软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(11):3306−3325 [doi: 10.13328/j.cnki.jos.005479]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

移动云计算中基于延时传输的多目标工作流调度*

周业茂[1], 李忠金[1,2], 葛季栋[1], 李传艺[1], 周筱羽[1], 骆 斌[1]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210046)

[2](杭州电子科技大学 计算机学院,浙江 杭州 310018)

通讯作者: 葛季栋, E-mail: gjd@nju.edu.cn

摘 要: 云计算和移动互联网的不断融合,促进了移动云计算的产生与发展.在移动云计算环境下,用户可将工作流的任务迁移到云端执行,这样不但能够提升移动设备的计算能力,而且可以减少电池能源消耗.但是不合理的任务迁移会引起大量的数据传输,这不仅损害工作流的服务质量,而且会增加移动设备的能耗.基于此,提出了基于延时传输机制的多目标工作流调度算法 MOWS-DTM.该算法基于遗传算法,结合工作流的调度过程,在编码策略中考虑了工作流任务的调度位置和执行排序.由于在用户不断移动的过程中,移动设备的无线网络信号也在不断变化,当传输一定大小的数据时,网络信号越强则需要的时间越少,从而移动设备的能耗也越少.而且工作流结构中存在许多非关键任务,延长非关键任务的执行时间并不会对工作流的完工时间造成影响.因此,在工作流调度过程中融入了延时传输机制 DTM,该机制能够同时有效地优化移动设备的能耗和工作流的完工时间.仿真结果表明,相对于 MOHEFT 算法和 RANDOM 算法,MOWS-DTM 算法在多目标性能上更优.

关键词: 移动云计算;工作流调度;多目标优化;遗传算法;延时传输

中图法分类号: TP18

中文引用格式: 周业茂,李忠金,葛季栋,李传艺,周筱羽,骆斌.移动云计算中基于延时传输的多目标工作流调度.软件学报, 2018,29(11):3306−3325. http://www.jos.org.cn/1000-9825/5479.htm

英文引用格式: Zhou YM, Li ZJ, Ge JD, Li CY, Zhou XY, Luo B. Multi-Objective workflow scheduling based on delay transmission in mobile cloud computing. Ruan Jian Xue Bao/Journal of Software, 2018,29(11):3306−3325 (in Chinese). http://www.jos.org.cn/1000-9825/5479.htm

Multi-Objective Workflow Scheduling Based on Delay Transmission in Mobile Cloud Computing

ZHOU Ye-Mao[1], LI Zhong-Jin[1,2], GE Ji-Dong[1], LI Chuan-Yi[1], ZHOU Xiao-Yu[1], LUO Bin[1]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210046, China)

[2](School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: The integration between cloud computing and mobile Internet promotes the development of mobile cloud computing. The tasks of workflow can be migrated to cloud that can not only improve the computing capacity of mobile device, but also reduce the energy consumption of battery. However, a great amount of data transmission introduced by using unreasonable tasks scheduling strategies can

* 基金项目: 国家重点研发计划(2016YFC0800803); 国家自然科学基金(61802095, 61802167, 61572162, 61572251, 61702144); 浙江省自然科学基金(LQ17F020003); 浙江省科技厅重点研发项目(2018C01012); 中央高校基本科研业务费专项资金

Foundation item: National Key Research and Development Program of China (2016YFC0800803); National Natural Science Foundation of China (61802095, 61802167, 61572162, 61572251, 61702144); Zhejiang Provincial National Science Foundation (LQ17F020003); Zhejiang Provincial Key Science and Technology Project Foundation (2018C01012); Fundamental Research Funds for the Central Universities

本文由面向智能制造的业务过程管理与服务技术专题特约编辑王建民教授、刘建勋教授推荐.

收稿时间: 2017-07-20; 修改时间: 2017-09-16; 采用时间: 2017-11-24; jos 在线出版时间: 2017-12-06

CNKI 网络优先出版: 2017-12-06 15:37:31, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1537.026.html

damage the QoS (quality of service) of workflow and increase the energy consumption of mobile device. In this paper, a multi-objective workflow scheduling is proposed based on delay transmission mechanism (MOWS-DTM) to optimize execution time of workflow and energy consumption of mobile device in mobile cloud computing environment. MOWS-DTM, derived from genetic algorithm, is combined with the process of workflow scheduling and takes both task scheduling location and execution sequence into consideration in coding strategy. When mobile user is moving, wireless network signal of mobile device is changing with the pace of different location. The stronger the network signal, the less time it takes to transmit data with fixed size, and the less energy the mobile device will consume. Moreover, there are many non-critical tasks reside in workflow, and increasing their execution time will not affect the makespan of workflow. Therefore, the delay transmission mechanism (DTM), incorporated in the process of workflow scheduling, can optimize the energy consumption of mobile device and the makespan of workflow simultaneously. Simulation results demonstrate significant multi-objective performance improvement of MOWS-DTM over the MOHEFT algorithm and RANDOM algorithm.

Key words: mobile cloud computing; workflow scheduling; multi-objective optimization; genetic algorithm; delay transmission

近年来,随着移动互联网的蓬勃发展,移动终端设备(手机、平板、笔记本电脑等)由于其便携性,极大地方便人们的生活,在网络游戏、图像或视频处理、电子商务、社交网络等方面广受欢迎[1,2].然而,移动终端设备在计算能力和电池能源的限制下,难以发挥和发掘全部的移动性潜力[3−5].

移动云计算的发展,为移动设备提升计算能力和节约电池能源带来了机遇[6].在移动云计算环境中,移动用户可以将应用的一部分任务迁移到资源丰富的云端执行,从而提高应用执行的服务质量.不是在任何情况下迁移任务都会带来服务质量的提高,不合理的任务迁移会引起大量的数据传输,这不仅导致任务执行的延迟,而且会增加移动设备的网络传输能耗.而且,在用户不断移动的过程中,设备的无线网络信号也呈不断变化的状态.由文献[7]可知,当下载或发送一定大小的数据时,网络信号越好,需要的传输时间越少,则移动设备的能耗越少.所以在移动设备发送或者接收数据时,需要特别关注网络信号的状态.

用户的应用都是建模成一般拓扑结构的工作流模型,即应用的任务执行必须满足一定的先序限制关系.相比在移动云计算环境下的并行任务或序列任务调度[8−13],工作流调度具有更大的复杂性和挑战.由于移动云计算环境下的计算资源有限,所以工作流中的任务执行顺序对工作流的完工时间和移动设备能耗都影响很大.因此,如何合理地在移动设备和云端调度工作流的任务,保证工作流的服务质量,并减少移动设备的电池能源消耗,是移动云计算研究的一个重要问题.

鉴于此,本文主要研究移动云计算环境下工作流执行能耗和时间的多目标调度优化问题,提出了面向多目标优化的工作流调度算法 MOWS-DTM.该算法是在遗传算法的基础上,结合了工作流的调度过程,在编码策略中考虑了工作流任务的调度位置和执行排序.针对网络信号对能耗的影响,MOWS-DTM 算法中还融入了延时传输机制 DTM,该机制能够同时有效地优化移动设备的能耗和工作流的完工时间.在仿真实验中,我们还模拟了 MOHEFT 算法和 RANDOM 算法.结果表明,在随机生成的工作流模型下,MOWS-DTM 算法在多目标性能上都要比 MOHEFT 算法和 RANDOM 算法优化.本文的主要贡献包括:(1) 从移动设备执行工作流的能耗和时间出发,提出了一种面向移动云计算环境的多目标工作流调度算法;(2) 采用了基于遗传算法的工作流调度算法,设计了相应的多目标工作流调度编码策略,并且在编码策略中考虑了工作流任务的执行顺序;(3) 在遗传算法中融入了延时传输机制,该机制能够同时有效地优化移动设备的能耗和工作流的完工时间;(4) 实现了大量的仿真对比实验,结果表明,MOWS-DTM 算法具有可行性和优越性,且 MOWS-DTM 算法的多目标性能都要比 MOHEFT 算法和 RANDOM 算法优化.

本文第 1 节介绍相关研究工作.第 2 节介绍移动云计算环境下调度工作流模型及问题描述.第 3 节给出基于遗传算法的多目标工作流调度算法的具体设计方案.第 4 节通过仿真实验对所提算法进行多方面的性能评估.第 5 节对本文工作进行总结和展望.

1 相关工作

近年来,移动云计算的应用得到了飞速的发展,很多学者对该环境下的任务迁移或卸载进行了大量的研究.其中,有相当一部分工作是对简单结构的工作流进行调度,即工作流中的任务只是并行任务或者是序列任务.通

常情况下,工作流的结构非常复杂,是并行、序列等结构的综合,我们称之为一般拓扑结构工作流.下面分别从这两个方面出发,介绍移动云计算环境下的工作流调度研究的现状.

1.1 简单拓扑工作流调度

在移动云计算环境中调度工作流的难点在于网络信号的不断变化,因此,大量的工作是针对并行任务或是序列任务的调度和迁移,来优化应用的执行时间和移动设备的能耗.所用的调度或迁移方法有马尔科夫模型、动态规划、拉格朗日方法等.

Jia 等人[9]分别对线性拓扑结构任务和并行拓扑结构任务在移动云计算环境下的迁移问题进行了研究,并提出了最优线性任务卸载策略和基于负载平衡的启发式并行任务迁移算法.由于移动设备本身的特性,设备的执行能力是不断波动变化的.而且除了实际的网络传输产生能耗,移动通信过程中的尾能耗也不可忽视.基于此,Tong 等人[14]在均衡移动设备能耗和应用服务质量的基础上,提出了应用感知的无线传输调度算法,目的是在满足应用截止时间的基础上,最小化移动设备的能耗.Elgazzar 等人[15]提出了一个基于云辅助的移动服务提供框架,来帮助移动设备交付服务.由于移动系统的资源状态和当前网络信号不断变化,该移动框架能够在满足用户要求的能耗限制条件下支持动态计算卸载.在移动云计算环境下,具有相同下载内容的一组用户组成了合作移动云(collaborative mobile clouds,简称 CMC).它们通过设备到设备的通信,实现用户组成员之间的数据共享.Chang 等人[16]提出了一种通过最优调度数据迁移和无线资源最小化能源有效性和公平性的资源分配算法.Chen[17]研究了移动云计算环境下移动用户之间的计算量卸载决策问题,提出了计算量分解的博弈方法.结果表明,不管是在同构还是异构无线通道,都能够高效地得到纳什均衡.在移动云计算环境下,多个移动设备之间可以进行有效的合作.但是在移动过程中,移动设备之间可能发生连接中断的情况.基于此,Truong-Huu 等人[10]提出了一种动态机会任务卸载算法,该算法是基于马尔科夫决定过程(Markov decision process,简称 MDP),能够让移动用户有效地对需要处理的并行任务进行卸载.Zhang 等人[8,11]提出了一种基于随机网络带宽的序列任务迁移算法,在满足移动应用截止时间的限制下,最小化移动设备的能耗.首先,移动云计算环境下的任务执行被表示成受限最短路径问题;然后,使用低时间复杂度的近似算法 LARAC(lagrangian relaxation based aggregated cost)解决该问题.

1.2 一般拓扑工作流调度

一般拓扑工作流的结构较为复杂,直接在移动云计算框架调度会非常困难,所用方法包括基于 HEFT 的列表式方法、模拟退火、遗传算法等.

Lin 等人[18]研究了在移动云计算环境中的工作流调度问题,提出了一种基于 HEFT 算法的最小延迟调度方案,通过在本地与云平台之间迁移任务来减少移动设备的能耗.Gao 等人[12]使用二次规划对任务分配问题进行建模,并使用了模拟退火算法和贪心自动卸载算法得到近似最优解.Deng 等人[19]提出了一个基于遗传算法的工作流任务计算量卸载策略,该策略把工作流的一部分任务放到云端去执行,达到工作流执行时间和设备能耗的双重优化.由于在移动环境下无线信号有时会失去连接,所以文中还着重研究了容错卸载问题.Zhang 等人[20]提出了移动云计算环境下的工作流调度算法,在保证工作流延时限制的条件下,最小化移动设备的能耗.文献首先使用关键路径对工作流进行分析,一般情况下使用 one-climb 策略求得工作流调度最优解;如果工作流中一些任务有固定的执行位置,则使用 LARAC 算法计算最优解.Guo 等人[21]提出一个能源有效的动态卸载和资源调度策略,来减少能源消耗和缩短完成时间.该策略在考虑任务之间的依赖关系和工作流应用完成的截止时间限制的条件下,最小化能源成本,并通过计算卸载选择、时钟频率控制和传输功率分配这 3 种子算法共同实现.

2 模型及问题描述

本节首先介绍移动云计算下工作流调度的计算资源模型、网络信号模型、延时传输模型和任务执行模型,然后对本文的多目标工作流调度问题进行描述.各部分的详细介绍如下.

2.1 计算资源模型

移动云计算环境下的计算资源主要由移动设备和云端服务器两部分组成,它们之间通过基站作为中间桥梁进行通信.本文使用用一个四元组 $(C_{local},P^c_{local},P^s_{local},P^r_{local})$ 表示移动设备,其中,C_{local}(单位为 MIPS)表示移动设备的计算能力, P^c_{local} (单位为 Kbps)表示移动设备计算功率,而 P^s_{local} 和 P^r_{local} 是移动设备发送数据和接收数据的功率(单位均为 Kbps).由于移动设备的计算能力不足,且电池能源有限,需要将工作流应用中的部分任务迁移到云端执行.令 C_{cloud}(单位为 MIPS)表示云端的计算能力.云端的服务器拥有较大的网络带宽,所以移动设备与云端进行数据交互的局限性,在于移动设备自身在移动过程中的网络信号强度.

2.2 网络信号模型

假设时间系统是离散且按等长间隔变化的,即 τ=0,1,2,…,且每个时间段的时间长度为 T_{slot}.那么通过对网络信号的预测,在每个时间段内的信号强度为一个恒定值 $B(\tau)$.结合离散时间系统,用户在移动过程中的信号状态如图 1 所示.由于不断运动,移动用户的位置呈不断地变化的状态,所以设备与外界连接的网络信号也是不断变化的.在用户移动过程当中,传输数据的网络信号由用户的位置决定[22,23].随着移动网络的发展,手机 GPRS 功能可以非常容易地对用户的位置进行定位.而且,使用百度地图、Google 地图、高德地图等 APP 软件能精确地规划用户出行的路线.根据精确的运动轨迹信息,移动用户可以对路线上的网络信号进行预测.当没有运动路线信息时,可以采用随机路点的方法对位置进行建模[19,24],最终获得每个时间段的网络信号状态.

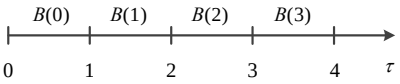


Fig.1 Signal state of each time slot

图 1 每个时间段的信号状态

2.3 延时传输模型

节能问题是移动设备执行应用的一个重要关注点.众所周知,能耗 E 与功率 P 和时间 T 的关系如下:

$$E=P\cdot T \tag{1}$$

假设无线网络带宽为 B,移动用户需要发送数据大小为 D,那么总的时间为 $T=D/B$.因此,当用户需要发送或者接受固定大小的数据时,所消耗的能源与带宽之间是呈非线性的关系.文献[7]测试了智能手机下载 10MB 数据时,不同带宽对应的能源消耗.结论表明,移动设备在发送或接收同样大小的数据时,网络信号越差,将会消耗更多的能源;网络信号越好,只消耗很少的能源.

下面我们介绍一种减少移动设备发送或接收数据能耗的延时传输机制,其思想就是通过推迟发送或接收数据的时间,使得数据只在网络状态良好的时候传输.我们以举例的方式来说明延时传输机制的运作方式.假设移动用户当前需要发送大小为 D=4 的数据,而当前处于时间段 $\tau=h$,如图 2 所示.从图中可以看出,如果在 $\tau=h$ 时开始发送数据,那么需要 $T_h=3\cdot T_{slot}$ 才能发送完成.那么此时的能耗为

$$E_h=P\cdot T_h=P\cdot 3\cdot T_{slot} \tag{2}$$

若推迟数据的发送时间,在时间段 $\tau=h+3$ 时才发送数据,那么只需要 $T_{h+3}=T_{slot}$ 的时间,而此时的能耗为

$$E_{h+3}=P\cdot T_{h+3}=P\cdot T_{slot} \tag{3}$$

通过延时传输机制节省的能耗为

$$\Delta E=E_h-E_{h+3}=2\cdot P\cdot T_{slot} \tag{4}$$

通过延时传输机制,移动设备很大程度上可以有节省能源消耗的机会.但是需要值得注意的是:延时了数据的发送和接收,必然会导致相关任务执行的推迟,从而造成任务完成时间的延后.如本例所示,数据发送或接收完成的时间推迟了一个 T_{slot}.但是,工作流结构中存在一定数量的非关键任务[25,26],在一定范围延长非关键任务的执行时间,并不会对整个工作流的完工时间造成影响.

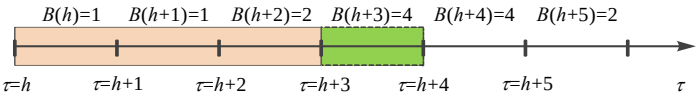


Fig.2 An example of delay transmission mechanism

图 2 延时传输机制示例

令δ为延时系数,它表示移动用户容忍任务最大的延时时间.即如果任务 t_i 在时间段τ时可以传输数据,且传输完成时间为$\tau+l$,那么移动用户容忍该数据在$\tau+l+\delta$之前传输完成.延时系数δ越大,就越有机会减少更多的能耗.至于延时传输机制的实现方法,我们将在算法设计中详细叙述.

2.4 任务执行模型

在移动云计算环境中,工作流的任务可以执行在本地或者云端两种选择.如果任务 t_i 在本地执行,所需要的输入数据可能存在于云端或者移动设备本身,此时只需要把云端的输入数据传输到本地即可.在不考虑延时传输机制的情况下,假设当前时间段是$\tau=h$,此时,移动设备从云端接收数据的传输时间(transmission time)为

$$TT^r_{local}(t_i)=\left\{l:\sum_{\tau=h}^{h+l-1}B(\tau)\geqslant D^s_{cloud}(t_i)\right\}\cdot T_{slot} \tag{5}$$

其中,l 表示接收数据大小为 $D^s_{cloud}(t_i)$ 时所用的时间段,且

$$D^s_{cloud}(t_i)=\sum_{t_j\in pre(t_i)\wedge t_j \text{ on cloud}} D(t_i,t_j) \tag{6}$$

其中,$D(t_i,t_j)$表示前驱任务 t_j 需要传输给任务 t_i 的数据大小.

定义传输跨度时间(span time)为从开始可传输数据到实际传输完成的这一时间跨度.在无传输机制时,任务传输跨度时间与传输时间相等,即

$$ST^r_{local}(t_i)=TT^r_{local}(t_i) \tag{7}$$

在使用延时传输机制时,那么数据接收的跨度时间用公式表示为

$$ST^r_{local}(t_i)=k\cdot T_{slot} \tag{8}$$

公式(8)隐含地说明,数据在 $h+k-1$ 时刻传输完毕.那么实际上传输数据使用的传输时间为

$$TT^r_{local}(t_i)=\left\{k-v:\sum_{\tau=h+v}^{h+k-1}B(\tau)\geqslant D^s_{cloud}(t_i)\right\}\cdot T_{slot} \tag{9}$$

其中,k 和 v 的值都可以通过传输延时机制得到.公式(9)表示在使用延时传输机制时,实际的数据开始传输时间为 $h+v$,发送完成时刻是 $h+k-1$.由于 $k\geqslant l$,所以数据的发送时间延时了.为了便于对上述一些时间定义的理解,图3 中给出了数据传输时间和传输跨度时间的示例.

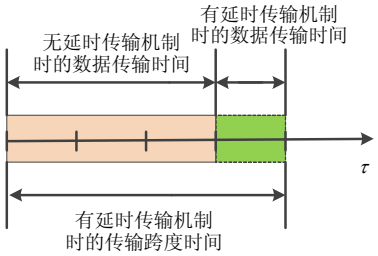


Fig.3 Transmission time and transmission span time

图 3 传输时间和传输跨度时间

传输时间既是数据真正通过无线网络传输使用的时间,也是移动设备消耗能源的时间.那么,移动设备接收数据所需要的能耗表示为

$$E^r_{local}(t_i)=TT^r_{local}(t_i)\cdot P^r_{local} \tag{10}$$

当所需的数据都已经在移动设备上时,任务 t_i 就可以开始在本地执行,执行时间(execution time)为

$$ET^c_{local}(t_i)=W(t_i)/P_{local} \tag{11}$$

那么,在移动设备上执行任务 t_i 所消耗的能源表示为

$$E^c_{local}(t_i)=T^c_{local}(t_i)\cdot P^c_{local} \tag{12}$$

当任务 t_i 迁移到云端执行时,同样需要把移动设备上的数据发送到云端.在无延时传输机制下,此时移动设备发送数据的传输时间(transmission time)为

$$TT^s_{local}(t_i)=\left\{l:\sum_{\tau=h}^{h+l-1}B(\tau)\geqslant D^s_{local}(t_i)\right\}\cdot T_{slot} \tag{13}$$

其中,

$$D^s_{local}(t_i)=\sum_{t_j\in pre(t_i)\wedge t_i,t_j \text{ on local}}D(t_i,t_j) \tag{14}$$

任务传输跨度时间与传输时间相等,即

$$ST^s_{local}(t_i)=TT^s_{local}(t_i) \tag{15}$$

在延时传输机制下,令数据从移动设备上发送的跨度时间为

$$ST^s_{local}(t_i)=k\cdot T_{slot} \tag{16}$$

实际上发送数据所用的时间为

$$TT^s_{local}(t_i)=\left\{k-v:\sum_{\tau=h+v}^{h+k-1}B(\tau)\geqslant D^s_{local}(t_i)\right\}\cdot T_{slot} \tag{17}$$

那么从云端的角度来看,接收数据的跨度时间与移动设备发送跨度时间相等,即

$$ST^r_{cloud}(t_i)=ST^s_{local}(t_i) \tag{18}$$

云端在数据接收完成之后,即可对任务 t_i 开始执行,执行时间为

$$ET^c_{cloud}(t_i)=W(t_i)/P_{cloud} \tag{19}$$

任务 t_i 迁移到云端执行的过程中,移动设备的能耗只在数据发送时产生,则移动设备发送数据的能耗为

$$E^s_{local}(t_i)=TT^s_{local}(t_i)\cdot P^s_{local} \tag{20}$$

经过以上任务执行过程分析,工作流的时间开销主要在于 4 个方面,即任务在本地设备执行的时间、任务在云端执行的时间、移动设备发送数据的跨度时间(即云端接收数据的跨度时间)和移动设备接收数据的跨度时间(即云端发送数据的跨度时间).而移动设备的电池能源消耗主要在本地执行任务消耗的能源、移动设备发送和接收数据所产生的能耗.

2.5 问题描述

本文的多目标工作流调度主要是对工作流执行的完成时间和移动设备的能耗两方面进行考虑.对于给定的工作流 WF,调度的目标是产生一个或多个可行解 $\Gamma=(X,Y)$.其中,$X=\{x_0,x_1,\ldots,x_i,\ldots,x_{n-1}\}$ 表示的是工作流任务调度位置的集合,令 $x_i=0$ 表示任务 t_i 在本地执行,$x_i=1$ 表示任务 t_i 在调度到云端执行;Y 表示工作流中任务执行顺序的排序,该排序必须满足工作流任务之间的先序限制关系.同一个任务在不同的执行顺序下所经历的网络状态会不一样,这就给数据发送或者接收的时间带来很大的差异,会直接影响工作流的调度结果.为了计算工作流的完工时间,需要计算每个任务的开始时间(start time)和结束时间(end time).通常情况下,任务 t_i 的前驱可能存在于移动设备本身或者云端,那么任务 t_i 的开始时间即为所有前驱任务中最大的完成时间,即

$$T_{start}(t_i)=\max_{t_j\in pre(t_i)}T_{end}(t_j) \tag{21}$$

任务执行的位置不同,产生的结束时间也不一样.如果任务 t_i 在移动设备本地执行,那么结束时间为

$$T_{end}(t_i)=T_{start}(t_i)+ST^r_{local}(t_i)+ET^c_{local}(t_i) \tag{22}$$

同理,如果任务在云端执行,则任务 t_i 的结束时间为

$$T_{end}(t_i)=T_{start}(t_i)+ST^r_{cloud}(t_i)+ET^c_{cloud}(t_i) \tag{23}$$

由公式(22)和公式(23)可知,任务的结束时间即为开始时间的基础上,加上数据传输跨度时间和任务执行时间;同一个任务在不同的处理端上的结束时间是不同的.基于以上分析,工作流的完工时间为

$$makespan=T_{end}(t_{exit})+ST^r_{exit}(t_{exit}) \tag{24}$$

因为工作流中的任何任务都可以迁移到云端执行,尾任务也不例外.但是当整个工作流执行完成之后,最终的结果必须在移动设备上.所以如果工作流的尾任务是在云端执行的,那么需要将结果从云端发送到移动设备,这就解释了公式(24)中 $ST^r_{exit}(t_{exit})$ 存在的原因.

- 如果尾任务就在本地执行的,那么 $ST^r_{exit}(t_{exit})=0$;
- 否则,根据延时传输机制,则有:

$$ST^r_{exit}(t_{exit})=k\cdot T_{slot} \tag{25}$$

根据之前描述的移动设备能源消耗的 3 个方面,可以计算出在整个工作流执行过程中移动设备的能耗:

$$energy=\sum_{t_i\in T\wedge t_i \text{ on local}}(E^c_{local}(t_i)+E^r_{local}(t_i))+\sum_{t_i\in T\wedge t_i \text{ on local}}E^s_{local}(t_i) \tag{26}$$

基于以上分析,移动云计算环境下的多目标工作流调度问题可描述为:找到一种调度策略Γ,在这种调度策略下,能够最小化工作流的执行时间和移动设备的能耗,即

$$\text{目标最小化 } F(\Gamma)=(energy,makespan) \tag{27}$$

下面通过一个例子来简单描述移动云计算环境下工作流任务的调度的过程,如图 4 所示.假设移动用户的工作流包含 5 个任务,经过任务迁移决策之后,其中,任务 t_0 和 t_3 在移动设备本地处理,而任务 t_1,t_2 和 t_4 则需要迁移到云端执行.那么当任务 t_0 和 t_3 执行完毕之后,需要将结果数据通过基站发送到云端服务器,使得任务 t_1 和 t_4 具备执行条件.由于任务 t_1 和 t_2 均在云端执行,所以它们之间不再需要数据传输.任务 t_4 是工作流的最后一个任务,当它执行完毕之后,需要通过数据传输的方式把结果返回给移动设备.

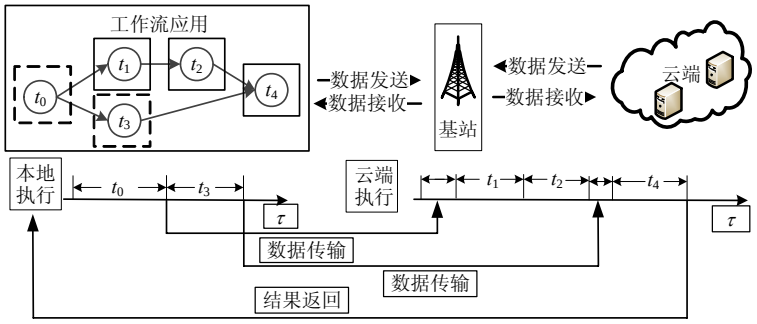


Fig.4 An example of workflow scheduling in mobile cloud environment

图 4 移动云计算环境下的工作流调度示例

3 算法设计

多目标与单目标问题有着本质的区别,多目标优化是对多个有冲突性质的子目标同时优化,冲突意味着优化一个子目标至少必然损害另外一个子目标.公式(28)是多目标优化的方程式:

$$\text{最小化 } F(\bar{x})=(f_1(\bar{x}),f_2(\bar{x}),...,f_m(\bar{x}),...,f_M(\bar{x})) \tag{28}$$

其中, $f_m(\bar{x})$ 表示一个子目标,M 是目标数量.每个子目标都依赖于包括 K 个决策变量的向量 $\bar{x}$,即向量的维数为 K.而且 $\bar{x}\in X$,X 为决策空间.由于子目标之间的冲突关系,一般使用 Pareto 占优(Pareto dominance)的概念来比较解的优劣性[27].

定义 1(Pareto 占优). 两个解 $\vec{x}_1,\vec{x}_2 \in X$,如果解非 $\vec{x}_1$ 占优解 $\vec{x}_2$,当且仅当下式满足:

$$\forall i: f_i(\vec{x}_1) \leqslant f_i(\vec{x}_2) \wedge \exists j: f_j(\vec{x}_1) < f_j(\vec{x}_2) \tag{29}$$

如果一个解 $\vec{x}^*$ 是 Pareto 最优的,那么说明它不被其他任何解占优.

定义 2(Pareto 最优解). 对于公式(28)中描述的多目标优化问题,如果有 $\vec{x}^* \in X$,且不存在 $\forall \vec{x} \in X$,使得 $\vec{x}^* \prec \vec{x}$,则称 $\vec{x}^*$ 为 Pareto 最优解或者非被占优解.

大多数情况下,多目标优化问题的最优解是不存在的.取而代之,多目标优化的目标是得到 Pareto 最优解的集合,称之为 Pareto 最优解集.根据 Pareto 最优解集,可以定义 Pareto 前沿.

定义 3(Pareto 前沿). 所有 Pareto 最优解对应的目标函数值在空间中形成的曲面称为 Pareto 前沿.

总之,多目标优化的目标是得到问题的 Pareto 最优解.实际情况而言,决策人员会需要根据偏好,从 Pareto 最优解集合中选择一个作为问题的最优解.

在工作流的调度过程中,本文主要关注与调度结果密切相关的因数:工作流任务的执行位置执行顺序.在移动云计算环境中,一个任务可以在移动设备本地或者在云端执行,所以不同的执行位置造成不同的设备能耗和工作流完工时间.由于移动云计算环境中的计算资源有限,仅有移动设备和云端服务器两种.当两个任务同时竞争一个资源时,这就带来了资源竞争的问题.因此,本文通过排序任务执行的顺序来解决任务执行过程中的资源竞争,排序中靠前的任务将具有占据资源的优先权.

下面以图 5 为例,阐述不同的任务排序方式对工作流调度的影响.假设一个包含 5 个任务的工作流,调度结果为 3 个任务在本地执行和 2 个任务在云端执行.当工作流任务的执行顺序是 t_0,t_1,t_2,t_3,t_4 时,由于任务 t_2 在云端执行,任务 t_1 完成之后就可以把数据传输到云端,那么任务 t_2 在第 3 个时间段就可以执行.但是当任务的执行顺序是 t_0,t_3,t_1,t_2,t_4 时,由于任务 t_1 的执行延后,同样导致任务 t_2 执行的延后,这样会影响到整个工作流的完工时间.

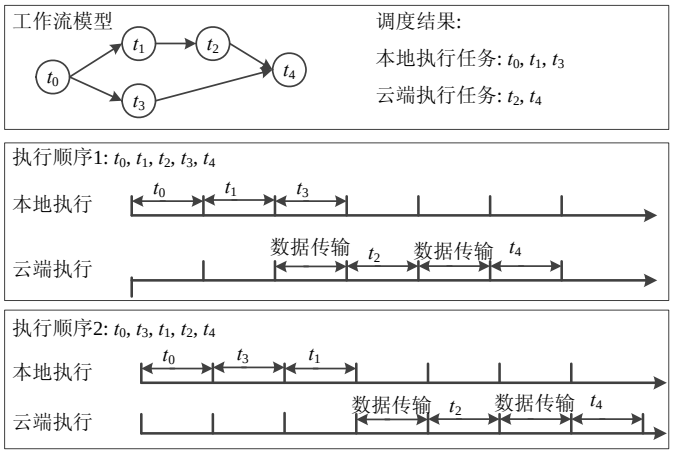


Fig.5 Impact of task execution order on workflow scheduling

图 5 任务执行顺序对工作流调度的影响

目前,很多学者使用进化算法(evolutionary algorithm)解决多目标优化问题[28−32].进化多目标(evolutionary multi-objective)算法可以得到近似最优的 Pareto 解集合,其中每个解表示众多目标之间的一个权衡.在移动云计算环境下的多目标工作流调度中,我们选择进遗传算法作为基础算法,在遗传算法中融入工作流调度问题,设计考虑任务执行位置和任务执行排序的编码策略,以及相应的初始化、选择、交叉、变异等操作;然后,介绍如何使用延时传输机制,在增加任务执行时间的基础上减少能耗;最后,本文提供多目标工作流调度的 Pareto 最优解的评估方法.

3.1 算法框架

本文的多目标工作流调度(multi-objective workflow scheduling,简称 MOWS)算法基于遗传算法实现.遗传

算法最早由美国的 Holland 博士提出[33],是通过生物进化规律演化而来的随机搜索方法.根据实际的应用表明,利用遗传算法来解决多目标优化问题是一个非常有效的方法,并且在自动控制、机器人学、图像处理、人工生命、机器学习等方面获得了广泛的运用.遗传算法模拟了自然选择和遗传中发生的交叉、变异等现象,从初始种群出发,通过选择(selection)、交叉(crossover)和变异(mutation)来产生下一代种群,如此进化下去,直至满足期望的终止条件.多目标与单目标遗传算法的过程相似,主要包括编码、初始化种群、适应度值评价、选择、交叉和变异等 6 个部分.在本文的多目标工作流调度中,适应度值评价分别是公式(24)和公式(26)中对工作流完工时间和移动设备能耗的计算,下面将详细介绍其余五个部分的设计方法.

3.2 编码策略

使用遗传算法实现多目标工作流调度时,需要考虑工作流任务执行的位置和任务执行顺序.假设一个工作流有 n 个任务,用集合 $X=\{x_0,x_1,\ldots,x_i,\ldots,x_{n-1}\}$来表示工作流任务调度位置的集合,令 $x_i=0$ 表示任务 t_i 在本地执行,$x_i=1$ 表示任务 t_i 在调度到云端执行.使用集合 $Y=\{y_0,y_1,\ldots,y_i,\ldots,y_{n-1}\}$表示任务的执行顺序集合,其中,$y_i$ 表示工作流中的某个任务.理论上来说,$y_i\in\{t_0,\ldots,t_i,\ldots,t_{n-1}\}$,但是任务的排序需要满足先序限制关系.因此,源任务 t_0 必须排在第 1 个,即 $y_0=t_0$;尾任务 t_{n-1} 在集合 Y 中必须排在最后一个位置,即 $y_{n-1}=t_{n-1}$.不合理的任务排序,会造成工作流执行的死锁.图 6 中描述的是移动云计算环境下多目标工作流调度的编码方案示例.

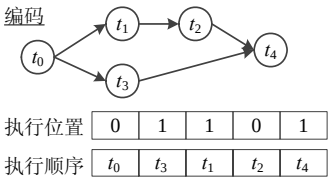


Fig.6 Coding strategy of genetic algorithm

图 6 基于遗传算法的编码方案

3.3 初始化

在多目标工作流调度过程中,当工作流的结构和任务数量很大时,进化算法的搜索空间同样非常巨大,会导致算法收敛很慢.在 MOWS 算法中,为了加速搜索过程,我们对任务执行位置和排序均采用随机方法.

首先,对于一个任务 t_i 的执行位置,产生[0,1]之间的随机数 p_i,如果 $p_i\leqslant 0.5$,那么设置 $x_i=0$;否则,$x_i=1$.用同样的方法遍历所有任务,就可以得到位置集合 X.工作流任务位置初始化的伪代码如图 7 所示,需要注意的是,需要循环 N 次初始化种群中的每个个体,其中,N 为种群的规模.

```
算法 1. 任务位置初始化.
BEGIN
01.  for t_i∈T do
02.      生成[0,1]之间的随机数 p_i;
03.      if p_i≤0.5 then
04.          x_i=0;
05.      else
06.          x_i=1;
07.  end for
END
```

Fig.7 Pseudocode of the initialization of task positions

图 7 任务位置初始化伪代码

然后,对于任务排序初始化,由于任务之间必须满足一定的先序限制关系,所以使用可排序任务集合 S 来记录当前可排序的任务,并从该集合中随机选取一个任务进行排序.其中,可排序任务的定义是该任务没有前驱或者其前驱已经排序完毕.按此方法继续选取下一个任务,直到形成一组可行的任务序列.任务排序初始化伪代码如图 8 所示.同样需要循环 N 次初始化种群中的每个个体,随机产生的任务序列可能会存在重复.

算法 2. 任务排序初始化.
BEGIN
01. $S=\varnothing$; //待排序任务集合
02. $R=\{t_0\}$; //已排序任务
03. $T=T-\{t_0\}$;
04. $y_0=t_0$;
05. *index*=0; //排序的编号
05. while $T\neq\varnothing$ do
06. for $t_i\in T$ do
07. if $pre(t_i)\subset R$ then
08. $S=S+\{t_i\}$;
09. end if
10. end for
11. 随机从集合 S 中选取一个任务 t_i;
12. $y_i=t_{index}$;
13. *index*++
14. $T=T-\{t_i\}$;
15. $R=R+\{t_i\}$;
16. end while
END

Fig.8 Pseudocode of the initialization of tasks order

图 8 任务排序初始化伪代码

3.4 选择算子

选择操作是建立在种群个体适应度评估的基础上,从进化种群中淘汰劣质个体,选择优胜个体.

目前,NSGA-II 中的非支配排序(non-dominated sorting)和拥挤距离排序算法是最广泛使用的保留精英的方法之一[28],因此,本文也采用这两种算法来选择个体生成新的下一代种群.首先,将当代种群和新一代种群进行合并;然后,根据种群中个体之间的支配与非支配关系进行排序分层,即所谓的非支配排序;然后,对每一个分层中的个体计算拥挤距离(crowding distance),并根据该距离进行排序,即为拥挤距离排序;最后,根据个体的排序选择 N 个体作为新的种群.根据非支配分层和计算拥挤距离之后,每个个体 j 会有两个属性:非支配等级 $rank(j)$ 和拥挤距离 $distance(j)$.在选择新种群时,如果个体 i 优于个体 j,即 $i\prec j$,那么必须满足以下条件:

$$rank(i)<rank(j) \text{ or } (rank(i)=rank(j)) \&\& (distance(i)>distance(j)) \tag{30}$$

在选择新种群生成下一代种群时,首先倾向于选择非支配排序等级低的个体.如果两个个体同属于一个非支配层级,那么认为拥挤距离大的个体较优化,作为保留的对象.根据以上操作,直到选择到的新个体数目等于 N 即可,其中,非支配排序和拥挤距离排序的详细算法实现见文献[28].

3.5 交叉算子

交叉算子可以有助于将优良个体的染色体片段遗传给后代,同时起到搜索全局、开拓未知空间的作用.根据设计的编码策略,交叉算子需要分别考虑任务执行位置和执行排序两个方面.在任务执行位置的交叉操作中,首先对种群中的个体进行两两配对,然后采用单点交叉的方式生成新的个体.假设有两个个体,它们表示的任务位置集合分别为 $X_1=\{x_0^1,x_1^1,...,x_i^1,...,x_{n-1}^1\}$ 和 $X_2=\{x_0^2,x_1^2,...,x_i^2,...,x_{n-1}^2\}$,经过交叉操作之后得到新的个体为 X_{12} 和 X_{21} 的过程如图 9 所示,其算法伪代码如图 10 所示.

文献[34]提出了适用于工作流任务执行排序的交叉操作,使用优先约束矩阵来满足任务之间的先序限制关系,从而能够返回合法个体.这里我们对其稍作改进,不需要使用优先约束矩阵也可以实现交叉操作.首先,同位置交叉过程一样,也需要通过生成随机数的方式设定一个交叉点.我们称任务排序集合中第 1 个任务到交叉点之间的任务序列称为匹配区域.假设有两个任务排序集合 Y_1 和 Y_2,接下来把集合 Y_1 的匹配区域加到集合 Y_2 前面,集合 Y_2 的匹配区域加到集合 Y_1 前面,形成两个新的临时新个体.最后,分别对两个临时新个体,从开始到最后进行重复任务移除,这样遗留下来的任务序列即为新个体.在移除重复任务过程中,因为集合 Y_1 和 Y_2 中的任务排序已经满足先序限制关系的,所以移除任务操作不会造成任务执行依赖性的冲突[35].

两个任务排序集合 $Y_1=\{y_0^1,y_1^1,...,y_i^1,...,y_{n-1}^1\}$ 和 $Y_2=\{y_0^2,y_1^2,...,y_i^2,...,y_{n-1}^2\}$,经过交叉操作之后的新个体为 Y_{12}

和 Y_{21} 的过程如图 11 所示,其算法伪代码如图 12 所示.

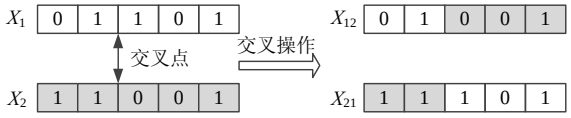


Fig.9 Process of the single crossover of task execution position

图 9 任务执行位置单点交叉过程

算法 3. 位置交叉算法.

BEGIN

01. 生成[0,n−1]之间的随机数 r;
02. X_{12}=X_{21}=∅;
03. for k=0 to r do
04. $X_{12} = X_{12} + x_k^1$;
05. $X_{21} = X_{21} + x_k^2$;
06. end for
07. for k=r+1 to n−1 do
08. $X_{12} = X_{12} + x_k^2$;
09. $X_{21} = X_{21} + x_k^1$;
10. end for

END

Fig.10 Pseudocode of the single crossover of task execution position

图 10 任务执行位置单点交叉算法伪代码

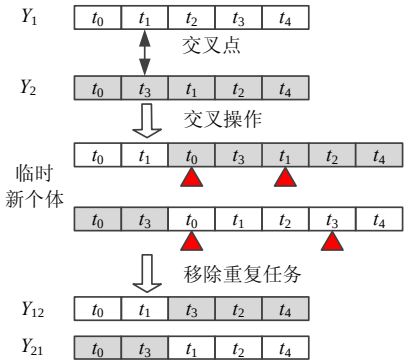


Fig.11 Process of the single crossover of task execution order

图 11 任务执行排序单点交叉过程

算法 4. 排序交叉算法(algorithm 4 order crossover algorithm).

BEGIN

01. 生成[0,n−1]之间的随机数 r;
02. Y_{12}=Y_{21}=∅;
03. for k=0 to r do
04. $Y_{12} = Y_{12} + y_k^1$;
05. $Y_{21} = Y_{21} + y_k^2$;
06. end for
07. Y_{12}=Y_{12}+Y_2;
08. Y_{21}=Y_{21}+Y_1;
09. 移 Y_{12} 中的重复任务;
10. 移除 Y_{21} 中的重复任务;

END

Fig.12 Pseudocode of the single crossover of task execution order

图 12 任务执行排序单点交叉算法伪代码

3.6 变异算子

遗传算法中的变异运算是将个体中的某些基因值做变动,让其他值来替换.任务执行位置的变异操作相对简单,因为个体中的每个基因只有两种值.所以,只需要在个体中随机选择一个基因,并以突变概率 p_m 决定是否产生变异.若确定变异,则对该基因值取反,即 0→1 或 1→0,从而形成一个新的个体.工作流任务执行位置的变异过程如图 13 所示,其算法伪代码如图 14 所示.

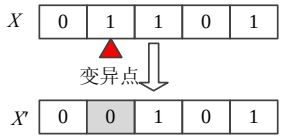


Fig.13 Process of the mutation of task execution position

图 13 任务执行位置变异过程

算法 5. 位置变异算法(algorithm 5 position mutation algorithm).

```
BEGIN
01.  生成[0,1]之间的随机数 p_i;
02.  if p_i≤p_m then
03.      x_i=|x_i−1|;   //值取反
04.  end if
END
```

Fig.14 Pseudocode of the mutation of task execution position

图 14 任务执行位置变异伪代码

类似于任务执行排序的交叉操作一样,排序变异操作的结果也要满足任务之间的先序限制关系.首先,我们从集合 Y 中随机选择一个需要变异的任务 y_i,由于源任务和尾任务的执行位置是固定的,所以需要的变异的任务不能是源任务或是尾任务.然后,我们从集合 Y 的正向进行搜索,如果到某个任务 y_b 时,任务 y_i 的全部前驱都在集合$\{y_0,\ldots,y_b\}$中,则停止搜索;同时,从集合 Y 的逆向进行搜索,如果到某个任务 y_a 时,任务 y_i 的全部后继都在集合$\{y_a,\ldots,y_{n-1}\}$中,则停止搜索.此时,任务 y_i 必定处于集合$Y'=\{y_{b+1},\ldots,y_{a-1}\}$之中,而且任务 y_i 可以处于集合 Y'中的任何一个位置.最后,任务 y_i 排除原来的位置,随机选择一个新的位置进行插入操作.任务执行排序的变异过程如图 15 所示,其算法伪代码如图 16 所示.

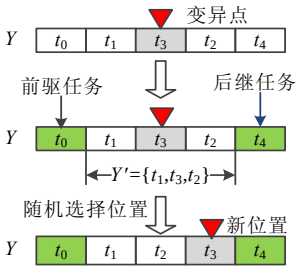


Fig.15 Process of the mutation oftask execution order

图 15 任务执行排序变异过程

算法 6. 排序变异算法(algorithm 6 order mutation algorithm).

```
BEGIN
01.  生成[1,n−2]之间的随机数 r;
02.  for i=0 to n−1 do
03.      找到任务 y_b,使得 pre(y_r)⊂{y_0,…,y_b};
04.  end for
05.  for i=n−1 to 0 do
06.      找到任务 y_a,使得 succ(y_r)⊂{y_a,…,y_{n−1}};
07.  end for
08.  得到集合 Y′={y_{b+1},…,y_{a−1}};
09.  排除任务 y_b 当前位置,在集合 Y′中随机选择新位置;
10.  变异后新个体 Y={y_0,…,y_b}+Y′+{y_a,…,y_{n−1}};
END
```

Fig.16 Pseudocode of the mutation of task execution order

图 16 任务执行排序变异算法伪代码

3.7 延时传输机制

在第 3.3 节中,我们讨论了移动设备的能耗与网络信号之间的关系,结论中说明:可以通过延时传输数据,在增加工作流执行时间的基础上减少设备能耗.本节将介绍用户设置的延时系数为δ时,延时传输机制的具体实现方式.假设任务t_i在时间段$\tau=h$时可以传输数据,其传输完成时间为l个时间段,即在$h+l-1$时刻完成.假设数据传输过程中不能中断,否则会造成传输失败[19].在延时系数为δ的条件下,那么移动用户容忍该数据在 $h+l+\delta-1$之前传输完成.所以需要在$[h,h+l+\delta-1]$的时间段内,确定最佳的开始传输时间,使得数据传输时间最小.已知,从时刻$\tau=h$开始,传输l个时间段可以把数据传输完毕.设此时任务t_i需要传输的数据大小为$D(t_i)$,我们用具体公式表示如下:

$$TT(t_i)=\left\{l:\sum_{\tau=h}^{h+l-1}B(\tau)\geqslant D(t_i)\right\}\cdot T_{slot} \tag{31}$$

当使用传输机制时,我们从 $h+k-1(k\geqslant l+1)$时间段开始逆向遍历来累计数据发送的时间段,找到一个时间点$h+v$,使得从$h+v$时刻发送数据到$h+k-1$时刻能将$D(t_i)$大小的数据发送完成,即

$$TT(t_i)=\left\{k-v:\sum_{\tau=h+v}^{h+k}B(\tau)\geqslant D(t_i)\right\}\cdot T_{slot} \tag{32}$$

因此,当$k-v<l$时,我们才使用延时传输机制.在选择不同的时刻进行逆向遍历时,可能会产生多个可行的结果,我们将选择$k-v$值最小的,且时间跨度也最小的延时方案.需要注意的是,延时传输机制影响工作流执行的完工时间和移动设备的能耗,其伪代码实现如图 17 所示.

算法 7. 延时传输机制(algorithm 7 delay transmission mechanism).
```
BEGIN
01. v*;   //记录最优开始数据传输时刻
02. k*;   //记录最优结束数据传输时刻
03. k*−v*=l;
04. for k=l to l+δ−1 do
05.     for v=l−1 to 0 do   //逆向遍历
06.         if Σ_{τ=h+v}^{h+k} B(τ) ⩾ D(t_i) then
07.             if k−v<k*−v* then
08.                 k*=k;
09.                 v*=v;
10.             end if
11.             break;
12.         end if
13.     end for
14. end for
END
```

Fig.17 Pseudocode of delay transmission mechanism

图 17 延时传输机制伪代码

3.8 性能评估

我们使用 Q-metric,FS-metric 和 S-metric 来衡量多目标算法搜索的 Pareto 前沿的质量[31].当衡量一个多目标算法的性能时,需要从以上 3 个方面综合进行考虑.其中,Q-metric 是用来衡量两个多目标对比算法解的收敛性[36],FS-metric 用来评价算法寻找的 Pareto 最优解的多样性[37],S-metric 用来衡量 Pareto 前沿的均匀性[37].下面分别详细介绍这 3 种评价方法的具体计算方式.

- Q-metric

假设两个对比算法的 Pareto 前沿分别为P_{ref_1}和P_{ref_2},其中,ref_1和ref_2表示对比算法 1 和对比算法 2.

令$Y=P_{ref_1}\cup P_{ref_2}$,$\Phi=Y\cap P_{ref_1}$和$\Omega=Y\cap P_{ref_2}$.定义 $Q(ref_1,ref_2)=|\Phi|/|Y|$,$Q(ref_2,ref_1)=|\Omega|/|Y|$,此时,$Q(ref_1,ref_2)$与$Q(ref_2,ref_1)$的和为 1.如果$ref_1$解的收敛性比$ref_2$的好,则当且仅当:

$$Q(ref_1,ref_2)>Q(ref_2,ref_1)\text{ or }Q(ref_1,ref_2)>0.5 \tag{33}$$

- FS-metric

其计算方法如下式所示:

$$FS=\sqrt{\sum_{i=1}^{m}\min_{(x_0,x_1)\in P_{ref1}\times P_{ref1}}(f_i(x_0)-f_i(x_1))^2} \tag{34}$$

公式(34)表明,FS-metric 的值越大,得到的 Pareto 最优解的多样性越好,即最优解广泛分布在 Pareto 前沿上.

- S-metric

其计算公式如下:

$$S=\sqrt{\frac{1}{N_P}\sum_{i=1}^{N_P}(d_i'-\overline{d}')^2} \tag{35}$$

其中,N_P是 Pareto 最优解的个数,$\overline{d}'=\sum_{i=1}^{N_P}d_i'\Big/N_P$,$d_i'$表示 Pareto 前沿中某个解与其最近的一个解之间的距离.如果 S-metric 的值越小,那么所求得的 Pareto 最优解越均匀.

4 仿真实验

本节对本文提出的 MOWS-DTM 算法的性能进行仿真实验.我们采用随机生成工作流的方式,其中,工作流中每个任务的工作量服从[1,10]之间均匀分布(单位为 GHz);每个任务的输出数据大小服从[1,10]均匀分布(单位为 Mb);在离散时间轴上,每个时间段的网络信号服从[1,10]的均匀分布(单位为 Mb);并且,移动设备的计算能力、计算功率、数据发送功率、数据接收功率以及云端计算能力见表 1[8].

Table 1 Performance parameters of mobile device and cloud

表 1 移动设备和云端性能参数

移动设备	计算能力	C_{local}=2.4GHz
	计算功率	$P_{local}^{c}=0.5$W
	发送数据功率	$P_{local}^{s}=0.1$W
	接收数据功率	$P_{local}^{r}=0.05$W
云端服务器	计算能力	C_{cloud}=3GHz

除了 MOWS-DTM 算法以外,我们还分别仿真了 MOHEFT 和 RANDOM 算法.

- MOHEFT 算法

MOHEFT 算法是一个基于列表式调度的多目标工作流调度算法[38],是对 HEFT 算法进行进行了多目标的改进.其思想是,每次通过穷举映射一个任务可能的执行位置来扩展目标解集合.这样扩展后,解的个数可能会超过想要的解的个数,然后使用拥挤距离排序的方式选择最优的解.

- RANDOM 算法

RANDOM 算法就是对于任务的执行位置和排序都采用随机的方式.我们将采用第 3.3 节中的算法 1 和算法 2 对工作流任务的执行位置和排序进行随机.

另外,在遗传算法中,交叉操作和变异操作的概率为 1,种群中的个体数目为 N=50.在实验中,RANDOM 算法和 MOHEFT 算法设置的初始目标解的个数均为 50.不作特殊说明,随机工作流的任务数为 50,延时系数δ=5.在展示结果时,工作流完工时间的单位是 Time slot,即为时间段的个数;能耗的单位是 $W\times Time\ slot$,即功率和时间的乘积.下面我们分别从工作流任务数变化、算法性能评估、传输数据变化、工作量变化和延时系数变化这 5 个方面,对 MOWS-DTM 算法的性能进行实验仿真.

4.1 任务数变化的影响

在本节,我们主要研究 MOWS-DTM 算法、MOHEFT 算法和 RANDOM 算法在不同工作流任务数量时能耗和时间方面的比较,我们模拟任务数分别为 50,100,150 和 200 的情况,如图 18 所示.首先可以看到:随着任务

数量的增加,工作流的完工时间和移动设备的能耗都整体增大了;而且,MOWS-DTM 算法得到 Pareto 最优解是最优的,体现了遗传算法在解决多目标问题的优越性.MOHEFT 算法通过每次穷举当前任务的映射方案,然后使用拥挤距离选择最优的任务序列,所以在能耗和时间上要比 RANDOM 算法优化.但是 MOHEFT 算法中任务执行排序是固定不变的,而且在最优解选择时并不能保证最终的 Pareto 解是最好的,所以在能耗和时间方面,要比 MOWS-DTM 算法要差.RANDOM 算法通过随机选择任务执行位置和排序,所以得到的 Pareto 解的个数不但不多,而且优化性差.

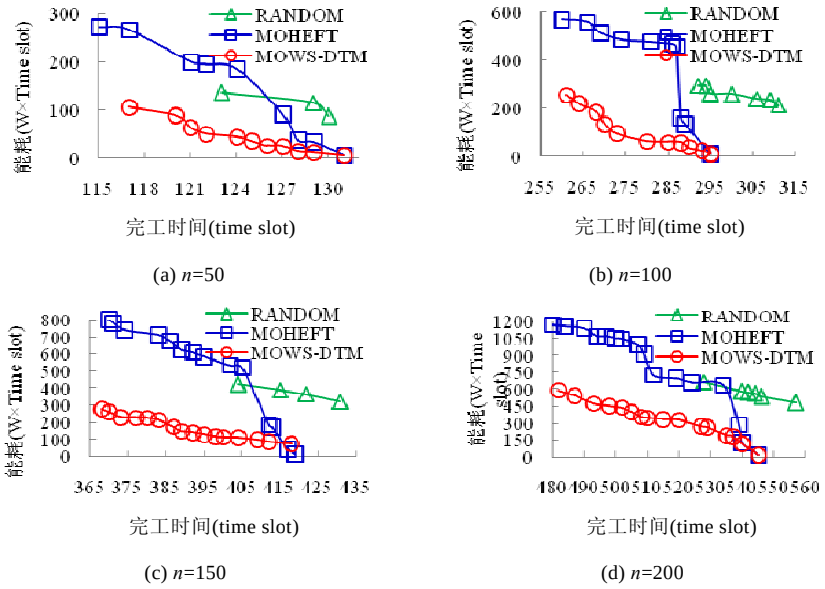


Fig.18 Impact of the number of tasks on energy and makespan

图 18 任务数变化对能耗和完工时间的影响

4.2 算法性能评估

本节在第 4.1 节的基础上,分别计算了不同任务数时的 MOWS-DTM 算法、MOHEFT 算法和 RANDOM 算法的 Q-metric,FS-metric 和 S-metric 的值,来比较 3 种算法多目标的性能.其中,在计算 Q-metric 时,我们把 3 种算法进行两两对比,并用 TURE 和 FALSE 来表示是否优化.对于 Q-metric 值比较,从表 2~表 5 中看出,MOWS-DTM 比其他两个算法都更优.由定义可知,Pareto 最优解越好,则对应的 Q-metric 值越大.可以从图 18 中看出,MOWS-DTM 算法的 Pareto 最优解优于其他两个算法,所以 MOWS-DTM 算法的 Q-metric 值最大.同理,MOHEFT 算法得到的 Pareto 最优解优于 RANDOM 算法,所以其 Q-metric 要大于 RANDOM 算法.相对于 MOHEFT 算法和 RANDOM 算法,MOWS-DTM 算法的 Pareto 最优解分布广泛,具有多样性,所以其 FS-metric 值较大.对于 S-metric 值,MOWS-DTM 算法的 Pareto 解分布间隔最均匀,所以值最小.总体来说,MOWS-DTM 算法在 3 项多目标性能指标上都要较 MOHEFT 算法和 RANDOM 算法优化,其次是 MOHEFT 算法.RANDOM 算法的随机性特别强,难以得到稳定而优化的多目标解.

Table 2 Multi-Objective performance comparisons while n=50

表 2 任务数 n=50 时的多目标性能比较

Q-metric	RANDOM	MOHEFT	MOWS-DTM
MOHEFT	TURE	–	–
MOWS-DTM	TURE	TURE	–
FS-metric	1.41	4.12	20.02
S-metric	15.75	5.67	3.83

Table 3 Multi-Objective performance comparisons while n=100

表 3 任务数 n=100 时的多目标性能比较

Q-metric	RANDOM	MOHEFT	MOWS-DTM
MOHEFT	TURE	–	–
MOWS-DTM	TURE	TURE	–
FS-metric	1.41	2.23	8.08
S-metric	7.48	13.68	5.89

Table 4 Multi-Objective performance comparisons while n=150

表 4 任务数 n=150 时的多目标性能比较

Q-metric	RANDOM	MOHEFT	MOWS-DTM
MOHEFT	TURE	–	–
MOWS-DTM	TURE	TURE	–
FS-metric	1.41	12.04	25.0
S-metric	6.70	7.5	4.59

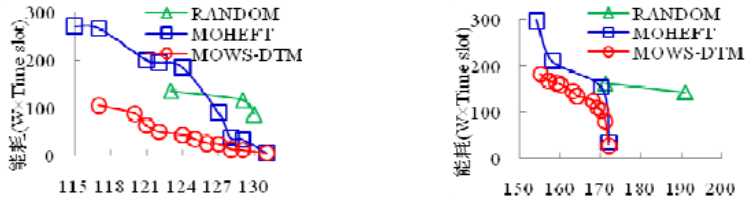
Table 5 Multi-Objective performance comparisons while n=200

表 5 任务数 n=200 时的多目标性能比较

Q-metric	RANDOM	MOHEFT	MOWS-DTM
MOHEFT	TURE	–	–
MOWS-DTM	TURE	TURE	–
FS-metric	4.12	5.32	8.25
S-metric	14.64	9.22	8.58

4.3 传输数据变化的影响

本节分别仿真了工作流任务的输出数据大小在[1,10]和[11,20]两种范围变化时,3 种算法在能耗和时间上的变化情况.如图 19 中所示,任务的输出数据大小在[11,20]范围变化时的能耗和时间整体上要比在[1,10]范围变化时的能耗和时间要大.这是因为任务的输出数据大小对工作流执行过程中的数据接收和发送有较大的影响,在同样的调度策略下,较大的数据传输会产生较长的时间,会导致整个工作流的完工时间变长.同样,较多的数据传输会给移动设备带来更多的传输能耗,所以移动设备的能耗也增加了.但是在这两种输出数据大小变化范围内,MOWS-DTM 算法的 Pareto 解都要比 MOHEFT 算法和 RANDOM 算法更优.所以,输出数据大小会对工作流的执行时间和移动设备的能耗造成一定的影响,但对本文提出的 MOWS-DTM 算法不产生影响.



(a) 输出数据变化范围[1,10] (b) 输出数据变化范围[11,20]

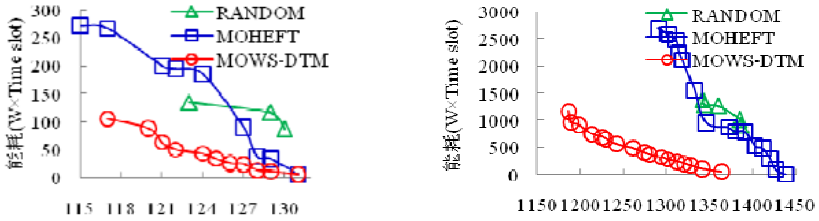
Fig.19 Impact of the number of transmission data on energy and makespan

图 19 传输数据大小变化对能耗和完工时间的影响

4.4 工作量变化的影响

图 20 研究了任务工作量大小变化对能耗和时间的影响,其中仿真了工作量变化范围为[1,10]和[1,100]的情况.从图中可以看出,工作量在[1,100]范围时的能耗和时间整体要比在[1,10]范围时的能耗和时间要大.这是因为更大的工作量带来更大任务执行时间,那么工作流的完工时间会变长.当任务在本地执行时,较大工作量的任务会消耗移动设备更多的电池能耗,所以在执行工作流时移动设备的能耗也增加了.同样,MOWS-DTM 算法的

Pareto 解都要比 MOHEFT 算法和 RANDOM 算法更优.综上所述,任务工作量变化会对能耗和时间造成很大的影响,但我们提出的 MOWS-DTM 算法仍然适用.



(a) 工作量变化范围[1,10] (b) 工作量变化范围[1,100]

Fig.20 Impact of the number of workload on energy and makespan

图 20 工作量大小变化对能耗和完工时间的影响

4.5 延时系数变化的影响

本节我们仿真了不同延时系数时能耗和时间的变化情况,以此来验证延时传输机制的可行性.我们一共仿真了延时系数δ=1,δ=3,δ=5 和δ=7 这 4 种情况.从图 21 中可以看出,随着延时系数的不断增大,对应 Pareto 最优解的整体能耗呈下降趋势.这是因为使用延时传输机制时,移动用户可以有机会选择网络信号强的时候传输数据,这样实际的数据传输时间变小,从而减少了能耗.但是移动用户选择网络信号强的时候传输数据,那么会造成云端或者本身接收到数据的时间延后,这样会增大整个工作流的执行时间.4 种延时系数下的 MOWS-DTM 算法都找到一个能耗最小的方案,即全部工作流任务迁移到云端执行的情况,如图 21 中最右端的解.该解相当于一个极限点,当延时系数δ=1 和δ=3 时,MOWS-DTM 算法在两种情况下的极限点相等.这是由于当延时系数差距较小时,延时传输机制在极限点处没有发挥太大的作用.而当延时系数δ=5 和δ=7 时,延时机制产生作用,极限点的能耗减少.

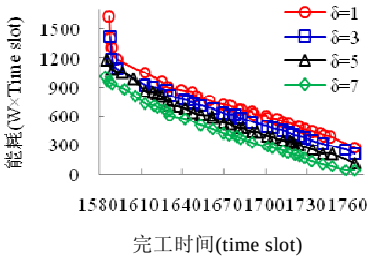


Fig.21 Impact of delay coefficient on energy and makespan

图 21 延时系数变化对能耗和完工时间的影响

而且,在工作流任务执行的过程中,存在相当数量的非关键任务.在一定范围内延长非关键任务的执行时间,并不会对工作流的完工时间带来影响.所以,我们的延时传输机制能充分利用非关键任务的执行时间,在减少能耗的同时,并不对工作流的完成时间造成影响.

总之,延时传输机制能够有机会让移动设备使用更少的能耗,在 Pareto 最优解上表现为解空间的变化.移动用户需要根据对能耗和时间的偏好,通过设置合适的延时系数来满足要求.

5 总结与展望

本文针对移动云计算环境下工作流执行的时间和移动设备能耗问题,提出了面向多目标优化的工作流调

度算法 MOWS-DTM.首先考虑了移动云计算环境下的资源模型、网络模型等.针对数据传输和发送时的能耗与网络信号之间的关系,提出了延时传输的思想.MOWS-DTM 算法是建立在遗传算法的基础上的,因此我们设计了工作流任务调度位置和执行顺序的编码策略.基于该编码策略,在遗传算法的遗传操作中,我们分别对交叉和变异算子进行了详细的叙述.仿真结果表明,相比 MOHEFT 算法和 RANDOM 算法,MOWS-DTM 算法在多目标的性能上更优越.

本文所提算法的性能在仿真实验中已经得到证实,然而本文工作仍有有待改善的地方.如在延时传输机制中,我们假设传输数据时网络是不可中断的.因此,在网络传输中断的情况下如何恢复数据传输,是未来的工作之一.此外,研究可中断的数据传输机制来减少能源消耗,也是一个非常实用的技术.

References:

[1] Cohen J. Embedded speech recognition applications in mobile phones: Status, trends, and challenges. In: Proc. of the IEEE Int'l Conf. on Acoustics, Speech and Signal (ICASSP 2008). 2008. 5352−5355. [doi: 10.1109/ICASSP.2008.4518869]

[2] Soyata T, Muraleedharan R, Funai C, Kwon M, Heinzelman W. Cloud-Vision: Real-Time face recognition using a mobile-cloudlet cloud acceleration architecture. In: Proc. of the 17th IEEE Symp. on Computers and Communications (ISCC 2012). 2012. 59−66. [doi: 10.1109/ISCC.2012.6249269]

[3] Kumar K, Liu J, Lu YH, Bhargava B. A survey of computation offloading for mobile systems. Mobile Networks and Applications, 2013,18(1):129−140.

[4] Liu F, Shu P, Jin H, Ding L. Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications. IEEE Wireless Communications, 2013,20(3):14−22. [doi: 10.1109/MWC.2013.6549279]

[5] Yang B, Feng DG, Qin Y, Zhang YJ. Secure access scheme of cloud services for trusted mobile terminals using TrustZone. Ruan Jian Xue Bao/Journal of Software, 2016,27(6):1366−1383 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4611.htm [doi: 10.13328/j.cnki.jos.004611]

[6] Cui Y, Song J, Miao CC, Tang J. Mobile cloud computing research progress and trends. Chinese Journal of Computers, 2017, 40(20):273−295 (in Chinese with English abstract).

[7] Shu P, Liu FM, Jin H, Chen M, Wen F, Qu YP, Li B. eTime: Energy-Efficient transmission between cloud and mobile devices. In: Proc. of the IEEE Int'l Conf. on Computer Communications (INFOCOM 2013). 2013. 195−199. [doi: 10.1109/INFCOM.2013.6566762]

[8] Zhang W, Wen Y, Wu DO. Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans. on Wireless Communications, 2015,14(1):81−93. [doi: 10.1109/TWC.2014.2331051]

[9] Jia M, Cao J, Yang L. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In: Proc. of the IEEE Conf. on Computer Communications Workshops (INFOCOM Workshops 2014). 2014. 352−357. [doi: 10.1109/INFCOMW.2014.6849257]

[10] Truong-Huu T, Tham CK, Niyato D. To offload or to wait: An opportunistic offloading algorithm for parallel tasks in a mobile cloud. In: Proc. of the IEEE 6th Int'l Conf. on Cloud Computing Technology and Science (CloudCom 2014). 2014. 182−189. [doi: 10.1109/CloudCom.2014.33]

[11] Zhang W, Wen Y, Wu DO. Energy-Efficient scheduling policy for collaborative execution in mobile cloud computing. In: Proc. of the IEEE Conf. on Computer Communications (INFOCOM 2013). 2013. 190−194. [doi: 10.1109/INFCOM.2013.6566761]

[12] Gao B, He L, Lu X, Chang C, Li K, Li K. Developing energy-aware task allocation schemes in cloud-assisted mobile workflows. In: Proc. of the IEEE Int'l Conf. on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM) 2015. 1266−1273. [doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.188]

[13] Nir M, Matrawy A, St-Hilaire M. An energy optimizing scheduler for mobile cloud computing environments. In: Proc. of the IEEE Conf. on Computer Communications Workshops (INFOCOM Workshops 2014). 2014. 404−409. [doi: 10.1109/INFCOMW.2014.6849266]

[14] Tong L, Gao W. Application-Aware traffic scheduling for workload offloading in mobile clouds. In: Proc of the IEEE Conf. on Computer Communications (INFOCOM 2016). 2016. 1–9. [doi: 10.1109/INFOCOM.2016.7524520]

[15] Elgazzar K, Martin P, Hassanein H. Cloud-Assisted computation offloading to support mobile services. IEEE Trans. on Cloud Computing, 2016,4(3):279–292. [doi: 10.1109/TCC.2014.2350471]

[16] Chang Z, Gong J, Zhou Z, Ristaniemi T. Resource allocation and data offloading for energy efficiency in wireless power transfer enabled collaborative mobile clouds. In: Proc. of the IEEE Conf. on Computer Communications Workshops (INFOCOM Workshops) 2015. 336–341. [doi: 10.1109/INFCOMW.2015.7179407]

[17] Chen X. Decentralized computation offloading game for mobile cloud computing. IEEE Trans. on Parallel and Distributed Systems, 2015,26(4):974–983. [doi: 10.1109/TPDS.2014.2316834]

[18] Lin X, Wang YZ, Xie Q, Pedram M. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Trans. on Services Computing, 2015,8(2):175–186. [doi: 10.1109/TSC.2014.2381227]

[19] Deng S, Huang L, Taheri J, Zomaya AY. Computation offloading for service workflow in mobile cloud computing. IEEE Trans. on Parallel and Distributed Systems, 2015,26(12):3317–3329. [doi: 10.1109/TPDS.2014.2381640]

[20] Zhang W, Wen Y. Energy-Efficient task execution for application as a general topology in mobile cloud computing. IEEE Trans. on Cloud Computing, 2018,6(3):708–719. [doi: 10.1109/TCC.2015.2511727]

[21] Guo S, Xiao B, Yang Y, Yang Y. Energy-Efficient dynamic offloading and resource scheduling in mobile cloud computing. In: Proc. of the IEEE Conf. on Computer Communications (INFOCOM). 2016. 1–9. [doi: 10.1109/INFOCOM.2016.7524497]

[22] Taheri J, Zomaya AY. On the performance of static and dynamic location management strategies in mobile computing. Int'l Journal of Foundations of Computer Science, 2011,22(3):519–546.

[23] Taheri J, Zomaya AY, Iftikhar M. Fuzzy online location management in mobile computing environments. Journal of Parallel and Distributed Computing, 2011,71(8):1142–1153.

[24] Rango FD, Guerriero F, Fazio P. Link-Stability and energy aware routing protocol in distributed wireless networks. IEEE Trans. on Parallel and Distributed Systems, 2012,23(4):713–726. [doi: 10.1109/TPDS.2010.160]

[25] Wang XJ, Xiong X. Research of workflow scheduling based on improved genetic algorithm. Computer Technology and Development, 2013,23(7):108−111 (in Chinese with English abstract).

[26] Li ZJ, Ge JD, Hu HY, Song W, Hu H, Luo B. Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. IEEE Trans. on Services Computing, 2018,11(4):713–726. [doi: 10.1109/TSC.2015.2466545]

[27] Huang Q, Su S, Li J. Enhanced energy-efficient scheduling for parallel applications in cloud. In: Proc. of the 12th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGrid 2012). 2012. 781–786. [doi: 10.1109/CCGrid.2012.49]

[28] Yuan Y, Xu H, Wang B, Yao X. A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans. on Evolutionary Computation, 2016,20(1):16–37. [doi: 10.1109/TEVC.2015.2420112]

[29] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002,6(2):182–197. [doi: 10.1109/4235.996017]

[30] Alvarez-Benitez JE, Everson RM, Fieldsend JE. A MOPSO algorithm based exclusively on Pareto dominance concepts. In: Proc. of the Int'l Conf. on Evolutionary Multi-Criterion Optimization (EMO 2005). LNCS 3410. Springer-Verlag, 2015. 459–473.

[31] Li H, Zhang Q. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Trans. on Evolutionary Computation, 2009,13(2):284–302. [doi: 10.1109/TEVC.2008.925798]

[32] Wei J, Zhang M. A memetic particle swarm optimization for constrained multi-objective optimization problems. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC 2011). 2011. 1636–1643. [doi: 10.1109/CEC.2011.5949811]

[33] Li M, Yang S, Liu X. Shift-Based density estimation for Pareto-based algorithms in many-objective optimization. IEEE Trans. on Evolutionary Computation, 2014,18(3):348–365. [doi: 10.1109/TEVC.2013.2262178]

[34] Holland J. Adaptation in Natural and Artificial Systems. The University of Michigan Press, 1975.

[35] Zhu Z, Zhang G, Li M, Liu X. Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. on Parallel and Distributed Systems, 2016,27(5):1344−1357. [doi: 10.1109/TPDS.2015.2446459]

[36] Wang Y, Dang C. Improving multiobjective evolutionary algorithm by adaptive fitness and space division. In: Proc. of the Int'l Conf. on Advances in Natural Computation (ICNC 2005). LNCS 3612. Springer-Verlag, 2005. 392–398.

[37] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. on Evolutionary Computation, 1999,3(4):257−271. [doi: 10.1109/4235.797969]

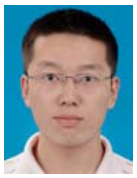
[38] Durillo JJ, Prodan R, Fard HM. Moheft: A multi-objective list-based method for workflow scheduling. In: Proc. of the IEEE 4th Int'l Conf. on Cloud Computing Technology and Science (CloudCom 2012). 2012. 185−192. [doi: 10.1109/CloudCom.2012.6427573]

附中文参考文献:

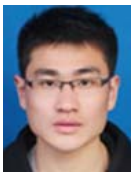
[5] 杨波,冯登国,秦宇,张英骏.基于 TrustZone 的可信移动终端云服务安全接入方案.软件学报,2016,27(6):1366−1383. http://www.jos.org.cn/1000-9825/4611.htm [doi: 10.13328/j.cnki.jos.004611]

[6] 崔勇,宋健,缪葱葱,唐俊.移动云计算研究进展与趋势.计算机学报,2017,40(20):273−295.

[25] 王晓军,熊潇.基于改进遗传算法的工作流调度研究.计算机技术与发展,2013,23(7):108−111.



周业茂(1985－),男,山西五台人,博士生,工程师,主要研究领域为云计算,工作流技术,过程挖掘,文本挖掘,数据仓库.



李传艺(1991－),男,博士,助理研究员,主要研究领域为工作流技术,过程挖掘,文本挖掘,机器学习.

李忠金(1986－),男,博士,讲师,主要研究领域为云计算,工作流技术,过程挖掘,文本挖掘.

周筱羽(1985－),女,博士,工程师,主要研究领域为软件工程,形式化方法,模型检验.

葛季栋(1978－),男,博士,副教授,CCF 高级会员,主要研究领域为工作流技术,过程挖掘,文本挖掘,软件协同,软件过程技术.

骆斌(1967－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为软件工程,工作流技术,过程挖掘.