种算法时延都有明显的下降。相比其他几种算法,本文算法能够获得最低的时延。

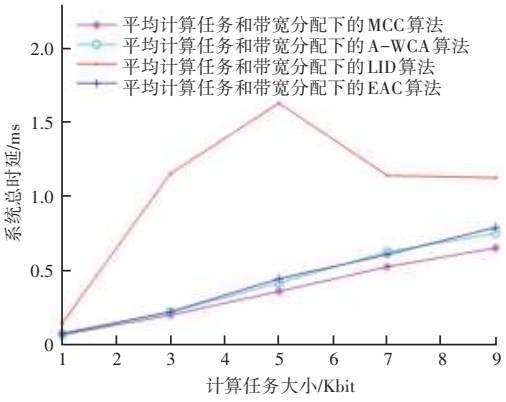

图 5　系统总时延与计算任务的关系

Fig. 5　Relationship between total system delay and computing tasks

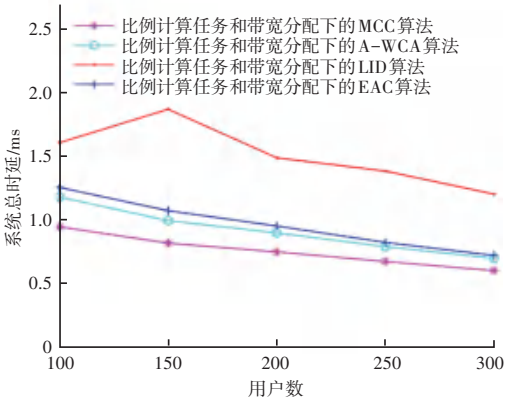

图 6　系统总时延与用户数的关系

Fig. 6　Relationship between the total system delay and the number of users

图 7 表示为用户数为 150，计算任务大小为 10^4 bit，计算任务与带宽比例分配下，不同算法下时延随小区覆盖范围的变化趋势。在用户数固定的情况下，小区覆盖范围越大，簇首的邻居节点变少，系统计算合力也变小，因此各种算法时延都有明显的上升，LID 聚簇的不稳定性，导致其变化趋势存在差异。相比其他几种算法，本文算法能够获得最低的时延。

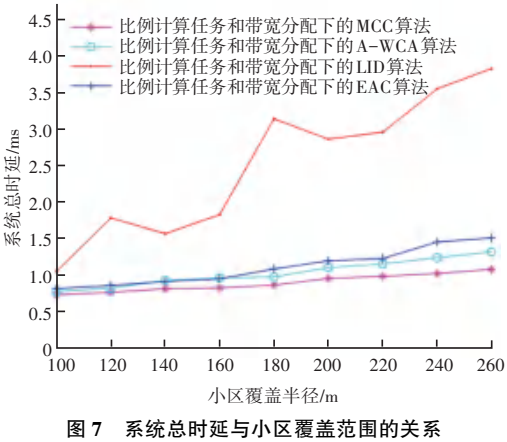

图 7　系统总时延与小区覆盖范围的关系

Fig. 7　Relationship between total system delay and cell coverage

4　结束语

本文提出一种基站端计算资源调度策略，其过程包括基于最大计算力的聚簇选举簇首算法、基于终端节点的可用计算力进行比例分配计算任务和带宽算法，能在基站无法完成计算任务时最有效地降低系统总时延，实现终端计算力共享。论文首先利用最大计算力选出簇首组建簇，确定簇内节点后，通过比例计算任务分配和带宽分配的方法，为簇内节点求得最佳的资源分配方案。由此，在 BTC 中采用有效资源调度策略，将计算任务分配至移动终端簇内部，利用终端节点本地计算能力，解决 EC 中的边缘服务器损坏后计算力不足的问题。

参考文献

[1] AGEEDZ S, ZEEBAREE S R M, SADEEQ M A M, et al. Comprehensive study of moving from grid and cloud computing through fog and edge computing towards dew computing[C]// 2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA). Piscataway, NJ:IEEE, 2021: 68-74.

[2] HUANG Zeyu, XIA Geming, WANG Zhaohang, et al. Survey on edge computing security[C]//2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE). Piscataway, NJ:IEEE, 2020: 96-105.

[3] SHIRAZI S N, GOUGLIDIS A, FARSHAD A, et al. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective[J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2586-2595.

[4] CHEN Nanxi, YANG Yang, LI Jin, et al. A fog-based service enablement architecture for cross-domain IoT applications[C]// 2017 IEEE Fog World Congress (FWC), Piscataway, NJ:IEEE, 2017: 1-6.

[5] CHEN Nanxi, CLARKE S, CHEN Shu. Fog-based service enablement architecture[C]// Fog and Fogonomics: Challenges and Practices of Fog Computing, Communication, Networking, Strategy, and Economics. New York: Wiley, 2020:151-177.

[6] LI Xiaomin, WAN Jialu, DAI Hongning, et al. A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing[J]. IEEE Transactions on Industrial Informatics, 2019, 15(7): 4225-4234.

[7] 赵兴兵，李波，杨志军，等. 边缘计算中的边缘服务器放置策略[J]. 计算机工程与设计，2022，43(11)：3008-3014.

[8] SALEEM U, LIU Yu, JANGSHER S, et al. Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing[J]. IEEE Transactions on Wireless Communications, 2021,20(1):360-374.

[9] HU Guangming, HUANG Zunguo, HU Huaping, et al. SLID: A secure lowest-ID clustering algorithm[J]. Wuhan University Journal of Natural Sciences, 2005(1):39-42.

[10] 胡光明，蒋杰，龚正虎. 移动自组网络分簇算法综述[J]. 计算机工程与科学，2006，27(1)：48-50，53.

[11]ZHANG Jian, WANG Bin, ZHANG Fei. A distributed approach of WCA in Ad-Hoc network[C]// 2006 International Conference on Wireless Communications, Networking and Mobile Computing. Piscataway, NJ:IEEE, 2006:1-5.

[12]LIN C R, GERLA M. A distributed architecture for multimedia in dynamic wireless networks[C]// Proceedings of GLOBECOM′95. Piscataway, NJ:IEEE, 1995, 2: 1468-1472.

[13]EPHREMIDES A, WIESELTHIER J E, Baker D J. A design concept for reliable mobile radio networks with frequency hoppingsignaling[J]. Proceedings of the IEEE, 1987, 75(1): 56-73.

[14]GERLA M, TZU-CHIEH TSAI J. Multicluster, mobile, multimedia radio network[J]. Wireless Networks, 1995, 1(3): 255-265.

[15]LUO Jun, HUBAUX J P. A survey of research in inter-vehicle communications[M]//LEMKE K, PAAR C, WOLF M. Embedded Security in Cars. Cham:Springer,2006: 111-122.

[16]CHATTERJEE M, DAS S K, TURGUT D. An on-demand weighted clusteringalgorithm(WCA) for Ad Hoc networks[C]// Globecom′00-IEEE. Global Telecommunications Conference. Conference Record (Cat. No. 00CH37137). Piscataway, NJ: IEEE, 2000, 3: 1697-1701.

[17]林超. Ad Hoc 网络聚簇节点组通信功能的设计与实现[D]. 天津:天津大学,2007.

[18]SHI Feng, SHI Yongge, LAI Lin. A clustering algorithm of Ad-Hoc network based on honeycomb division[C]//2011 IEEE International Conference on Granular Computing. Piscataway, NJ: IEEE, 2011: 863-866.

[19]ZHENG Sihai, ZHENG Changrui. A weight-based clustering routing algorithm for Ad Hoc networks[C]//2021 20th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES). Piscataway, NJ: IEEE, 2021: 123-126.

[20]李旭杰,刘春燕,孙颖. 面向蜂窝网络的 D2D 多播通信的聚簇和中继选择方法[J]. 电子与信息学报,2023,45(2):488-496.