一种基于改进遗传算法的雾计算任务调度策略

韩奎奎 谢在鹏 吕 鑫
(河海大学计算机与信息学院 南京 211100)

摘 要 任务的调度与分配一直以来都是云计算技术发展中的关键问题。然而,随着物联网连接设备的爆炸式增长,云计算已不能满足一些任务的调度需求,如健康检测、应急响应等都需要较低的延迟,雾计算应运而生。雾计算将云的服务扩展到网络边缘。雾计算架构下的任务调度与分配目前还是一个较新的研究热点。文中介绍了一种改进的遗传算法(IGA),该算法将适应度判断引入到亲代变异操作中,克服了基本遗传算法(SGA)在变异操作中的盲目性。在雾计算架构下采用该算法调度任务时考虑了服务等级目标(SLO)中响应时间的约束(FOG-SLO-IGA)。实验结果表明,FOG-SLO-IGA调度用户任务时在时延、SLO违规率以及服务提供商的花费上均低于云计算架构下采用IGA的调度(CLOUD-IGA);同时,在雾端调度任务时,IGA算法在执行速度上要快于传统SGA算法和轮询调度算法(RRSA)。

关键词 任务调度,云计算,雾计算,服务等级目标,遗传算法

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.022

Fog Computing Task Scheduling Strategy Based on Improved Genetic Algorithm

HAN Kui-kui XIE Zai-peng LV Xin
(College of Computer and Information, Hohai University, Nanjing 211100, China)

Abstract Task scheduling and assignment has always been a key issue in the development of cloud computing. However, with the explosive growth of internet connection devices, cloud computing has been unable to meet some requirements such as health monitoring, emergency response and so on, which all require low latency. Thus fog computing appears. Fog computing extends the cloud services to the edge of network. Under the fog computing architecture, task scheduling and assignment is still a relatively new research hotspot. This paper introduced an improved genetic algorithm(IGA). The algorithm introduces the fitness judgment into the parental mutation operation which overcomes the blindness of simple genetic algorithm(SGA) in mutation operation. The response time restriction in the service level objective(SLO) is considered when the IGA is used to schedule tasks(FOG-SLO-IGA). The experimental results show that when scheduling user tasks are under the fog computing architecture, FOG-SLO-IGA is superior to the scheduling which uses IGA under cloud computing architecture in latency, SLO violation rate and service provider's cost. Futhermore, IGA algorithm is superior to the traditional SGA algorithm and the round-robin scheduling algorithm(RRSA) in the execution of the tasks under the fog computing architecture.

Keywords Task scheduling, Cloud computing, Fog computing, Service level objective, Genetic algorithm

1 引言

随着互联网时代信息与数据的快速增长,云计算作为一种新型的计算模式受到越来越多的关注。近年来,云计算为传统的计算、数据存储以及服务形式指明了新方向[1-3],然而连接到互联网的移动和传感设备的迅速增加给云计算框架的传统网络架构带来了挑战。据思科估计,目前全球有250亿个物联网连接设备,到2020年这个数字可能会达到500亿[4]。大量的位于网络边缘的终端设备需要低延迟、位置感知、可移动性的服务[5-7]。为此,思科提出了一种新的服务计算模式——雾计算。雾计算是一个高度虚拟化的平台,在物联网终端节点和传统云之间提供计算、存储以及网络服务[8]。

近年来,国内外许多学者对雾计算进行了初步研究,主要研究方向有任务调度与分配、雾计算参考架构、功耗时延优化等。基于终端用户对时间的要求,雾计算架构下的任务调度还是一个较新的研究热点。吴翠云等人[9]提出了一种在"云+雾"架构下的任务请求分配模式,即通过雾端和云端相互协作来处理终端用户的请求数据,之后根据处理结果调整被控终端。Gupta等人[10]介绍了两个应用在雾计算架构下的案例模型,指出按照要求将应用模型的不同模块调度到云端和

到稿日期:2017-05-23 返修日期:2017-06-22 本文受国家自然科学基金面上项目(61272543),NSFC-广东联合基金重点项目(U1301252)资助。
韩奎奎(1990-),女,硕士生,主要研究方向为云雾计算,E-mail:15850670386@163.com;**谢在鹏**(1982-),男,博士,讲师,CCF会员,主要研究方向为云计算与大数据平台、嵌入式系统架构,E-mail:zxiehhu@163.com(通信作者);**吕 鑫**(1983-),男,博士,讲师,主要研究方向为密码学、网络信息安全。

雾端资源进行处理，节省了时延，降低了网络使用量。在雾计算架构下，设计一个高效的任务调度和资源管理策略是非常重要的。Zeng 等人[11]针对雾计算架构中的任务调度和资源管理进行了两方面的研究：1）如何平衡客户端设备和计算服务器之间的工作负载，即任务调度；2）如何将任务放置到服务器上，即资源管理。在雾计算参考架构方面，文献[12]提出了一种基于雾计算的新型“云＋雾”混合网络参考架构，并基于该架构进一步提出了一种多设备分布式计算方法，利用带约束的粒子群优化负载均衡算法达到最小化任务处理时延的目标。文献[13]提出了一种基于“智能前端化”思想的雾计算参考架构，该架构是在云服务和移动终端之间再扩展出的一层雾计算层；同时，针对医院就医场景研发了能提供就医信息查询及多媒体资源点播的服务系统。在功耗时延方面，文献[14]为雾计算这种新型的计算模式提出了一个数学模型，并针对雾计算和云计算在时延、能耗方面的性能进行了比较，结果表明在 25％的物联网应用程序需要实时、低延迟服务的情况下，雾计算的平均能耗比常规云计算模型的能耗低 40.48％。

SLA 是服务提供商和用户经过协商而确定的关于服务质量的等级协议，主要涵盖服务性能、服务时间、服务计费等可计量的服务质量特性[15]。SLA 是整个协议，指定将提供哪些服务以及怎样支持时间、位置、成本、绩效和责任等；SLO 是 SLA 的具体可测量特征，如可用性、吞吐量、频率、响应时间、质量等。SLO 提供了一种量化手段来定义客户可以从提供商处获得的服务水平。

遗传算法是一种全局随机优化算法，具有并行性和全局解空间搜索两个显著的特点[16]；但是相关应用表明，传统遗传算法也存在一些缺陷，如遗传操作中存在局限性。本文基于传统遗传算法在变异操作中的盲目性引入了亲代适应度判断，并基于 SLO 中的响应时间特征提出基于 SLO 的花销方程。改进的遗传算法在雾计算架构下根据任务请求对时间的约束分类调度用户任务，弥补了传统任务完全在云上处理的不足，降低了时延以及 SLO 违规率，提升了终端用户的满意度。

2 问题描述

通用的雾计算架构是一个三层网络结构[14]，如图 1 所示。底层是终端用户的物联网设备层，主要由智能手机、PC 机、智能手表等组成；中间层是雾计算层，主要由具有一定计算能力的雾设备（如路由器、网关、小型服务器等）组成；上层是云计算中心层，主要由能够处理和存储大量数据的云服务器组成。在该架构下，终端用户的任务请求类型有时间紧迫型和非时间紧迫型[17]。对于时间紧迫型请求，按照调度策略将其调度到离用户近的雾端进行处理，以减少时延，提升用户满意度；而对于其他非时间紧迫型的请求，根据调度策略将其调度到云端进行处理。调度的步骤如下：

1）用户提交任务请求，即集合 $T=\{T_1,T_2,T_3,\cdots,T_k\}$，其中 k 是正整数，表示请求的序号。

2）判断任务请求类型是时间紧迫型还是非时间紧迫型。

3）根据调度策略将相应的任务请求调度到雾计算资源节点和云计算资源节点，即集合：

$VM^C=\{VM_1^C,VM_2^C,VM_3^C,\cdots,VM_m^C\}$

$VM^F=\{VM_1^F,VM_2^F,VM_3^F,\cdots,VM_n^F\}$

其中，m 和 n 为正整数，分别表示云计算资源节点和雾计算资源节点的序号。

4）处理任务请求。

5）释放资源。

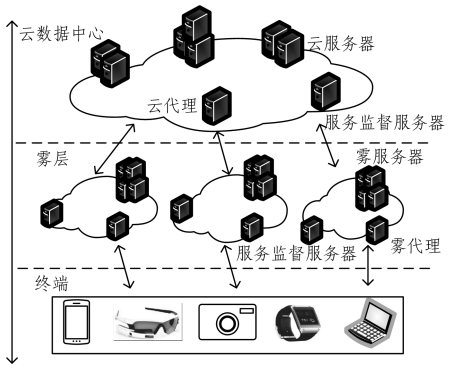

图 1 雾计算架构

Fig. 1 Architecture of fog computing

3 雾计算任务调度的目标函数

雾计算架构下任务的调度与分配是指在一个特定的雾计算环境中，根据一定的调度策略，尽可能满足终端用户的请求，同时最小化服务提供商的花费。文中的目标函数主要考虑服务提供商的花费，主要包括 SLO 违规时的惩罚和云雾计算资源节点处理终端请求的花费两个方面。

1）SLO 违规惩罚函数。设 $Penalty$ 表示发生 SLO 违规的总惩罚花费，则违规惩罚函数的定义为：

$$Penalty=\sum_{k=1}^{q}[\gamma_k+\eta\times(t_k^{\text{stop}}-t_k^{\text{deadline}})] \tag{1}$$

其中，k 表示因为服务超时而发生 SLO 违规的请求的序号；q 表示发生 SLO 违规的请求的总个数；γ_k 表示第 k 个发生 SLO 违规的请求的基本惩罚花销；η 表示因发生超时而导致的单位时间的惩罚花销；t_k^{stop} 表示第 k 个请求的实际响应时间；t_k^{deadline} 表示第 k 个请求在 SLO 规定中的请求截止响应时间。

2）云雾计算资源节点的花费函数。设 vm 表示云雾计算资源节点处理请求的总花费，则计算资源花费函数的定义为：

$$vm=\sum_{i=1}^{n}vmc_i \tag{2}$$

$$vmc_i=\sum_{j=1}^{M_i}(P_{i,j}^{vm}\times t_{i,j}) \tag{3}$$

其中，n 表示云端和雾端虚拟资源的类型总数；vmc_i 表示第 i 个类型的计算资源处理请求的总花费；M_i 表示类型为 i 的虚拟资源节点的个数；$P_{i,j}^{vm}$ 表示类型为 i 的虚拟资源节点中序号为 j 的单个虚拟资源节点运行的单价成本；$t_{i,j}$ 表示类型为 i 的虚拟资源节点中序号为 j 的虚拟资源节点处理请求所花费的时间。

综上可得服务提供商的总花费函数。设 $cost$ 表示服务提供商的总花费，则其定义为：

$$cost=vm+Penalty \tag{4}$$

4 相关调度算法

4.1 基本遗传算法

美国 Michigan 大学的 J. H. Holland 提出的遗传算法被

称为基本遗传算法(SGA)。该算法是其他一些遗传算法的雏形和基础，不仅给各种遗传算法提供了一个基本框架，同时也具有一定的应用价值。SGA 的基本思想是模拟自然界中的生物遗传和进化机理，即通过模仿生物遗传和进化过程中的选择、交叉、变异机理来完成对问题最优解的自适应搜索过程[18]。SGA 的实现过程实质上是对个体进行重组，其流程如图 2 所示。

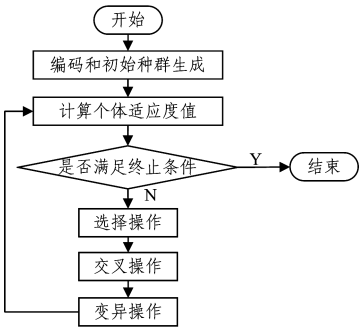

图 2　基本遗传算法实现流程

Fig. 2　Flow chart of simple genetic algorithm

4.2　改进的遗传算法设计

1)对个体进行编码

在采用 IGA 求解雾计算终端用户的任务调度问题的过程中，必须建立任务与资源之间的映射关系，即对染色体进行编码。文中对染色体采用混合编码的方式[19]，染色体的长度等于请求数量的两倍。假设请求数量为 L，则染色体的长度为 $2*L$，前 L 个基因表示请求分配的资源情况，后 L 个基因表示请求的调度顺序。假设一条染色体为 5 1 2 3 4 5 2 1 2 3 4 5 6 7，则请求和资源的对应关系如表 1 所列。

表 1　个体的编码方式

Table 1　Individual coding

请求编号	1	2	3	4	5	6	7
资源编号	5	1	2	3	4	5	2

2)生成初始种群

初始化种群对于雾计算架构下的终端用户任务调度问题相当重要。随机生成初始种群，并根据约束条件排除无效方案。设初始种群的大小为 M，迭代次数 $i=0$。采用的约束条件主要是终端用户请求中的时间紧迫型请求必须调度到距离用户近的雾端进行处理，非时间紧迫型的请求则被调度到云端进行处理，且用户的所有请求不能同时放到同一个资源上，以免造成严重的负载不均。

3)计算适应度函数

适应度函数是雾计算架构下终端用户任务调度中评估群体进化方向的关键。为了使适应度函数的定义方式在 $cost$ 值为 0 时依然有意义，将其定义为：

$$f(cost)=e^{-cost} \tag{5}$$

4)选择操作

采用轮盘赌的方式进行选择。假设种群的规模为 M，个体 i 的适应度值为 f_i，p_i 为个体 i 被选择的概率，则概率计算式为：

$$p_i=f_i/\sum_{j=1}^{M}f_j \tag{6}$$

5)交叉操作

实验中针对资源队列采用单点交叉，随机产生交叉点位置，从而产生资源队列的新个体。针对任务请求队列采用部分匹配交叉，从而产生请求队列的新个体。将资源队列和请求队列合并，从而产生新的染色体，检查产生的新个体是否满足约束条件，若不满足则将其舍弃。对亲代个体重新进行交叉，直至交叉产生满足约束条件的新个体。

例如，对于亲代个体的染色体

M：5 1 2 3 4 5 1 2 3 4 5 6

F：5 2 1 3 5 4 2 4 3 1 5 6

其中，前面的 6 位数中的 1，2，3，4，5 表示 5 个计算资源；后面的 6 位数 1，2，3，4，5，6 表示 6 个设备请求。

资源队列采用单点交叉方式，设随机产生交叉点的位置为 2，可得交叉后子代 D 和 S 的资源队列如下：

M：5 1|2 3 4 5 $\rightarrow$ D：5 1 1 3 5 4

F：5 2|1 3 5 4 $\rightarrow$ S：5 2 2 3 4 5

请求队列采用部分匹配交叉，设随机产生两个交叉点的位置为 2 和 4，可得交叉后子代 D 和 S 的资源队列如下：

M：1 2|3 4|5 6 $\rightarrow$ $TEMPD$：1 2 3 1 5 6

F：2 4|3 1| 5 6 $\rightarrow$ $TEMPS$：2 4 3 4 5 6

$TEMPD$：1 2 3 1 5 6 $\rightarrow$ D：4 2 3 1 5 6

$TEMPS$：2 4 3 4 5 6 $\rightarrow$ S：2 1 3 4 5 6

其中，对于产生的交叉子代 $TEMPD$ 和 $TEMPS$ 中在匹配区域以外出现的重复数字，依据匹配区域内的数字逐一进行替换，匹配关系有：4↔1。

资源队列和请求队列合并后产生的新染色体如下：

D：5 1 1 3 5 4 4 2 3 1 5 6

S：5 2 2 3 4 5 2 1 3 4 5 6

6)变异操作

实验中针对资源队列采用随机操作的方式，即随机选取变异的基因位置并用其等位基因来替换。针对任务请求队列，采用的变异方法为：设定整数 D 为发生变异的基因位数，随机生成变异点，对以变异点为起点的随后 D 个基因位采用完全排列组合的方式产生出新个体。将变异后的资源队列和请求队列合并，从而产生新的染色体，检查子代是否满足约束条件，若不满足则将其舍弃。对亲代个体重新进行变异，直至变异产生满足约束条件的新个体。

例如，对于亲代个体的染色体

X：5 1 2 3 5 4 4 2 3 1 5 6

资源队列采用随机的操作方式，假设随机产生变异点的位置为 3，则变异后子代的资源队列如下：

X：5 1 $\underline{\underline{2}}$ 3 5 4 $\rightarrow X^{new}$：5 1 1 3 5 4 或 5 1 5 3 5 4 或 5 1 3 3 5 4 或 5 1 4 3 5 4

请求队列随机生成的变异点位置为 2，对其后的 2 个基因位采用排列组合的方式，则变异后子代的请求队列如下：

X：4 $\underline{\underline{2}}$ 3 1 5 6 $\rightarrow X^{new}$：4 3 2 1 5 6

资源队列和请求队列合并后产生的新染色体如下：

X'：5 1 1 3 5 4 4 3 2 1 5 6 或 5 1 5 3 5 4 4 3 2 1 5 6 或 5 1 3 3 5 4 4 3 2 1 5 6 或 5 1 4 3 5 4 4 3 2 1 5 6

7)判断适应度值

对于变异后产生的新个体，若其适应度值小于亲代个体，

则用亲代个体替换；若大于或等于亲代，则替换亲代个体。

4.3 算法复杂性的分析

根据 4.2 节对改进遗传算法的设计，分析 IGA 算法的时间复杂度。设 M 为初始种群的大小，L 为终端请求的数量，I 为迭代次数，$2*L$ 为一条染色体的长度。

整个 IGA 算法各部分的时间复杂度如下：

初始化种群的时间复杂度为 $O(M*2*L)\approx O(M*L)$，初始化操作只计算一次，对时间影响较小，可忽略；

适应度值计算的时间复杂度为 $O(M*L)$；

选择操作的时间复杂度为 $O(3*M+M*M)\approx O(M*M)$；

交叉操作的时间复杂度为 $O(M/2)\approx O(M)$；

变异操作的时间复杂度为 $O(M)$。

综合以上各部分的复杂度，可得整个算法的时间复杂度为 $O(I*(M*M+2*M+M*L))$。

4.4 雾计算任务调度的步骤

改进遗传算法的操作步骤如下：

1)初始种群。用户提交终端请求，并根据请求随机设置初始种群 $g(0)$，设定迭代计数器 $i=0$，$g(i)$ 表示第 i 代种群。

2)计算适应度值。根据 4.2 节设计的改进遗传算法中的适应度值公式来计算个体的适应度值。

3)判断终止条件。当进化代数达到规定迭代次数 I 时，输出结果，并找到较优的雾计算任务调度方案；否则转到步骤 4)。

4)选择操作。采用轮盘赌的方式对种群进行选择操作。

5)交叉操作。根据 4.2 节设计的改进遗传算法中的交叉方法，在选择操作产生的个体中随机选择两个个体，并根据交叉概率 p_c 对其进行操作，从而产生新个体。

6)变异操作。根据 4.2 节设计的改进遗传算法中的变异方法，通过交叉操作产生新个体，并根据变异率 p_m 对新个体中被选中的个体进行操作，从而产生新个体。

7)判断适应度值。对变异后的个体进行适应度值判断，若其适应度值小于亲代个体则用亲代替换；若大于或等于亲代则替换亲代。新一代个体产生 $g(i+1)$。

8)更新迭代计数器。迭代计数器 $i=i+1$，转到步骤 2)。

IGA 的具体流程如图 3 所示。

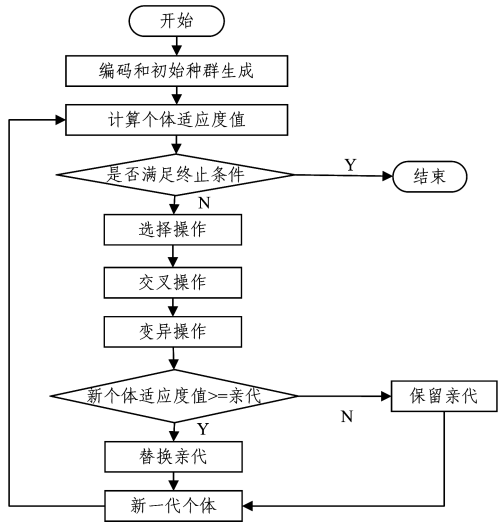

图 3 改进遗传算法的流程

Fig. 3 Flowchart of improved genetic algorithm

5 仿真实验

5.1 数据及参数设置

实验仿真平台采用开源 CloudSim 3.0[20]，计算机的 CPU 为 Intel i5-4210 M，内存为 12 GB，操作系统为 Windows10 旗舰版。

为了验证本文提出的 IGA 在云计算架构和雾计算架构任务调度中的性能，本节对云计算架构下的 CLOUD-RRSA，CLOUD-IGA 以及雾计算架构下的 FOG-SLO-IGA 进行仿真并对其结果进行分析；评价 FOG-SLO-IGA 调度策略，对 IGA 算法与 SGA 算法、CloudSim 自带的 RRSA 等经典的任务调度算法在雾端调度任务时的性能进行比较。实验中的终端用户请求数量分别设置为 20，40，60，80，100，200，300，400，500，600；请求大小为 500～15000MI；云雾计算的虚拟资源节点个数为 6，其中 4 个为云计算的虚拟资源节点，2 个为雾计算的虚拟资源节点。云计算资源的参数如表 2 所列，雾计算资源的参数如表 3 所列，IGA 的参数如表 4 所列。

表 2 云计算资源的参数列表

Table 2 Parameter list of cloud computing resource

资源编号	处理能力/MIPS	单价/(CENT/S)	带宽/MB
0	5000	6	1000
1	2500	5	1000
2	2500	5	1000
3	1500	4	1000

表 3 雾计算资源的参数列表

Table 3 Parameter list of fog computing resource

资源编号	处理能力/MIPS	单价/(CENT/S)	带宽/MB
4	1000	3	2000
5	1000	3	2000

表 4 IGA 算法的参数列表

Table 4 Parameter list of IGA

参数名称	参数值
基础惩罚 γ_k	2
单位时间惩罚 η	5
迭代次数 I	50
交叉概率参数 p_c	0.8
变异概率参数 p_m	0.01
初始种群大小 m	20

5.2 结果与分析

随着请求数量的变化，FOG-SLO-IGA 和 CLOUD-IGA 和 CLOUD-RRSA 的时延对比关系如图 4 和图 5 所示。

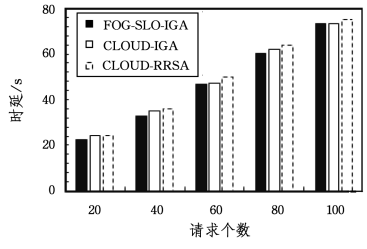

图 4 不同请求数下的时延对比

Fig. 4 Comparison of delay under different requests

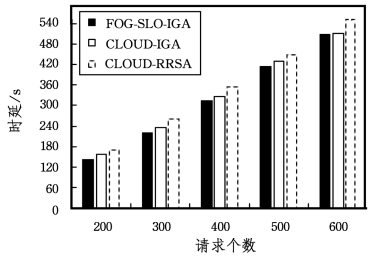

图 5　不同请求数下的时延对比(请求数较多)

Fig. 5　Comparison of delay under different requests when the unmber of requests is large

从图 4 和图 5 可以看出，相对于 CLOUD-IGA 和 CLOUD-RRSA 调度，FOG-SLO-IGA 调度在时延上更具优势，但随着请求数量的逐渐增多，该优势逐渐减弱，主要是因为雾计算资源节点的处理能力相对较弱，但更接近终端用户的特点使得其对时间要求较高的请求仍然具有优势。FOG-SLO-IGA 将对时间要求较高的请求调度到雾计算资源节点上，相比于任务完全在云上处理的传统算法，可以更有效地降低时延。

采用 SLO 违规率、服务提供商的花费作为评价指标，对 FOG-SLO-IGA 和 CLOUD-IGA 的调度性能进行全面分析。两种算法的各项评价指标的具体对比关系如图 6 和图 7 所示。

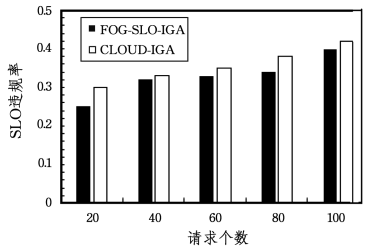

图 6　不同请求数下的违规率对比

Fig. 6　Comparison of the rate of violation under different requests

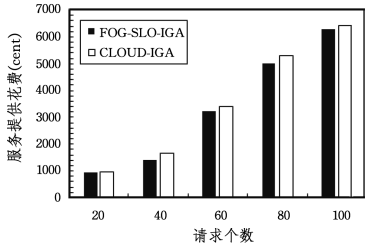

图 7　不同请求数下服务提供商的花费对比

Fig. 7　Comparison of the service provider's cost under different requests

从图 6 可以看出，FOG-SLO-IGA 的违规率比 CLOUD-IGA 平均约低 3%。这主要是因为 FOG-SLO-IGA 在调度请求时考虑了请求对时间的约束，分类在云雾计算资源节点上调度；而 CLOUD-IGA 没有考虑任务对时间的约束，完全在距离用户较远的云端调度用户请求，增加了响应时间。因此，FOG-SLO-IGA 减少了 SLO 违规。从图 7 可以看出，与 CLOUD-IGA 相比，FOG-SLO-IGA 在服务提供商的花费方面也较少，平均减少约 131cent。这主要是因为 FOG-SLO-IGA 降低了 SLO 违规，且雾计算资源节点在单位时间内的费用较低，因此其在服务提供商的花费上有所降低。

为了进一步验证本文提出的 IGA 算法的性能，将其与 SGA 算法、RRSA 算法等经典的任务调度算法在雾端调度时间紧迫型任务时进行了请求执行时间的对比实验，对比结果如图 8 所示。

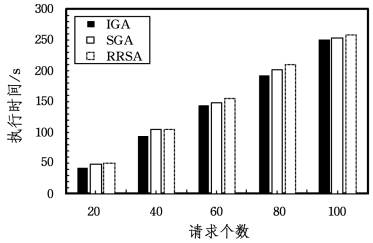

图 8　不同请求数下执行时间的对比

Fig. 8　Comparison of execution time under different requests

从图 8 可以看出，在雾端调度时间紧迫型任务时，传统的 SGA 算法在总执行时间上要优于 RRSA 算法，其执行时间平均约减少 4s。同理可以看出，本文提出的 IGA 算法通过引入变异操作适应度判断后，总执行时间也短于传统 SGA 的执行时间，平均约减少7s。因此，优化后的算法更适应雾端的任务调度。

结束语　随着信息技术的进一步发展，雾计算正逐渐成为处理数据问题的较有吸引力的解决方案，其具有比终端设备更强的处理能力，并且比强大的云计算资源更接近这些终端设备，减少了应用的延迟。基于雾计算的这种特点，本文提出一种改进的遗传算法(IGA)，在雾计算架构下采用该算法调度任务时考虑了服务等级目标(SLO)中响应时间的约束，因此在时延、SLO 违规以及服务提供商的花费上均优于云计算架构下采用 IGA 的调度；同时，该算法在雾端调度时间紧迫型任务时也要优于传统的 SGA 算法和 RRSA 算法。

参 考 文 献

[1] LI Q, ZHENG X. A Review of the Present Situation about Cloud Computing[J]. Computer Science, 2011, 38(4): 32-37. (in Chinese)
李乔，郑啸. 云计算研究现状综述[J]. 计算机科学，2011，38(4)：32-37.

[2] LAI C F, WANG H, CHAO H C, et al. A Network and Device Aware QoS Approach for Cloud-Based Mobile Streaming[J]. IEEE Transactions on Multimedia, 2013, 15(4): 747-757.

[3] RIMAL B P, CHOI E, LUMB I. A Taxonomy and Survey of Cloud Computing Systems[C]// International Joint Conference on Inc, Ims and IDC. IEEE, 2009: 44-51.

[4] MarketWatch: 'Cisco delivers vision of fog computing to accelerate value from billions of connected devices'[OL]. http://www.the-iet.org/resources/journals/research/index.cfm.

[5] BONOMI F, MILITO R, ZHU J, et al. Fog computing and its role in the internet of things[C]// Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. ACM, 2012: 13-16.

[6] LUAN T H, GAO L, LI Z, et al. Fog computing: Focusing on mobile users at the edge[J]. arXiv preprint arXiv: 1502.01815, 2015.

[7] AAZAM M,HUH E N. Fog computing and smart gateway based communication for cloud of things[C]// 2014 International Conference on Future Internet of Things and Cloud(FiCloud). IEEE,2014:464-470.

[8] AAZAM M, HUH E N. Fog Computing: The Cloud-IoT/IoE Middleware Paradigm[J]. IEEE Potentials,2016,35(3):40-44.

[9] 北京物联远信息技术有限公司. 一种面向物联网的雾计算架构:中国,201511019375. 3[P/OL]. (2016-05-25). http://www.soopat.com/patent/201511019375.

[10] GUPTA H,DASTJERDI A V,GHOSH S K,et al. i-FogSim:A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things,Edge and Fog Computing Environments[J]. arXiv preprint arXiv:1606.02007,2016.

[11] ZENG D,GU L,GUO S,et al. Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System[J]. IEEE Transactions on Computers,2016,65(12):3702-3712.

[12] HE X L,REN Z Y,SHI C H,et al. Cloud and Fog network for Medical Big Data and its Distributed Computing Scheme[J]. Journal of Xi'an Jiaotong University,2016,50(10):71-77. (in Chinese)
何秀丽,任智源,史晨华,等. 面向医疗大数据的云雾网络及其分布式计算方案[J]. 西安交通大学学报,2016,50(10):71-77.

[13] CHENG D M,LI Z. Hospital Information Service System Based on Fog Computing[J]. Computer Science,2015,42(7):170-190. (in Chinese)
程冬梅,李志. 基于雾计算的医院信息服务系统[J]. 计算机科学,2015,42(7):170-190.

[14] SARKAR S,MISRA S. Theoretical modelling of fogcomputing: a green computing paradigm to support IoT applications[J]. IET Journals,2016,5(2):23-29.

[15] WU L. SLA-based resource provisioning for management of Cloud-based Software-as-a-Service applications[D]. The University of Melbourne,2014.

[16] LI J F,PENG J. Task Scheduling Algorithm Based on Improved Genetic Algorithm in Cloud Computing Environment[J]. Journal of Computer Application,2011,31(1):184-186. (in Chinese)
李剑锋,彭舰. 云环境下基于改进遗传算法的任务调度算法[J]. 计算机应用,2011,31(1):184 -186.

[17] CHIANG M, ZHANG T. Fog and IoT: An Overvi-ew of Research Opportunities[J]. IEEE Internet of Things Journal, 2016,3(6):854-864.

[18] 周明,孙树栋. 遗传算法原理及应用[M]. 北京:国防工业出版社,1996:18-32.

[19] HE X L. A Generalized Genetic Algorithm for Task Scheduling [J]. Computer Engineering,2010,36(17):184-186. (in Chinese)
贺晓丽. 一种用于任务调度的广义遗传算法[J]. 计算机工程,2010,36(17):184-186.

[20] CALHEIROS R N,RANJAN R,BELOGLAZOV A,et al. CloudSim:a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software Practice & Experience,2011,41(1):23-50.

(上接第 111 页)

[10] ZHAO Y C,HU Z H,BAI Y,et al. An accurate segmentation approach for disease and pest based on DRLSE guided by texture difference[J]. Transactions of the Chinese Society for Agriculture Machinery,2015,46(2):14-19. (in Chinese)
赵瑶池,胡祝华,白勇,等. 基于纹理差异度引导的 DRLSE 病虫害图像精准分割方法[J]. 农业机械学报,2015,46(2):14-19.

[11] ZHAO Y C,HU Z H. Segmentation of fruit with diseases in natural scenes based on logarithmic similarity constraint Otsu [J]. Transactions of the Chinese Society for Agriculture Machinery,2015,46(11):9-15. (in Chinese)
赵瑶池,胡祝华. 基于对数相似度约束 Otsu 的自然场景病害果实图像分割[J]. 农业机械学报,2015,46(11):9-15.

[12] LI Z,HONG T S,ZENG X Y,et al. Citrus red mite image target identification based on K-means clustering[J]. Transactions of the Chinese Society of Agricultural Engineering,2013,28(23):147-153. (in Chinese)
李震,洪添胜,曾祥业,等. 基于 K-means 聚类的柑橘红蜘蛛图像目标识别[J]. 农业工程学报,2013,28(23):147-153.

[13] PANG X M,MIN Z J,KAN J M. Color image segmentation based on HSI and LAB color space [J]. Journal of Guangxi University(Natural Science Edition),2011,36(6):976-980. (in Chinese)
庞晓敏,闵子建,阚江明. 基于 HSI 和 LAB 颜色空间的彩色图像分割[J]. 广西大学学报(自然科学版),2011,36(6):976-980.

[14] HAN Z Z,ZHAO Y G,YANG J Z. Detection of embryo based on independent components for kernel RGB images in maize[J]. Transactions of the CSAE,2010,26(3):222-226. (in Chinese)
韩仲志,赵友刚,杨锦忠. 基于籽粒 RGB 图像独立分量的玉米胚部特征检测[J]. 农业工程学报,2010,26(3):222-226.

[15] SWAIN M J,BALLARD D H. Color indexing [J]. International Journal of Computer Vision,1991,7(1):11-32.

[16] OTSU N. A threshold selection method from gray-level histogram [J]. IEEE Transactions on Systems, Man and Cybernetics,1979,9(1):62-66.

[17] XU X Y,SONG E M,JIN L H. Characteristic analysis of threshold based on Otsu criterion [J]. Acta Electronica Sinica, 2009,37(12):2716-2719. (in Chinese)
许向阳,宋恩民,金良海. Otsu 准则的阈值性质分析[J]. 电子学报,2009,37(12):2716-2719.

[18] HU M K. Visual pattern recognition by moment invariants[J]. Ire Transaction on Information Theory,1962,8(2):179-187.

[19] LIU J,GENG G H,REN Z B. Plant pest recognition system based on multi-structure element morphology[J]. Computer Engineering and Design,2009,30(6):1488-1490. (in Chinese)
刘军,耿国华,任治斌. 基于多结构元素的农作物病虫识别系统[J]. 计算机工程与设计,2009,30(6):1488-1490.