计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160757
54(9):1909-1919,2017

移动云计算环境下任务调度的多目标优化方法

胡海洋　刘润华　胡　华
(杭州电子科技大学计算机学院　杭州　310018)
(复杂系统建模与仿真教育部重点实验室(杭州电子科技大学)　杭州　310018)
(huhaiyang@hdu.edu.cn)

Multi-Objective Optimization for Task Scheduling in Mobile Cloud Computing

Hu Haiyang, Liu Runhua, and Hu Hua
(*College of Computer Science, Hangzhou Dianzi University, Hangzhou* 310018)
(*Key Laboratory of Complex Systems Modeling and Simulation (Hangzhou Dianzi University), Ministry of Education, Hangzhou* 310018)

Abstract Mobile cloud computing provides effective help for mobile users to migrate their workflow tasks to cloud servers for executing due to the mobile device's limited hardware capability and battery energy carried. When scheduling workflow tasks between mobile devices and cloud servers, it needs to consider both the energy consumed by the mobile device and the total amount of time needed for the workflow application. Traditional methods for scheduling workflow tasks in mobile cloud computing usually address only one of two issues: saving energy consumption or minimizing the time needed. They fail to provide methods for jointly optimizing the time and energy consumption at the same time. Based on the relations of workflow tasks, the time needed in the workflow application is computed due to the tasks scheduling between the cloud servers and the mobile devices that use the technique of dynamic voltage and frequency scaling. The energy consumption for executing tasks on the cloud server and mobile devices are modeled and computed. The scheduling scheme and objective function for jointly optimizing the time needed and energy consumption are proposed. Algorithms based on the simulated annealing are designed for the mobile devices. Their time complexities are analyzed. Extensive experiments are conducted for comparing the proposed methods with other research works, and the experimental results demonstrate the correctness and effectiveness of our approaches.

Key words mobile cloud computing; workflow; task scheduling; simulated annealing; multi-objective optimization

摘　要 移动云计算技术可帮助移动用户在执行工作流任务时将一些任务迁移至云端服务器执行,从而节省移动设备的电池能耗,并提高计算能力.传统研究工作在进行移动云计算环境中的任务调度时缺乏对能耗和运行时间的联合优化.为了实现有效的任务调度,基于工作流图中任务执行的先后关系,分析了采用动态电压频率调节技术的移动设备处理器执行工作流任务的运行时间与能耗,并考虑了将任务

收稿日期:2016-10-24;**修回日期**:2017-02-06
基金项目:国家自然科学基金项目(61572162,61272188);南京大学计算机软件新技术国家重点实验室开放基金项目(KFKT2014B15);江苏省自然科学基金项目(BK20131277)
This work was supported by the National Natural Science Foundation of China(61572162, 61272188), the Foundation of State Key Laboratory for Novel Software Technology of Nanjing University (KFKT2014B15), and the Natural Science Foundation of Jiangsu Province of China(BK20131277).

通过无线信道迁移到云端服务器执行所需的时间,给出了能耗与执行时间联合优化的任务调度模型和目标方程.提出基于模拟退火算法的任务调度方法,分析了算法时间复杂度,进行了系统性的对比实验,评估了所提出方法的正确性和有效性.

关键词 移动云计算;工作流;任务调度;模拟退火;多目标优化

中图法分类号 TP311

无线移动计算技术的普及为跨地域的企业内部工作流管理与移动办公提供了方便:借助于便携式移动计算设备,跨地域的各类工作人员可随时进行灵活的无线移动办公与协作支持,那些常需要出勤在外的企业管理与工作人员也能通过各类移动计算设备及时进入企业的工作流管理系统中审查日志、调配资源、交流协作和完成任务.

尽管智能手机和无线通信技术能力在日益进步,由于受电池容量、存储与计算能力等方面的约束,与传统服务器及桌面设备相比,移动计算设备的硬件能力仍显得非常有限[1].当执行数据密集型(data-intensive)或资源密集型(resource-intensive)的计算任务时,很难保证相应的服务质量.近年来移动云计算技术(mobile cloud computing)的出现[2-4]为克服上述困难提供了一个良好的解决方案:它提供了基于云端资源共享和增强移动计算两者相统一的平台[5-7],可允许计算任务、数据存储和海量信息处理被卸载(offloaded)到云端服务器执行以提高服务的可靠性和可用性,并减少移动设备端能量与计算资源的消耗.目前许多移动应用已经采用移动云计算技术来提供增强的移动服务[8-9].

然而,现有面向移动云计算环境的任务调度方法主要针对如何优化任务完成的总时间[10-13]或者如何优化移动客户端的能量消耗[14-17],很少有研究工作综合考虑这2方面的联合优化.本文针对这一多目标优化问题展开研究,给出调度模型与目标优化方程,并设计基于模拟退火方法的优化算法,来对工作流任务完成的总时间和移动客户端能量消耗这2方面进行综合优化.

1 相关工作

工作流任务调度问题已得到了广泛的研究,各种启发式算法被相继提出[10-17].这些工作可分为2类:1)优化应用程序的总完成时间[10-13];2)优化整体的能源消耗[14-17].

在缩短工作流应用的总完成时间方面,文献[10]主要通过任务优先级对异构处理器环境中的应用程序实现高速的任务调度.文献[11]首先通过快速确定算法进行初始化,然后通过迭代方法不断地改进目前的解决方案.文献[12]采用增量贪心策略并开发了一个运行系统,它能自适应地为移动交互感知应用程序进行任务卸载和并行执行,从而减少应用程序的完成时间.文献[13]提出了一种方法,可在移动设备和云环境中的最大吞吐量之间,对数据流应用的任务分配问题进行优化.

在减少总体的能量消耗方面,文献[14]提出了在执行周期性任务的计算机系统中降低能耗并使能耗达到最少的问题,在该方法中,研究者们假设任务的周期是足够大的,任务之间的空隙时间是可以用来降低能耗的.文献[15]将工作流任务调度问题视为一个最大流/最小割的问题来处理,通过优化在移动设备和云计算环境中执行的每个模块任务从而使能量消耗最小化.文献[16]在文献[10]的基础上进行了扩展,对异构处理器环境中的能源消费问题和任务完成时间都进行了阐述,但文献[16]中所给的任务调度算法不能保证调度结果一定都能满足用户所给的时间约束条件.文献[17]中提出了一种根据网络通信环境来进行简单直接的卸载策略以减少能源的消耗.

文献[18]在文献[16]基础上,考虑了如何满足时间约束条件下来进行任务调度,并通过动态电压频率调节(dynamic voltage and frequency scaling, DVFS)来减少任务之间的空隙时间从而降低能源消耗,但是该算法中使用到的动态电压频率调节技术是离散的,这对能耗减少的效果并不是特别明显,同时该算法对如何优化工作流任务的总完成时间也没有给出相应的算法,仅仅满足用户所给的时间约束条件.

综上所述,在任务分配时,大多数的任务分配算法仅考虑在最大截止时间条件下减少能耗,没有将完成时间也作为优化的一个目标.然而,为了减少能耗,会使得用户较多地去选择低功率的处理器或者通过技术手段动态降低处理器的运行功率,这样使

得工作流任务的总完成时间相应增加.为了解决这样的多目标优化问题,本文提出一种基于模拟退火算法的时间和能耗联合优化的工作流任务调度算法,该算法通过连续动态电压调节技术能在满足完成时间约束条件情况下使工作流应用的完成时间和能量消耗都得到优化,从而减少移动设备的能量消耗并提高工作流应用的执行速度.

2 移动云计算任务调度模型

2.1 问题描述和基本框架

由于移动云计算技术可以帮助移动用户将一些工作流计算任务迁移至云端服务器执行,从而节省移动设备的能耗,并提高计算效率.目前研究者已经开始探讨如何将移动云计算技术应用到日常工作环境中,例如物体/手势识别、图像/视频编辑、自然语言处理等领域[2,17-18].本文现通过一个具体实例来阐释相关问题.某大型企业为了提高管理效率、降低运营成本,建立了一个移动工作流管理系统,企业员工可利用移动终端设备进入系统中就可以随时随地地办公.在工作过程中,由于企业的一些数据处理工作较为复杂,此外一些复杂任务的计算量也较大,员工所携带的移动设备硬件能力与电池能耗已经不能满足移动办公的需要.为解决该问题,企业可搭建一个基于移动云计算环境的移动办公系统.下面我们通过一个能耗和任务执行时间都比较大的图片识别应用来进行说明,如图 1 所示.该应用包括图片数字化、色彩提取、形状提取、边缘提取、纹理提取、空间提取等一些处理步骤.对于图片采集和数字化工作可由移动设备自身完成;而对于色彩提取、形状提取、纹理提取等工作,由于计算量较大,则需要上传到公司的云端服务器去完成.对于这个工作流应用,

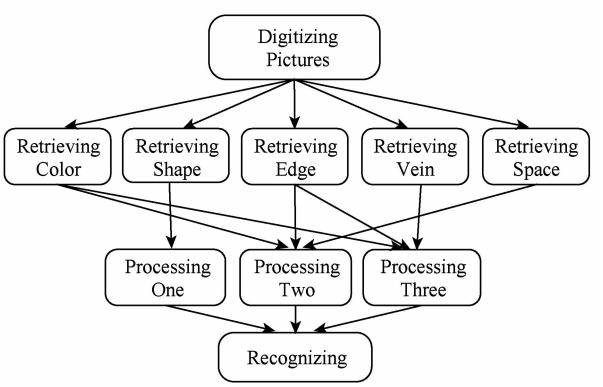

Fig. 1 Application of picture recognition

图 1 图片识别应用程序

我们可以用一个有向无环工作流图(如图 2 所示)$G=(V,E)$来表示,其中每个节点 $v_i \in V$ 表示一个任务,有向边 $e(v_i,v_j) \in E$ 表示 2 任务之间的先后关系,即只有等 v_i 完成之后 v_j 才可以开始执行.

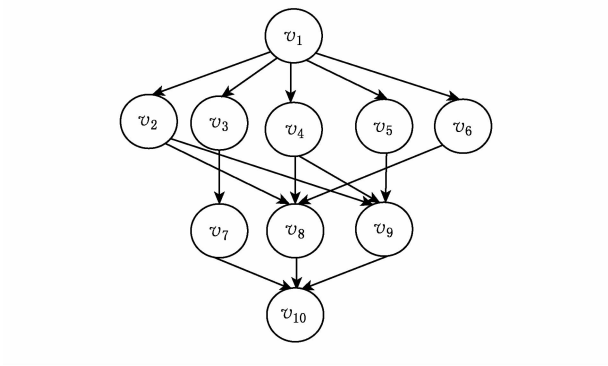

Fig. 2 Directed acyclic graph of the workflow

图 2 工作流的有向无环图

一个工作流由 N 个任务节点组成,一般情况下,N 不会超过 100,比如在计算机视觉应用程序中,N 的范围就是 10～30.在给定的工作流任务图中,没有任何前驱节点的任务节点为开始任务,没有任何后续节点的任务节点为结束任务.如图 2 所示,任务 v_1 是开始任务,任务 v_{10} 是结束任务;此外,其他一些工作流任务可以有前驱任务和后续任务(例如,对于任务 v_5 来说,v_1 是它的前驱,v_9 是它的后续).基于文献[18],我们给出图 2 中各个任务在移动设备处理器以及在云端服务器的处理时间如表 1 所示:

Table 1 Completed Time of Each Task

表 1 任务完成时间 s

Task	T_{core1}	T_{core2}	T_{core3}	$ts(v_i)$	$tc(v_i)$	$tr(v_i)$
v_1	9	7	5	3	1	1
v_2	8	6	5	3	1	1
v_3	6	5	4	3	1	1
v_4	7	5	3	3	1	1
v_5	5	4	2	3	1	1
v_6	7	6	4	3	1	1
v_7	8	5	3	3	1	1
v_8	6	4	2	3	1	1
v_9	5	3	2	3	1	1
v_{10}	7	4	2	3	1	1

表 1 表示 10 个任务分别在 3 个处理器上的完成时间(表示为 T_{core1},T_{core2},T_{core3})以及任务被迁移到云端服务器处理所需的发送时间 $ts(v_i)$、执行时间 $tc(v_i)$和接收时间 $tr(v_i)$,由于在云端可以高效地进行任务处理,任务在云端的处理时间统一为 3 s,

发送任务到云端和从云端接收运行结果的时间都统一为 1 s.

为了便于理解,我们总结了本文的主要符号及其含义如表 2 所示:

Table 2 Definitions of Notations

表 2 相关符号与定义

Notation	Definition
$V=\{v_i\}$	The set of tasks in the workflow
M	The number of kernels in the mobile equipment
N	The number of tasks in the workflow
$X(v_i,k)$	The indictor which shows whether a task is executed locally: if $k\leqslant M$, the task is executed locally on the k-th kernel; if $k=M+1$, the task is offloaded to the cloud for executing
α	The weight of time in the objective function.
$tc(v_i)$	The time needed for executing the task v_i in cloud servers
$ec(v_i)$	The energy consumed by mobile equipments in migrating the task v_i to cloud servers for executing
$tmk(v_i,k)$	The time needed for executing the task v_i on the k-th kernel in mobile device
$emk(v_i,k)$	The energy consumed for executing the task v_i on the k-th kernel in mobile device
$rat(v_i)$	The actual ratio of voltage changed for task v_i
$rat_{\min}(v_i)$	The minimal ratio of voltage that can be changed for task v_i

2.2 任务执行时间

设移动设备含有 M 个异构处理器,每个处理器都可进行动态电压频率调节[18],即在不同频率下对应有不同的电压.若用 $f_{\max}(k)$ 表示任务在第 k 个处理器上执行时的最大电压频率,则任务 v_i 运行时的实际电压 $f(v_i,k)$ 可由电压变化率 $rat(v_i)$ 控制,即 $f(v_i,k)=f_{\max}(k)\times rat(v_i)$,并且 $rat(v_i)\in[rat_{\min}(v_i),1]$,这里 $rat_{\min}(v_i)$ 由式(1)确定:

$$rat_{\min}(v_i)=\frac{1}{1+\dfrac{TL-TM}{tmk(v_i,k)}},\tag{1}$$

其中,TL 表示任务 v_i 给定的截止时间,TM 表示平均每个任务的理论最小完成时间:

$$TM=\frac{1}{N}\sum_{i=1}^{N}\min\{tc(v_i),\min_{1\leqslant k\leqslant M}\{tmk(v_i,k)\}\},\tag{2}$$

其中,$tc(v_i)$ 表示在云端服务器上运行任务 v_i 所需的执行时间,包括将 v_i 通过无线网络发送到云端服务器上所需时间、任务 v_i 在云端服务器执行所需时间以及将执行处理后的任务结果通过无线网络返回到移动设备上所需时间这 3 部分;$tmk(v_i,k)$ 为移动设备的第 k 个处理器上运行任务 v_i 所需的执行时间,若采用 DVFS 调节,则任务 v_i 在第 k 个处理器上的实际运行时间为 $tmk(v_i,k)/rat(v_i)$.综合考虑这 2 方面情况,则任务 v_i 的实际执行所需时间为

$$runT(v_i)=\sum_{k=1}^{M+1}(X(v_i,k)\times tmk(v_i,k)+\overline{X}(v_i,k)\times ct(v_i))/rat(v_i),\tag{3}$$

其中,$X(v_i,k)$ 为指示函数:当 $k\leqslant M$ 时,$X(v_i,k)=1$ 且 $\overline{X}(v_i,k)=0$,表示任务 v_i 在移动设备的第 k 个处理器上运行;当 $k=M+1$ 时,$X(v_i,k)=0$ 且 $\overline{X}(v_i,k)=1$,表示任务 v_i 在云端服务器上运行.此外,当 v_i 在云端服务器上执行时,其电压变化率被固定为 $rat(v_i)=1$.

2.3 能源消耗

在任务调度优化过程中,一方面需要缩短工作流任务的总完成时间,另一方面也需要考虑工作流任务执行过程中移动设备的能源消耗.

设 $emk(v_i,k)$ 为无 DVFS 调节时任务 v_i 在第 k 个处理器上执行的能耗;若 v_i 执行时,处理器上的电压变化率为 $rat(v_i)$,则 v_i 执行时的实际能源消耗为

$$E(v_i,k)=rat(v_i)\times emk(v_i,k).\tag{4}$$

则整个工作流 $G=(V,E)$ 执行时移动设备所需的总能源消耗可以表示为

$$E_{\text{Total}}=\sum_{i=1}^{N}\sum_{k=1}^{M+1}(X(v_i,k)\times E(v_i,k)+\overline{X}(v_i,k)\times ec(v_i)),\tag{5}$$

其中,$X(v_i,k)=1$ 时表示任务 v_i 在移动设备里第 k 个处理器上执行(此时 $\overline{X}(v_i,k)=0$);$ec(v_i)$ 表示任务 v_i 被迁移到云端服务器上执行时移动设备端的能耗.显然,如果任务在云端服务器上执行,那么对移动设备而言,其能源消耗只包括将任务发送至云端服务器以及从云端服务器接收任务执行结果这 2 部分.

3 算法设计

本文提出一种基于模拟退火算法的任务调度策略 SA-DVSA 算法,来解决工作流任务调度的能耗与执行时间优化问题,算法分为 3 个步骤:初始化可行解、改变参数可行解和新变量取舍的判断.我们将会分析加入了连续动态电压调节技术后对结果的影响,并在实验部分对 SA-DVSA 算法与其他相关工作进行比较.

3.1 算法框架

基于物理中固体物质的退火过程与一般优化问题之间具有一定的相似性,模拟退火算法[19-20]已被证明是一种适合求解大规模组合优化问题的通用且有效的近似算法.对于组合优化问题来说,解空间的每一点都代表一个具有不同目标函数值的解,而优化过程就是在解空间寻找函数最小解的过程.若把函数看成能量函数,把控制参数视为温度,解空间作为状态空间,那么模拟退火算法寻找基态的过程就可被是为求解目标函数极小值的优化过程,算法流程图[19]如图 3 所示.

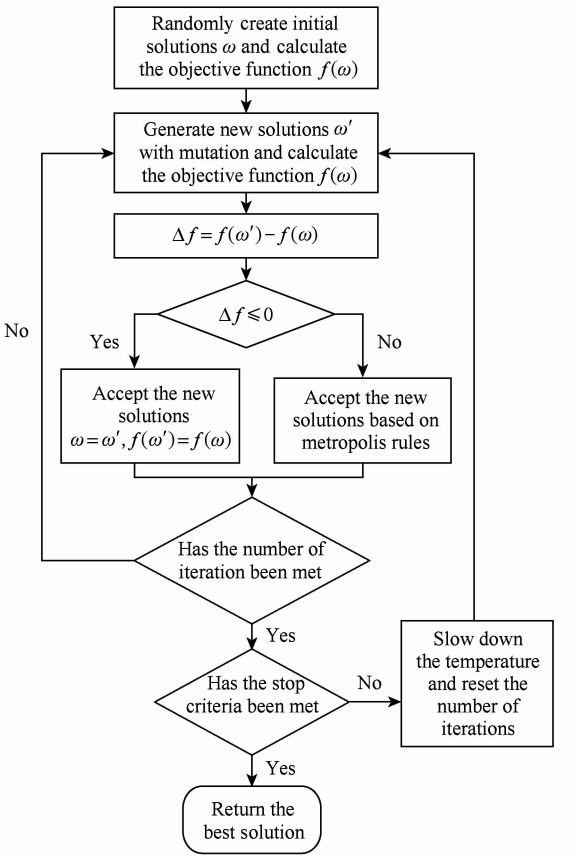

Fig. 3 Framework of the simulated annealing algorithm

图 3 模拟退火算法框架图

将任务的处理位置、执行顺序、处理器电压变化率作为变量,而工作流任务执行的总完成时间与移动设备的能源消耗这 2 方面的加权值作为适应度目标函数,则可将本文中的移动云计算环境中工作流任务调度优化问题转化成组合优化问题.对于这类组合优化问题,我们可以根据适应度函数,通过模拟退火算法迭代得到变量的最优值.下面给出变量的定义:1)$LOC=\{loc(v_i)\}$表示各工作流任务执行位置的集合,其中 $loc(v_i)=k(1\leqslant k\leqslant M+1)$表示任务 v_i 被执行的位置:即若 $k<M+1$,表示任务 v_i 在手机第 k 个处理器上执行;若 $k=M+1$,则表示任务 v_i 在云端服务器上执行.2)$RATIO=\{rat(v_i)\}$表示各工作流任务执行时所使用的实际电压变化率的集合,其中 $rat(v_i)$表示任务 v_i 被处理时所使用的实际电压变化率;此外,当任务在云端执行时,其电压变化率被固定为 1.0.3)$ORDER=\{ord(v_i)\}$表示各工作流任务被执行的次序集合,其中 $ord(v_i)=j$ 表示任务 v_i 的执行次序是第 j 个.显然,若这 3 个集合中所有变量的值已固定,则这个工作流任务调度的策略也将固定.

设任务 v_i 的开始执行时刻为 $startT(v_i)$,则由式(3)可得其实际完成时间为

$$AFT(v_i)=startT(v_i)+runT(v_i). \tag{6}$$

v_i 的开始执行时刻 $startT(v_i)$由 2 方面因素决定:1)其前驱任务节点的最迟完成时刻;2)其所在移动设备处理器或云端服务器上排序在前的那些任务的最迟完成时刻.这样,$startT(v_i)$可递归地用 $AFT(v_j)(v_j\neq v_i)$表示为

$$startT(v_i)=\max\{\max_{v_j\in pre(v_i)}\{AFT(v_j)\},\max_{\substack{v_j\neq v_i\wedge\\ ord(v_j)<ord(v_i)\wedge\\ loc(v_j)=loc(v_i)}}\{AFT(v_j)\}\}, \tag{7}$$

其中,$\max_{v_j\in pre(v_i)}\{AFT(v_j)\}$表示的是 v_i 所有前驱任务的实际完成时间的最大值(其中 $pre(v_i)$为 v_i 的前序任务节点集),$\max_{\substack{v_j\neq v_i\wedge\\ ord(v_j)<ord(v_i)\wedge\\ loc(v_j)=loc(v_i)}}\{AFT(v_j)\}$表示那些在 v_i 之前便被分配到相同处理位置(同一个移动设备处理器或云端服务器)的其他任务的实际最迟完成时刻.

这样整个工作流的总完成时间可以表示为

$$T_{\text{Total}}=\max_{v_i\in V}\{AFT(v_i)\}, \tag{8}$$

则根据式(5)和式(8),执行时间与能耗的联合优化目标函数可表示为

$$F=\alpha\times T_{\text{Total}}+(1-\alpha)\times E_{\text{Total}}, \quad (9)$$

其中 $\alpha\in[0,1]$表示完成时间的权重,则 $1-\alpha$ 为能耗的权重.

3.2 初始化可行解

在本文的算法中,初始化解是随机生成的. INIT_VAR 算法先随机产生工作流任务被处理的位置集 $LOC=\{loc(v_i),loc(v_2),\cdots,loc(v_N)\}$,然后根据式(1)和式(2)计算每个任务的最小电压变化率,再在此基础上随机产生任务的电压变化率集 $RATIO=\{rat(v_i)\}$,接着初始化 N 个任务的顺序为 $ORDER=\{ord(v_i)\mid ord(v_i)=i,1\leqslant i\leqslant N\}$,最后计算总能耗 E_{Total}、运行时间 T_{Total} 以及目标函数值 F. 下面给出 INIT_VAR 算法的执行过程:

算法 1. INIT_VAR 算法.

输入:任务数 N、移动设备的处理器数 M、任务在云端服务器运行时间集 $TC=\{tc(v_i)\}$、任务迁移到云端服务器运行时移动设备能耗集 $EC=\{ec(v_i)\}$、任务在移动设备的运行时间集 $TMK=\{tmk(v_i,k)\}$;任务在移动设备运行的能耗集 $EMK=\{emk(v_i,k)\}$、截止时间 TL;

输出:任务处理初始位置集 LOC、初始电压变化率集 $RATIO$、初始任务执行顺序集 $ORDER$.

随机生成处理位置集 $LOC:=\{loc(v_i)\}$,其中 $loc(v_i)=random(1,M+1)$;
FOR each $v_i\in V$ DO
  根据式(1)、式(2)计算 $rat_{\min}(v_i)$;
  IF($loc(v_i)\neq M+1$)
    $rat(v_i):=random(rat_{\min}(v_i),1)$;
  ELSE
    $rat(v_i):=1.0$;
  ENDIF
  将 $rat(v_i)$加入集合 $RATIO$ 中;
ENDFOR
生成初始任务执行顺序集 $ORDER:=\{ord(v_i)\}$,其中 $ord(v_i)=i$;
根据式(5)、式(8)和式(9)计算目标函数值 F.

3.3 改变参数可行解算法

CHANGE_VAR 算法将随机改变任务 v_i 执行所在的位置及电压. 在改变执行位置的过程中,随机选择某处理器或云端服务器位置 k,得到 k 处的任务 v_i 可被执行的次序区间 R_1. R_1 可通过如下过程计算得到:找出 v_i 在 k 处的全部前驱任务中的最大执行次序值 x_1,找到 v_i 在 k 处的全部后续任务中的最小执行次序值 x_2,则 R_1 区间为 (x_1,x_2). 在区间 R_1 上随机选择 v_i 的某个执行次列位 s_1;然后遍历 R_1 中所有其他执行次列上的当前任务,对每个任务 v_j,分析其在 k 处上的可执行的次序区间 R_2,分析 R_2 与 R_1 的交集 R_{cross},若 $R_{\text{cross}}\neq\varnothing$,则表示 v_i 与 v_j 可互选,则将 v_j 的执行次序位加入集合 R_A 中,最终从可互换的执行序列集 R_A 中随机选择一个序列位 s_2,交换 s_1 和 s_2 上的执行任务,从而得到新的可行解$\langle LOC',RATIO',ORDER'\rangle$和目标函数值 F'. 下面给出 CHANGE_VAR 算法的执行过程:

算法 2. CHANGE_VAR 算法.

输入:任务处理位置集 LOC、电压变化率集 $RATIO$、任务执行顺序集 $ORDER$;

输出:更新后的任务处理位置集 LOC'、电压变化率 $RATIO'$、新任务顺序 $ORDER'$.

$LOC':=LOC$; $RATIO':=RATIO$; $ORDER':=ORDER$;
从任务集 V 中随机选取任务 v_i;
随机选取任务新的处理位置 k, $loc(v_i):=k$;
IF($k\neq M+1$)
  $rat(v_i):=random(rat_{\min}(v_i),1)$;
ELSE
  $rat(v_i):=1.0$;
ENDIF
计算 v_i 处理位置 k 上的可执行次序区间 R_1;
在 R_1 中随机选择可执行序列位 s_1, $ord(v_i):=s_1$;
FOR each $s_2\in R_1$ DO
  IF($s_2\neq s_1$)
    获取 s_2 上所对应的执行任务 v_j;
    计算 v_j 在处理位置 k 上的可执行次序区间 R_2;
    令 $R_{\text{cross}}:=R_1\cap R_2$;
  ENDIF
  IF($R_{\text{cross}}\neq\varnothing$)
    $R_A:=R_A\cup\{s_2\}$;
  ENDIF
ENDFOR
IF($R_A\neq\varnothing$)
  从 R_A 中随机选取序列位 s_2,获取 s_2 上所对应的执行任务 v_j;
  交换 $ORDER$ 中 v_i 与 v_j 的执行序列;
ENDIF

3.4 变量值取舍算法

通过改变参数可行解算法得到了新可行解,新可行解是否被接受为当前最优解需要进一步计算确定.

在 ACC_NEW_VAR 算法中,设新可行解所对应的目标函数值为 F',若 $F'<F$,则用新的可行解作为当前解;若 $F'\geqslant F$,则判断 $e^{(F-F')/temp_k}>rand()$:若成立,则采用新的可行解 F',否则保留当前解 F,这里 $temp_k$ 表示当前迭代第 k 次时的退火温度.下面给出 ACC_NEW_VAR 算法的执行过程:

算法 3. ACC_NEW_VAR 算法.

输入:任务处理位置集 LOC、电压变化率集 $RATIO$、任务执行顺序集 $ORDER$、更新后的任务处理位置集 LOC'、电压比例 $RATIO'$、新任务顺序 $ORDER'$;

输出:True 或 False.

令 $LOC,RATIO,ORDER$ 所对应的当前目标函数值为 F;

根据 $LOC',RATIO',ORDER'$ 计算新的目标函数值 F';

$\Delta F:=F'-F$;

IF($\Delta F>0$)

IF($e^{(F-F')/temp_k}\leqslant rand()$)

RETURN False;

ENDIF

ENDIF

RETURN True.

3.5 算法流程

SA-DVSA 算法是基于迭代求解策略的一种随机寻优算法,它从一个较高的初始温度开始,在旧解基础上随机改变某个任务执行的位置和电压,并在满足任务执行关系的情况下,随机交换 2 个任务的顺序,伴随温度的不断下降重复抽样过程,直至得到问题的优化解,算法主要分以下 4 步:1)计算得到每个任务在各个移动设备处理器上电压变化率的取值范围;2)初始化可行解,计算得到适应度函数值 F,并作为当前最优值;3)根据当前变量值随机得到新的变量值,计算新变量值对应的适应度函数值 F',判断是否接受;4)在同一温度下求解并取舍新变量若干次,之后进行退温操作 $temp_{i+1}=\lambda\times temp_i$($\lambda$ 为温度下降率),直至温度小于给定值阈值 β.基于模拟退火算法的工作流任务调度 SA-DVSA 算法执行过程如下:

算法 4. SA-DVSA 算法.

输入:工作流图 $G=\langle V,E\rangle$、任务数 N、移动设备的处理器数 M、任务在云端运行时间集 $TC=\{tc(v_i)\}$、任务在云端运行能耗集 $EC=\{ec(v_i)\}$、任务在移动设备的运行时间集 $TMK=\{tmk(v_i,k)\}$;任务在移动设备运行的能耗集 $EMK=\{emk(v_i,k)\}$、截止时间 TL;

输出:最优调度策略$\langle LOC^*,RATIO^*,ORDER^*\rangle$.

调用算法 INIT_VAR(N,M,TC,EC,TMK,EMK,TL);

$temp:=num\times N\times M$; /* 初始化温度,$num$ 为同一温度下求解并取舍新变量的次数 */

$F^*:=+\infty,LOC^*:=\varnothing,ORD^*:=\varnothing,RATIO^*:=\varnothing$;

WHILE $temp>\beta$ do

FOR $i=1$ to num DO

调用算法 CHANGE_VAR($LOC,RATIO,ORDER$),获取新解$\langle LOC',RATIO',ORDER'\rangle$;

IF (ACC_NEW_VAR($LOC,RATIO,ORDER,LOC',RATIO',ORDER'$)= True)

$LOC=LOC',RATIO=RATIO',ORDER=ORDER'$;

根据式(5)计算得到总能耗 E_{Total};

根据式(8)计算得到任务总完成时间 T_{Total};

根据式(9)计算目标函数值 F;

ENDIF

IF($F<F^*$)

$F^*:=F,LOC^*:=LOC,RATIO^*:=RATIO,ORDER^*:=ORDER$;

ENDIF

ENDFOR

$temp:=temp\times\lambda$;

ENDWHILE

$RETURN\ \langle LOC^*,RATIO^*,ORDER^*\rangle$.

现在分析一下该算法的时间复杂度.SA-DVSA 算法外循环的迭代次数为 $\log_\lambda(\beta/(num\times N\times M))$(其中 λ 代表温度下降率,β 表示最终温度,N 表示任务总数,M 表示移动设备的处理器总数),时间复杂度约为 $O(\log_\lambda(M\times N))$且 $M<N$.外层 FOR 循环迭代次数为 num,而循环内的时间复杂度主要取决于如下 2 方面:1)CHANGE_VAR 算法.该算法的时间复杂度主要取决于遍历任务 v_i 在处理位置 k 上的可执行次序区间内每个序列位,由于 R_1 的长度小于任务数 N,而在寻找 s_1 的可调整队列时,实际上是从其对应任务在 k 处的全部前驱任务中的最大执行次序值开始搜索,直至其在 k 处的全部后续

任务中的最小执行次序值,包括后面的 $R_1 \cap R_2$ 操作,实际时间执行次数要小于 N,因此 CHANGE_VAR 算法的时间复杂度为 $O(N^2)$. 2)计算 E_{Total} 与 T_{Total}. 当调度策略已知时,在计算总能耗的过程中,我们只需要按照任务优先级,计算每个任务的能源消耗,然后相加即可,因此计算 E_{Total} 的时间复杂度为 $O(N)$;对于总完成时间 T_{Total} 的计算,我们用一个数组保存当前每个处理器的完成时间,而每个任务的开始时间等于其前驱任务最晚完成时间与处理器当前完成时间的较大值,需要遍历每个任务的前驱任务,因此时间复杂度主要由工作流图中边的个数决定,所以时间复杂度为 $O(N^2)$. 基于以上分析,SA-DVSA 算法的时间复杂度为 $O(N^2)$.

4 实 验

本文实验主要是基于 2.1 节中的工作流图示例(如图 2 所示)和随机产生的工作流图来与文献[18]中已有的 DVFS 算法,及没有采用 DVFS 技术的 MCC-SA 算法进行对比实验来得出结论,进而分析和评价本文所提出的 SA-DVSA 算法的效率. 实验采用的仿真工具为 MATLAB R2014a,计算环境为:Intel® Core™ i7-4790 CPU @ 3.60 GHz,内存 8 GB,运行在 64 位 Windows7 系统中.

由于文献[18]中的动态电压调节技术用到的是固定参数值,这样不利于寻找一个最优的参数从而使得可行解空间利用率达到最大,所以我们设计 SA-DVSA 算法时从目标函数中迭代寻找较优参数,这样以满足能耗与时间的多目标优化问题. 在适应度函数 $F=\alpha\times T_{\text{Total}}+(1-\alpha)\times E_{\text{Total}}$ 中,α 为权重因子,α 取值范围为 0～1,为了得到 α 的有效取值,在实验时,我们以内核数量为 3、随机生成的 10 个任务关系图为实验对象,得出 α 在不同取值时所对应的能耗和完成时间,结果如图 4 所示.

从图 4 可以看出,当 $\alpha=0$ 时,即只考虑优化时间,不考虑能耗问题,这时任务的完成时间最短,能耗最大;当 $\alpha=0.3$ 时,能耗明显减少,并且之后基本趋于稳定,此时任务完成时间相对于 $\alpha=0$ 时有所增加,但增值不是很多,α 在 0.3～0.7 这段值之间,能耗和时间都基本趋于稳定;当 $\alpha=1$ 时,此时能源消耗最少,但任务完成时间剧增. 因此,α 的有效取值范围在 0.3～0.7 之间,最优取值为 0.35,在后面的实验中 α 的取值都为 0.35.

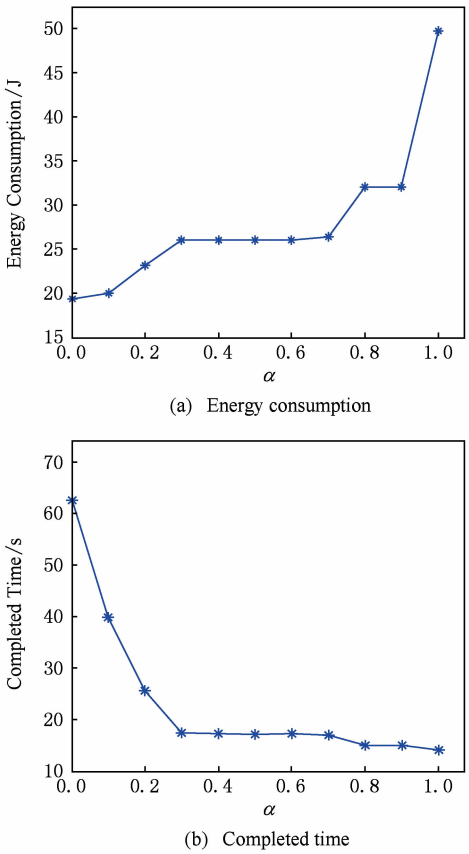

Fig. 4 Values of weight α

图 4 权重因子 α 的实验图

使用 SA-DVSA 算法进行任务调度实验时,算法过程中的迭代次数性能图如图 5 所示.

从图 5 中可以看出,在最开始迭代的时候,时间和能耗的波动都比较大,但整体的趋势是降低的,当次数迭代到 800 左右时 2 个值都逐渐趋于平稳,当次数达到 1 100 时 2 个值都达到了最优解.

下面基于 2.1 节中所给的工作流图来测试 3 种算法的调度结果,整个应用程序的总完成时间不能超过限制时间 30 s(即 $T_L \leqslant 32$ s). 移动设备有 3 个处理器,每个处理器在最大电压频率下能耗分别为 $P_1=1$J, $P_2=2$J, $P_3=4$J. DVFS 算法、MCC-SA 算法和 SA-DVSA 算法调度结果分别如图 6～8 所示.

图 6 是用 DVFS 算法调度分配的结果,总完成时间 $T_{\text{Total}}=26$ s,总能耗 $E_{\text{Total}}=23$J. 图 7 是采用没有加入连续动态电压技术的 MCC-SA 算法调度分配的结果,此时 $T_{\text{Total}}=26$ s, $E_{\text{Total}}=22$J,该算法虽然在执行时间上与 DVFS 算法的结果一样,但是所需能耗有所减少. SA-DVSA 算法调度结果如图 8 所示, $T_{\text{Total}}=23.03$ s, $E_{\text{Total}}=19.47$J,对比 DVFS 算法、MCC-SA 算法,其在总完成时间和总能耗方面的效果都更好.

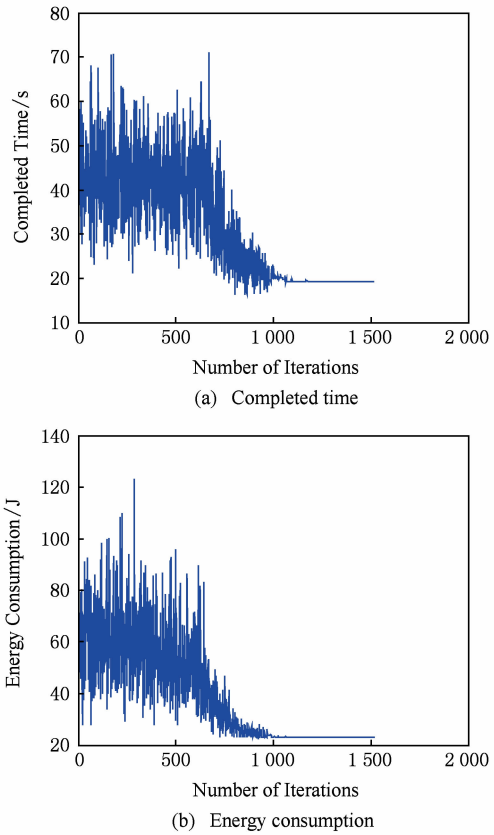

Fig. 5 Number of iterations

图 5 实验迭代次数图

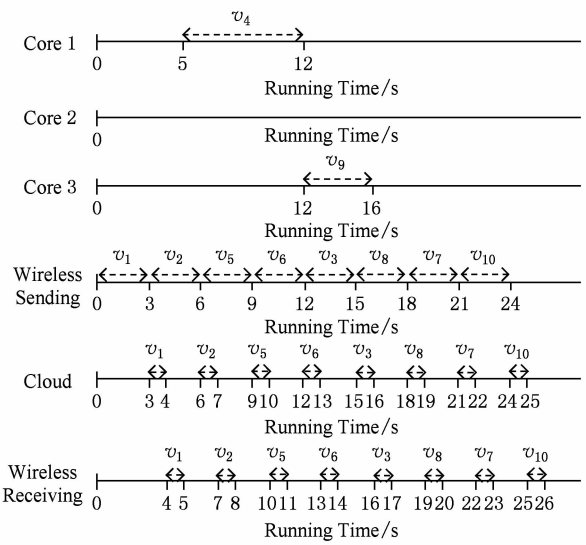

Fig. 6 Scheduling under DVFS algorithm

图 6 DVFS 算法调度分配结果

从图 7 可以看出,在 MCC-SA 算法中,任务 8～9 之间存在 1 s 的时间空闲,为了更好地利用这部分时间,在满足各个任务的开始和完成时间范围内,我们可以通过降低电压增加任务处理时间来降低能耗:例如在图 8 中,任务 3 在处理器 1 上用最大电压

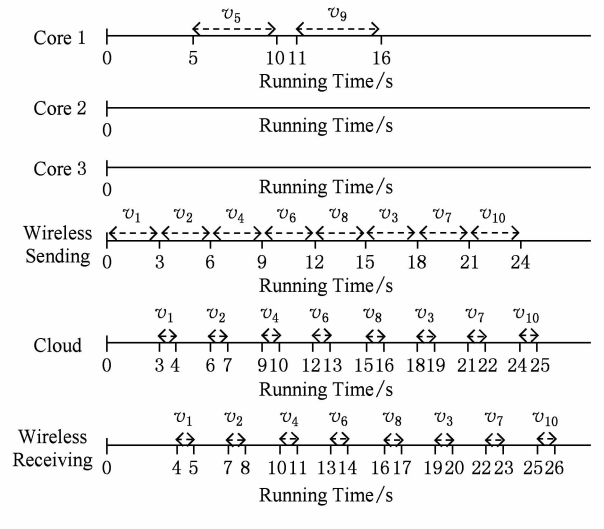

Fig. 7 Scheduling under MCC-SA algorithm

图 7 MCC-SA 算法调度分配结果

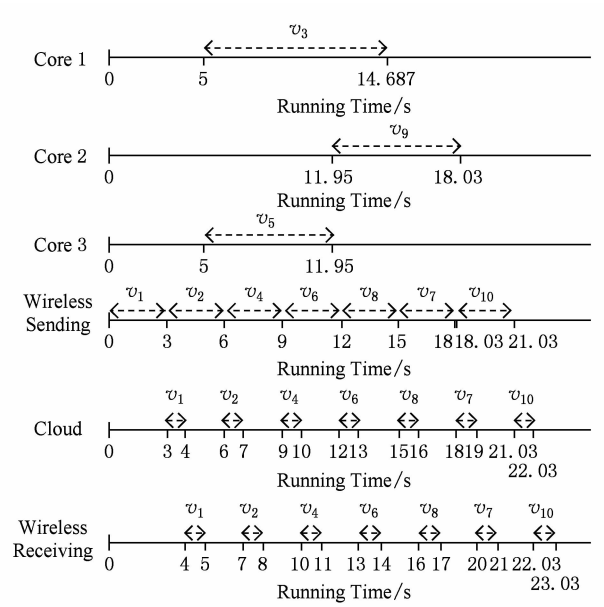

Fig. 8 Scheduling under SA-DVSA algorithm

图 8 SA-DVSA 算法调度分配结果

的完成时间是 6 s,即在时间为 11 s 的时候任务 3 结束;任务 3 完成之后执行任务 7,但是任务 7 的开始时间是 15 s,11～15 s 之间存在 4 s 时间段的空隙,这时降低处理器电压频率,使得任务 3 的完成时间延长为 14.687 s,这样既可以降低处理器的能耗,也可以得到任务 3 完成时间不超过 15 s 的调度策略,显然这种任务调度策略是更加优化的.

综上所述,可以看出本文提出的 SA-DVSA 算法无论是在完成时间上还是在能源消耗上都比 DVFS 算法及 MCC-SA 算法都要更好.

为了进一步证明本文算法的有效性,下面将通过随机产生的具有 50～100 个不同任务节点的工作

流图来进行对比实验(移动设备的处理器数 $M=3$).3 种算法各自任务完成时间和能源消耗情况如图 9 所示:

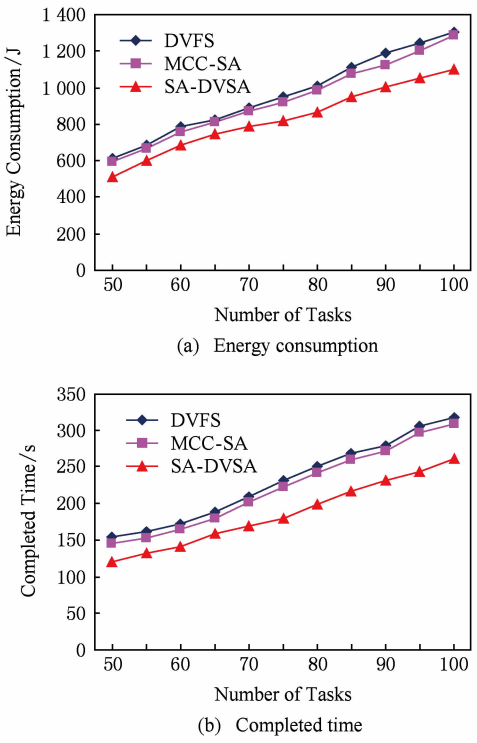

Fig. 9 Energy consumption and completed time with number of tasks changing where $M=3$

图 9 任务数与时间能耗关系图($M=3$)

图 9(a)显示当任务个数变化时 3 种算法下各自的移动设备能耗,图 9(b)表示 3 种算法在任务个数不同时与任务完成时间的关系. 从图 9 中可以观察到,SA-DVSA 算法无论是在任务完成时间上还是能耗上都比 DVFS 算法和 MCC-SA 算法要好. 随着任务节点个数的增加,任务完成时间和能耗都随之增加,但是节点个数增加到一定数量之后,SA-DVSA 算法的性能明显优于 DVFS 算法和 MCC-SA 算法,所以,SA-DVSA 算法在大规模的任务节点数中有更好的优势.

在图 10 的实验中,我们将移动设备的处理器数设为 $M=6$,每个处理器在最大电压频率下能耗分别为 $P_1=1$J,$P_2=2$J,$P_3=4$J,$P_4=8$J,$P_5=16$J,$P_6=32$J. 从图 10 中可以得出和图 9 中相似的结论. 此外,从图 9 和图 10 中可以观察到,当移动设备的处理器数 M 增加时,任务完成时间相应减少了,但是能耗却增加了,这是因为当移动设备有了更强计算能力的处理器,许多任务将被调度到上面执行,这样工作流任务执行所需的总能耗也将随之增多.

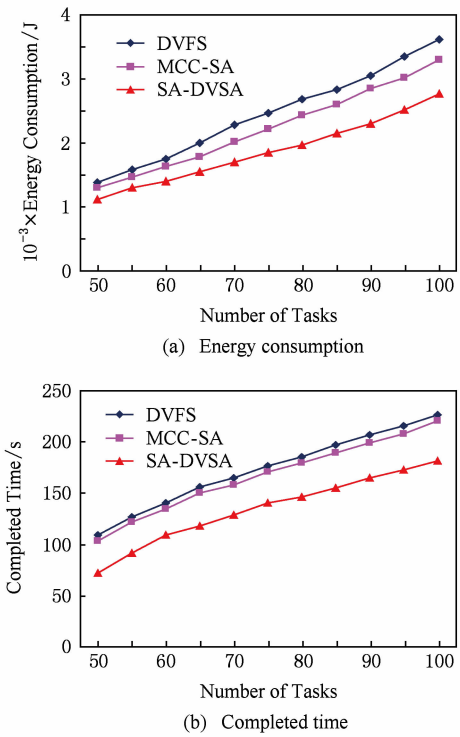

Fig. 10 Energy consumption and completed time with number of tasks changing where $M=6$

图 10 任务数与时间能耗关系图($M=6$)

5 结 论

任务调度是工作流系统管理中的重要研究内容,在移动云计算环境中进行任务调度时,一个重要问题就是如何制定调度策略使任务完成时间和能耗达到最小. 以往的研究中主要都集中在如何降低能耗的问题上,对能耗和时间的联合优化研究得很少. 本文所提的 SA-DVSA 算法对这 2 个目标进行联合优化,并通过实验验证了该方法比文献中的 DVFS 算法在任务完成时间以及能耗问题都有了进步. 在今后的工作中,本文将尝试设计其他启发式算法与模拟退火算法相结合来解决移动云计算的任务调度问题,并综合考虑能耗、时间、安全等多方面因素[21],这些有待于进一步的努力.

参 考 文 献

[1] Chen Shuang, Wang Yanzhi, Pedram M. A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud [C] //Proc of the 2013 Int Conf on Global Communications. Piscataway, NJ: IEEE, 2013: 2885-2890

[2] Deng Shuiguang, Huang Longtao, Taheri J, et al. Computation offloading for service workflow in mobile cloud computing [J]. IEEE Trans on Parallel and Distributed Systems, 2015, 26(2): 3317-3329

[3] Khan A, Ahirwar K. Mobile cloud computing as a future of mobile multimedia database [J]. International Journal of Computer Science and Communication, 2011, 2(1): 219-221

[4] Miettinen A P, Nurminen J K. Energy efficiency of mobile clients in cloud computing [C] //Proc of the USENIX Int Conf on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2010: 14-21

[5] Mavroeidakos T, Michalas A, Vergados D. Security architecture based on defense in depth for cloud computing environment [C] //Proc of the 2016 IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2016: 334-339

[6] Sarbazi H, Zomaya A. Large Scale Network-Centric Distributed Systems [M]. Piscataway, NJ: IEEE, 2014

[7] Singh B, Dhawan S, Arora A, et al. A view of cloud computing [J]. Communications of the ACM, 2013, 53(4): 50-58

[8] Patra B, Roy S, Chowdhury C. A framework for energy efficient and flexible offloading scheme for handheld devices [C] //Proc of the IEEE Int Conf on Advanced Networks and Telecommunications Systems. Piscataway, NJ: IEEE, 2015: 1-6

[9] Kumar K, Liu Jibang, Lu Yunhang, et al. A survey of computation offloading for mobile systems [J]. Mobile Networks and Applications, 2013, 18(1): 129-140

[10] Balamurugan M, Akila V. Effective processor selection on heterogeneous computing [C] //Proc of the 2016 Int Conf on Science Technology Engineering and Management. Piscataway, NJ: IEEE, 2016: 13-16

[11] Feng Bailong, Gao Jing. Distributed parallel Needleman-Wunsch algorithm on heterogeneous cluster system [C] // Proc of the 2015 Int Conf on Network and Information Systems for Computers. Piscataway, NJ: IEEE, 2015: 12-18

[12] Ra M, Sheth A, Mummert L, et al. Odessa: Enabling interactive perception applications on mobile devices [C] // Proc of the 9th Int Conf on Mobile Systems, Applications, and Services. New York: ACM, 2011: 43-56

[13] Yang Lei, Cao Jiannong, Yuan Yin, et al. A framework for partitioning and execution of data stream applications in mobile cloud computing [C] //Proc of the 5th Int Conf on Cloud Computing. Piscataway, NJ: IEEE, 2012: 794-802

[14] Terzopoulos G, Karatza H. Dynamic voltage scaling scheduling on power-aware clusters under power constraints [C] //Proc of the 17th Int Conf on Distributed Simulation and Real Time Applications. New York: ACM, 2013: 72-78

[15] Saravanan S, Venkatachalam V. Advance MapReduce task scheduling algorithm using mobile cloud multimedia services architecture [C] //Proc of the 6th Int Conf on Advanced Computing. Piscataway, NJ: IEEE, 2014: 21-25

[16] Deshmukh N, Deorankar A. Minimizing energy consumption in transmission efficient wireless sensor network [C] //Proc of the Int Conf on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics. Piscataway, NJ: IEEE, 2016: 475-479

[17] Kumar K, Lu Yunhang. Cloud computing for mobile users can offloading computation save energy [J]. Computer, 2010, 43(4): 51-56

[18] Lin Xue, Wang Yanzhi, Xie Qing, et al. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment [J]. IEEE Trans on Services Computing, 2015, 8(2): 175-186

[19] Davis L. Genetic Algorithms and Simulated Annealing: An Overview [M]. San Francisco, CA: Morgan Kaufmann, 1987

[20] Fleming P, Pashkevich A, Fleming P, et al. Computer aided control system design using a multiobjective optimization approach [C] //Proc of the Int Conf on Control. Piscataway, NJ: IEEE, 1985: 17-26

[21] Han Rui, Liu Yingbo, Wen Lijie, et al. A probabilistic approach to analyze and adjust time constraints in workflow management system [J]. Journal of Computer Research and Development, 2010, 47(1): 157-163 (in Chinese)
(韩锐, 刘英博, 闻立杰, 等. 工作流管理系统中一种概率性分析和调整时间约束的方法[J]. 计算机研究与发展, 2010, 47(1): 157-163)

Hu Haiyang, born in 1977. PhD and professor. His main research interests include workflow system and parallel computing.

Liu Runhua, born in 1990. Master. Her main research interests include workflow system and business process management.

Hu Hua, born in 1964. Professor and PhD supervisor. His main research interests include workflow system and database theory.