QoS 约束云环境下的工作流能效调度算法

李廷元[1] 王博岩[2]
(中国民用航空飞行学院计算机学院 四川 广汉 618307)[1]
(中国民航大学计算机科学与技术学院 天津 300300)[2]

摘 要 云环境可以为大规模工作流的执行提供高效、可靠的运行环境,但工作流执行时带来的高能耗不仅会增加云资源提供方的经济成本,还会影响云系统的可靠性,并对环境产生不利影响。为了在满足用户截止时间 QoS 需求的同时降低云环境中工作流调度的执行能耗,提出一种工作流能效调度算法 QCWES。该算法将工作流的能效调度方案求解划分为 3 个阶段:截止时间重分配、任务调度选择排序以及基于 DVFS 的最佳资源选择。截止时间重分配阶段旨在将用户定义的全局工作流截止时间在各个任务间进行重分配,任务调度选择排序阶段旨在通过自顶向下的任务分级方式得到任务调度序列;基于 DVFS 的最佳资源选择阶段旨在为每个任务选择带有合适电压/频率等级的最优目标资源,在满足任务的子截止时间的前提下使总体能耗达到最小。通过随机工作流和基于高斯消元法的现实工作流结构,对算法的性能进行仿真实验分析。结果表明,所提算法可以在满足截止时间约束下降低工作流的执行能耗,实现用户方的 QoS 需求与资源方的能耗间的均衡。

关键词 云计算,工作流调度,QoS 约束,能效调度,动态电压/频率缩放
中图法分类号 TP393 **文献标识码** A

Workflow Energy-efficient Scheduling Algorithm in Cloud Environment with QoS Constraint

LI Ting-yuan[1] WANG Bo-yan[2]
(School of Computer, Civil Aviation Flight University of China, Guanghan, Sichuan 618307, China)[1]
(School of Computer Science and Technology, Civil Aviation University of Chian, Tianjin 300300, China)[2]

Abstract Cloud provides a high-efficient and reliable execution environment for scheduling large-scale workflow. However, the high energy consumption resulted by workflow execution not only increases the economic cost of cloud resource providers, but influences the system reliability and has a negative effect to the environment. For meeting user-defined deadline QoS requirement and reducing the execution consumption of workflow scheduling in cloud, a workflow energy-efficient scheduling algorithm QCWES was proposed. QCWES divides the energy-efficient scheduling scheme of workflow into three phases: the deadline redistribution, the ordering of scheduled tasks and the best resource selection based on DVFS. The deadline redistribution phase is to redistribute the user-defined overall workflow deadline among all tasks, the ordering of scheduled tasks is to obtain the scheduling order of tasks by top-down task leveling, the best resource selection based on DVFS is to select the best available resource with appropriate voltage/frequency level for each task so that the total energy consumption is minimal while meeting its sub-deadline. Some simulation experiments were constructed to evaluate the performance of our algorithm by random workflow and the real-world workflow based on Gaussian Elimination. The results show that QCWES can reduce the energy consumption of workflow scheduling under meeting deadline constraint, and achieve the trade-off between users' QoS requirement and resources' energy consumption.

Keywords Cloud computing, Workflow scheduling, QoS constraint, Enegy-efficient scheduling, Dynamic voltage/frequency scaling

近年来,云计算作为高性能的计算平台,越来越多地为大规模计算密集型工作流应用提供高性能的异构资源和运行环境[1]。存在顺序约束的工作流应用即为工作流模型,是科学与工程领域的典型应用模式。云环境中的工作流调度旨在实现不同资源与工作流任务之间的有效映射[2],并维持任务间的执行顺序约束。基于用户方与服务提供方的不同需求,工作流调度的目标也是不同的,如:最小化执行时间、执行代价或最大化系统可靠性等。然而,能耗问题是目前云计算环境面临的最大难题,尤其随着更多的大规模云数据中心的建立,云计算的能耗问题变得日益突出。

DVFS 是目前云资源提供方在 CPU 上应用的一种调整能耗的重要手段,它可以通过缩放 CPU 的电压/频率的方式来降低处理器的能耗[3]。然而,此时能耗的降低是以降低 CPU 性能为代价的,这可能导致工作流的执行时间增加,无

本文受国家民航局科技创新引导项目(MHRD20140214),民航局和国家自然基金委民航联合基金项目(U1333113)资助。
李廷元(1967-),男,硕士,副教授,主要研究方向为计算机网络、云计算及民航计算机应用;王博岩(1976-),男,博士,副教授,主要研究方向为信息安全、航空节能减排。

法满足用户的截止时间 QoS 约束。为了提高满足截止时间约束下的工作流调度能效，实现能效与性能的均衡，本文设计了一种工作流能效调度算法，该算法分为 3 个阶段：截止时间重分配、任务调度排序及基于 DVFS 的最佳资源选择。该算法能够在满足截止时间约束下降低工作流的执行能耗，实现用户方的 QoS 需求与资源方的能耗均衡。

1　相关研究

启发式算法是工作流调度的常用算法，最经典的算法为 HEFT[4]，该算法能够以较低的时间复杂度实现调度时间最小。其他线性启发式算法的一般优化目标则包括：执行时间 makespan[5]、资源利用率[6]、系统可靠性[7] 和总体执行代价[8]。这类算法单目标居多，且均忽略了能耗问题。

降低云环境中的调度带来的能耗不仅可以减少资源方的经济代价，而且能够提高系统可靠性。考虑能耗问题的相关研究中，文献[9]通过建立能耗与执行时间的相对优越函数，设计了能量感知调度算法，该算法通过考虑当前最佳资源及其相应电压缩放等级来计算优越函数值，并选择最大函数值资源作为调度目标；但该调度结果是局部最优的。文献[10]基于复制与能量平衡机制设计了一种算法 EDB，其副本机制有效提高了执行效率，且能耗开销也较小；其缺点在于处理器资源的电压/频率等级过高，总体能耗仍然过大。文献[11]提出能量感知算法，该算法不仅能够以关键路径求得最小的调度时间，而且可以在空闲时槽中对处理器电压进行调整，降低资源能耗；但算法在选择目标资源上能效不是最优的。国内相关的云工作流调度研究工作[12-15] 对于能耗优化问题考虑较少。

不同于目前的工作，本文为了提高满足截止时间约束下的工作流调度能效，设计了一种能效工作流调度算法，并将工作流的最优调度目标资源的选择过程划分为 3 个阶段，通过截止时间再分配、满足顺序依赖的任务调度排序以及考虑电压/频率等级的最优资源选择等过程，来实现截止时间 QoS 约束的云工作流能效调度。

2　模型描述

本节描述了云环境中的工作流调度模型，首先给出模型描述中所需的相关符号说明，如表 1 所列。

表 1　符号含义说明

符号	符号含义
$G=(T,E)$	工作流 DAG 模型
R	m 个云资源集合，$\{R=r_j \mid 1\leqslant j\leqslant m\}$
V_j	资源 r_j 的 $N(j)$ 个电压等级集合，$V_j=\{v_{j,k} \mid 1\leqslant k\leqslant N(j)\}$
F_j	资源 r_j 的 $N(j)$ 个频率等级集合，$V_j=\{f_{j,k} \mid 1\leqslant k\leqslant N(j)\}$
T	工作流 DAG 任务集合，$T=\{t_i \mid 1\leqslant i\leqslant n\}$
$e_{ij}=(t_i,r_j)$	任务 t_i 与 t_j 间的边 e_{ij}，即依赖关系
$w(t_i)$	任务 t_i 的大小
$w(e_{ij})$	边 e_{ij} 对应的传输数据大小
$ET(t_i,r_j)$	任务 t_i 在资源 r_j 上的执行时间
$MET(t_i)$	任务 t_i 的平均执行时间
$EST(t_i)$	任务 t_i 的最早开始时间
$LFT(t_i)$	任务 t_i 的最迟完成时间
$EET(t_i)$	任务 t_i 的期望完成时间
D_{user}	用户定义的工作流截止时间
E^{total}	资源的计算总能耗
EL^{total}	资源间的链路总能耗
$Sub\_d(t_i)$	任务 t_i 的子截止时间

2.1　资源模型

云资源表示为集合 $R=\{r_1,r_2,\cdots,r_m\}$，由 m 个具有 DVFS 能力的异构资源组成，且资源间的通信链路为全连通方式。资源拥有不同的计算和存储能力，可运行于不同电压等级（对应于不同频率）。通信链路拥有不同的带宽，任意两个资源间的带宽链路以 $m\times m$ 的数据传输时间矩阵表示，其元素 $B_{s,j}$ 表示资源 r_s 与 r_j 间传输单位数据量的时间。

DVFS 可以使得云资源运行于供电电压 V 和运行频率 F 的集合中。对于资源 $r_j\in R$，可用的离散供电电压等级可表示为集合 $V_j=\{v_{j,1},v_{j,2},\cdots,v_{j,N(j)}\}$，其中，$N(j)$ 表示资源 r_j 的供电电压和运行频率等级的数量，同时，$v_{j,\min}\leqslant v_{j,1}\leqslant v_{j,2}\leqslant\cdots\leqslant v_{j,N(j)}\leqslant v_{j,\max}$。同样地，资源处理器的运行频率集合可表示为 $F_j=\{f_{j,1},f_{j,2},\cdots,f_{j,N(j)}\}$，且 $f_{j,\min}\leqslant f_{j,1}\leqslant f_{j,2}\leqslant\cdots\leqslant f_{j,N(j)}\leqslant f_{j,\max}$。

2.2　工作流任务模型

包含依赖关系任务集的工作流以有向无循环图 DAG 表示为 $G=(T,E)$，其中，T 代表 n 个任务的节点集，任务 $t_i\in T,1\leqslant i\leqslant n$。$E$ 代表边集，边 $e_{ij}=(t_i,r_j)(e_{ij}\in E,1\leqslant i\leqslant n,1\leqslant j\leqslant m,i\neq j)$ 用于表示任务 t_i 与 t_j 间的依赖关系，此时，任务 t_i 为 t_j 的父任务，t_j 为 t_i 的子任务。同时，不存在父任务的任务称为入口任务，不存在子任务的任务称为出口任务。分配给任务 t_i 的权值 $w(t_i)$ 表示任务的大小，单位为 MI。分配给边 e_{ij} 的权值表示调度于不同资源上的任务 t_i 与 t_j 间的数据传输量。图 1 表示拥有 9 个任务的典型工作流结构图，边上的权值表示 9 个任务间的数据传输量或通信时间。

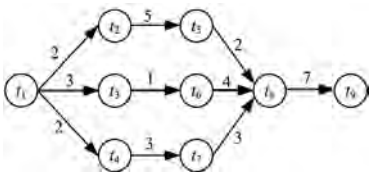


图 1　工作流模型

下面定义工作流任务的相关概念。

1）执行时间 $ET(t_i,r_j)$：任务 t_i 在资源 r_j 上的执行时间定义为：

$$ET(t_i,r_j)=\frac{w(t_i)\times CPI}{f_{j,k}} \tag{1}$$

其中，$f_{j,k}$ 表示资源（CPU）的运行频率，CPI 表示每条指令的 CPU 周期数。相应地，$ET^{\min}(t_i,r_j)$ 表示资源 r_j 以最大频率 $f_{j,\max}$ 执行任务 t_i 时的执行时间。

2）通信时间 $C_{p,i}^{s,j}$：表示资源 r_s 上的父任务 t_p 与资源 r_j 上的当前任务 t_i 间传输数据单元的通信时间，其定义为：

$$C_{p,i}^{s,j}=w(e_{p,i})\times B_{s,j} \tag{2}$$

3）最早开始时间 $EST(t_i)$：表示任务 t_i 的最早开始时间。由于云资源的异构特征，任务的执行时间在各资源上是不同的，因此，在未将任务调度至具体资源之前很难准确估计 EST。同样地，通信时间也取决于分配资源及其通信链路。为了计算 EST，考虑了平均执行时间 $MET(t_i)$ 和平均通信时间 $MC_{p,i}$。任务 t_i 的平均执行时间定义为：

$$MET(t_i)=\frac{1}{m}\sum_{j=1}^{m}ET(t_i,r_j) \tag{3}$$

即表示任务 t_i 在不同的可用资源上以最大频率执行时的平均时间。类似地，$MC_{p,i}$ 可以通过在可用资源间考虑单个数据

单元的平均传输时间 B 求得。基于 $MET(t_i)$ 和 $MC_{p,i}$，可得：

$$EST(t_i)=\begin{cases}0, & t_i=t_{\text{exit}}\\ \max\limits_{t_p\in pred(t_i)}\{EST(t_p)+MET(t_p)+MC_{p,i}\}, & t_i\neq t_{\text{exit}}\end{cases} \tag{4}$$

其中，$pred(t_i)$ 表示任务 t_i 的直接父任务集合。

4)最迟完成时间 $LFT(t_i)$：表示任务 t_i 的最迟完成时间。对于出口任务，其最迟完成时间等于用户定义的工作流截止时间 D_{user}。其他任务的 LFT 可定义为：

$$LFT(t_i)=\begin{cases}D_{\text{user}}, & t_i=t_{\text{exit}}\\ \min\limits_{t_c\in succ(t_i)}\{LFT(t_c)-MET(t_c)-MC_{i,c}\}, & t_i\neq t_{\text{exit}}\end{cases} \tag{5}$$

调度于资源 r_j 上的任务 t_i 的实际开始时间 $AST(t_i,r_j)$ 不同于 EST，它等于来自于父任务的最迟数据的到达时间与资源 r_j 的可用时间槽 $avail(r_j)$ 的较大值，且仅在任务调度至具体资源后才可进行计算：

$$AST(t_i,r_j)=\begin{cases}avail(r_j), & t_i=t_{\text{entry}}\\ \max\{\min\limits_{t_p\in pred(t_i)}\{AST(t_p,r_s)+ET(t_p,r_s)+C^{s,j}_{p,i}\}, & \\ avail(r_j)\}, & t_i\neq t_{\text{entry}}\end{cases} \tag{6}$$

2.3 基于 DVFS 的能耗模型

云环境中资源的主要功耗包括动态功能 P_{dynamic} 和静态功耗 P_{static}：

$$P_{\text{dynamic}}=A\times C\times v^2\times f=C_{eff}\times v^2\times f \tag{7}$$

其中，A 表示每个时钟周期的开关转换次数，C 表示总电容，C_{eff} 表示每个时钟周期的平均转换电容，$C_{eff}=A\times C$，v 表示 CPU 供电电压，f 表示 CPU 运行频率。CPU 的电压等级与频率等级的关系可表示为：

$$f=\frac{(V-V_{th})^{\alpha}}{K\times L_d} \tag{8}$$

其中，V_{th} 表示电压门限，α 和 K 表示与硬件相关的常量，L_d 表示逻辑深度，通常 $\alpha\in[1.2,2]$，实验中设置 $\alpha=1.5$。式(7)表明电压 v 是功耗的主要影响因素。

静态功耗 P_{static} 与 P_{dynamic} 成正比，其通常少于动态功耗的 30%，因此，资源的总体功耗由动态功耗决定。为了简化能耗模型，本文在实验中仅考虑资源动态能耗部分，表示为：

$$E=P_{\text{dynamic}}\times\Delta t=C_{eff}\times v^2\times f\times\Delta t \tag{9}$$

其中，Δt 表示任务的执行周期。

因此，在资源 r_j 上执行任务 t_i 的能耗 E_i 可定义为：

$$\begin{aligned}E_i&=C_{eff}\times v_{j,k}^2\times f_{j,k}\times ET(t_i,r_j)\\&=C_{eff}\times v_{j,k}^2\times w^*(t_i)\end{aligned} \tag{10}$$

其中，$v_{j,k}$ 表示资源 r_j 执行任务 t_i 时的运行频率，$w^*(t_i)$ 表示任务请求的 CPU 周期数。

给定工作流任务集 T、云资源集合 R 及调度解矩阵 X，所有任务的执行总能耗可定义为：

$$\begin{aligned}E^{\text{active}}&=\sum_{j=1}^{m}\sum_{i=1}^{n}x_{ij}\times E_i\\&=\sum_{j=1}^{m}\sum_{i=1}^{n}x_{ij}\times C_{eff}\times v_{j,k}^2\times f_{j,k}\times ET(t_i,r_j)\end{aligned} \tag{11}$$

其中，x_{ij} 为调度解矩阵 X 的元素，若任务 t_i 调度至资源 r_j，则 $x_{ij}=1$，否则 $x_{ij}=0$。

除任务执行能耗外，资源空闲时 CPU 也需要能量，此时 CPU 可运行于最低电压等级，空闲能耗可定义为：

$$\begin{aligned}E^{\text{idle}}&=\sum_{j=1}^{m}P^{\text{idle}}(\max_{i=1}^{n}\{FT(t_i,r_j)\}-\sum_{i=1}^{n}x_{ij}\times ET(t_i,r_j))\\&=\sum_{j=1}^{m}C_{eff}\times v_{j,\min}^2\times f_{j,\min}\times(\max_{i=1}^{n}\{FT(t_i,r_j)\}-\\&\quad\sum_{i=1}^{n}x_{ij}\times ET(t_i,r_j))\end{aligned} \tag{12}$$

其中，$\max\limits_{i=1}^{n}\{FT(t_i,r_j)\}$ 表示当前任务 t_i 在 r_j 上的最迟完成时间。

因此，计算资源的总体能耗为：

$$E^{\text{total}}=E^{\text{active}}+E^{\text{idle}} \tag{13}$$

父任务 t_p 与当前任务 t_i 间的数据传输链路能耗定义为资源 r_s 与 r_j 间链路的通信时间 $C^{s,j}_{p,i}$ 与链路功耗 PL_{sj}^{active} 之积：

$$EL_{sj}=C^{s,j}_{p,i}\times PL_{sj}^{\text{active}} \tag{14}$$

工作流结构中所有链路能耗为：

$$EL^{\text{active}}=\sum_{p=1}^{n}\sum_{i=1,p\neq i}^{n}\sum_{s=1}^{m}\sum_{j=1,s\neq j}^{m}x_{ps}\times x_{ij}\times C^{s,j}_{p,i}\times PL_{sj}^{\text{active}} \tag{15}$$

链路空闲时的能耗为链路功耗与链路空闲时间之积：

$$EL^{\text{idle}}=\sum_{s=1}^{m}\sum_{j=1,s\neq j}^{m}PL_{sj}^{\text{idle}}\times(\max_{i=1}^{n}\{FT(t_i,r_j)\}-\sum_{i=1}^{n}\sum_{p=1,p\neq i}^{n}(x_{ps}\times x_{ij}\times C^{s,j}_{p,i})) \tag{16}$$

因此，链路总能耗为：

$$EL^{\text{total}}=EL^{\text{active}}+EL^{\text{idle}} \tag{17}$$

综上，云资源方的总体能耗为：

$$E=E^{\text{total}}+EL^{\text{total}} \tag{18}$$

2.4 问题定义

云环境中的工作流调度即为分配包含 n 个任务的工作流至 m 个异构云资源集合上，在满足用户定义的截止时间约束的条件下使总体能耗达到最小，且不违背任务执行的顺序约束，可形式化为：

$$\begin{cases}\min E\\ \text{s.t. } makespan\leqslant D_{\text{user}}\end{cases} \tag{19}$$

3 QCWES 算法设计

本节设计了 QoS 约束下的工作流能效调度算法(QoS Constraint Workflow Energy-efficient Scheduling，QCWES)，该算法分为 3 个阶段：截止时间分配、任务调度选择排序和基于电压/频率等级的资源选择。

1)截止时间重分配：将用户定义的工作流截止时间重新分配给各个任务，使得每个任务能够在其子截止时间内完成，则工作流执行可以满足截止时间 QoS 约束。

2)任务调度选择排序：在满足任务间的依赖约束的前提下产生任务调度次序。

3)基于电压/频率等级的资源选择：为每个任务选择带有合适电压/频率的最优目标资源，在满足任务的子截止时间的前提下使总体能耗达到最小化。

3.1 截止时间重分配

算法 1 给出了截止时间的重分配过程。算法将分配节点作为输入节点，并分配子截止时间给其所有未分配关键父任务。为了便于说明，先定义决定路径和决定父任务的概念。

定义 1 任务 t_i 的决定路径 DP 为工作流入口任务与当前任务 t_i 间的最长路径。

定义 2　任务 t_i 的决定父任务为其未分配的父任务 t_p 中使 $EST(t_p)+MET(t_p)+MC_{p,i}$ 最大的父任务。

工作流的出口任务的决定路径表示所输入的工作流 DAG 的关键路径。首先，以正比于平均执行时间的方式，将全局截止时间分配至出口任务的决定路径的所有决定任务上。然后，为决定路径上的每个任务分配子截止时间，并进一步分配该子截止时间至所有未分配的决定父任务上。将决定路径上的节点作为输入节点，将其子截止时间作为其截止时间，重复以上过程，直到为所有任务分配子截止时间为止。

算法 1 开始于输入任务，并沿其决定父任务进行遍历，直到找到没有未分配父任务的任务为止，从而形成局部决定路径 P，如步骤 1－步骤 6 所示。步骤 7 调用算法 2，以路径 P 的次序将全局截止时间在每个任务间进行分配。步骤 8 则标记路径 P 上的任务为已分配。然后，任务的 EST 和 LFT 被更新，在路径 P 上的开始任务与结束任务间通过递归调用的方式重复以上过程，将全局截止时间在每个节点的未分配父任务间进行分配，如步骤 9－步骤 11 所示。

算法 1　截止时间重分配

```
procedure DeadlineDistribution(t)
1. while 任务 t 存在未分配父任务
2.   P←null,t_i←t
3.   While 任务 t_i 存在未分配父任务
4.     添加 t_i 的决定父任务至路径 P 首节点
5.     t_i←decisive_parent(t_i)
6.   end while
7.   调用算法 2,执行 AssignSubDeadline(P)
8.   标记路径 P 上的所有任务为已分配
9.   for 所有任务 t_i∈P
10.    更新 t_i 的所有未分配父任务和子任务的 EST 和 LFT
11.    递归调用 DeadlineDistribution(t_i)
12.  end for
13. end while
end procedure
```

算法 2 给出了子截止时间分配 AssignSubDeadline 的过程。算法沿着局部决定路径 P 为每个任务分配子截止时间。算法将局部决定路径 P 作为输入，即算法开始于 P 的第一个节点 t_{first}，结束于 P 的最后一个节点 t_{last}，如步骤 1－步骤 2 所示。步骤 3－步骤 8 将截止时间根据正比于平均执行时间的方式沿路径 P 的任务进行分配。

定义路径 P 的缩放因子 ρ 为：

$$\rho=\frac{(LFT(t_{last})-EST(t_{first})-PathLength(P))}{\sum_{i\in P}MET(t_i)} \tag{20}$$

其中，$PathLength(P)$ 表示局部决定路径 P 的长度。

定义 3（路径长度）　沿该路径的任务权值与边的权值之和。

缩放因子 ρ 用于在分配截止时间至任务时，调整任务的执行时间为期望执行时间 $EET(t_i)$，表示为：

$$EET(t_i)=MET(t_i)+MET(t_i)\times\rho \tag{21}$$

最后，路径 P 上任务的子截止时间的计算式为：

$$Sub\_deadline(t_i)=EST(t_i)+EET(t_i) \tag{22}$$

算法 2　路径 P 上任务的子截止时间分配

```
Procudure AssignSubDeadline(P)
1. 定义路径 P 的最后一个任务为 t_last
2. 定义路径 P 的第一个任务为 t_first
3. 计算路径 P 的长度 PathLength(P)
4. 利用式(20)计算缩放因子 ρ
5. t_i←t_first
6. while 任务 t_i 不为空
7.   利用式(21)、式(22)计算 EET(t_i)和 Sub_deadline(t_i)
8.   t_i←路径 P 的下一任务
9.   end while
end procedure
```

3.2　任务调度选择排序

工作流任务集的顺序约束必须确保先执行父（前驱）任务，后执行子（后继）任务。为了实现该目标，必须产生满足该约束的有序任务调度序列，本节提出基于自顶向下任务分级的方式来对任务调度进行排序。

定义 4　工作流结构中任务的自顶向下分级 $Level_{top\text{-}down}(t_i)$ 表示当前任务 t_i 至出口任务的最长路径，定义为：

$$Level_{top\text{-}down}(t_i)=\begin{cases}MET(t_i), & t_i\text{ 为出口任务}\\ MET(t_i)+\max\limits_{t_j\in succ(t_i)}\{MC_{i,j}+Level_{top-down}(t_j)\}, & \text{否则}\end{cases} \tag{23}$$

其中，$succ(t_i)$ 表示任务 t_i 的直接子任务集合。

算法 3 给出了任务调度选择排序过程。步骤 1 形成任务的倒序排列集合，即从出口任务排序（自顶向下分级需要倒序计算得到），然后在步骤 2－步骤 5 计算每个任务的自顶向下分级，最后在步骤 6 中根据任务自顶向下分级的降序排列对任务进行排序，此排序即为任务的调度序列。

算法 3　任务排序

```
procedure TaskOrdering()
1. 定义包含所有任务 t_i∈T 的倒序序列列表为 t_list
2. 初始化 t_list 中所有任务的自顶向下分级 Level_top-down(t_i) ←0
3. for t_list 中的每个任务 t_i
4.   利用式(23)计算任务 t_i 的自顶向下分级值
5. end for
6. 根据任务自顶向下分级值的降序对任务进行排序
end procedure
```

3.3　基于电压/频率等级的资源选择

该阶段的目标是为每个任务选择带有合适电压/频率等级的最优目标资源，在满足任务的子截止时间的前提下使总体能耗达到最小化。算法 4 给出基于电压/频率等级的资源选择过程。

步骤 1 首先得到可用资源列表，步骤 2－步骤 6 从任务调度序列中选择就绪任务并将其调度至最优可用目标资源 r_j，此时资源 r_j 的频率是使得资源计算能耗与通信能耗最小且满足任务子截止时间约束的频率 $f_{j,k}$，即：

$$\min E_i+\sum_{t_s\in pred(t_i)}EL_{sj} \tag{24}$$

约束条件为：

$$\begin{cases}AST(t_i,r_j)+ET(t_i,r_j)\leqslant Sub\_deadline(t_i)\\ ET(t_i,r_j)\geqslant ET^{\min}(t_i,r_j)\end{cases} \tag{25}$$

其中，$AST(t_i,r_j)$表示任务 t_i 在资源 r_j 上的实际开始时间。频率 $f_{j,k}$ 高于或等于 f_{best}，计算方式为：

$$f_{best}=\frac{f_{j,max}\times ET^{min}(t_i,r_j)}{ET(t_i,r_j)} \tag{26}$$

步骤 7－步骤 10 表明若 CPU 空闲或执行通信过程时，可以进一步通过按比例缩小电压/频率等级至最小值的方式来降低工作流执行的总体能耗。

算法 4 资源选择

Procedure ResourceSelection()
1. 定义包含所有可用资源的列表 r_list
2. for 每个任务排序中的任务 t_i
3. for 资源列表 r_list 中的每个资源 r_j
4. 利用基于式(26)的 DVFS 寻找任务 t_i 在资源 r_j 上的时间槽，使得总能耗(式(24))在满足约束条件(式(25))下最小化
5. end for
6. end for
7. for 每个时间槽中的每个资源 r_j
8. if r_j 为空闲或执行通信
9. 置该资源为低功耗状态，即最低电压/频率等级
10. end for
11. end procedure

3.4 算法过程

算法 5 给出 QCWES 算法的过程。算法输入为由任务集 T 和边集 E 构成的工作流结构 G，及云资源集 R；算法输出为满足截止时间 D_{user} 约束的工作流能效调度方案。步骤 3 首先将所有可用资源构建为资源列表 R，考虑到任务间的顺序依赖和截止时间约束，需要先执行截止时间分配，因此在正式调度任务之前需要得到 EST 和 LFT。为了得到这两个参数，需要计算每个未调度任务的执行时间和通信时间。因此，在步骤 4－步骤 7 中，需要计算最大频率时的平均执行时间 $MET(t_i)$和平均通信时间 $MC_{p,i}$。步骤 8－步骤 9 中，用户定义的截止时间 D_{user} 被分配至出口节点，同时标记其为已分配。然后，步骤 10 执行算法 1，即基于正比于平均执行时间的方式分配截止时间至每个任务。步骤 11 执行算法 3，即基于自顶向下分级值对调度任务进行排序。步骤 12 执行算法 4，即基于合适的电压等级为每个任务选择最优目标资源并执行任务，同时满足任务子截止时间约束，并最小化能耗。

算法 5 QCWES 算法

Procedure QCWES (G(T,E),D,R)
输入：由任务集 T 和边集 E 组成的工作流，资源集合 R
输出：满足截止时间 D_{user} 的工作流能效调度方案
1. 将所有可用资源置入资源列表 R
2. for 每个任务 $t_i\in T$
3. 计算任务平均执行时间 $MET(t_i)$和平均通信时间 $MC_{p,i}$
4. 利用式(4)和式(5)计算 $EST(t_i)$和 $LFT(t_i)$
5. end for
6. 初始化出口任务 $Sub\_deadline(t_{exit})\leftarrow D_{user}$
7. 标识出口任务 t_{exit} 为已分配
8. 调用截止时间重分配算法 DeadlineDistribution(t_{exit})
9. 调用任务调度选择排序算法 TaskOrdering()
10. 调用资源选择算法 ResourceSelection()
end procedure

3.5 算法时间复杂度分析

考虑算法输入为拥有 n 个任务和 e 条边的工作流结构，可用云资源数量为 m，算法 5 的步骤 4－步骤 8 需要计算每个任务的 $MET(t_i)$，$MC_{p,i}$，$EST(t_i)$和 $LFT(t_i)$，其时间复杂度为 $O((n+e)\times m)$。对于工作流 DAG，最大边数为 $(n-1)(n-2)/2$，因此 $e\approx O(n^2)$。则步骤 4－步骤 8 的时间复杂度为 $O(n^2\times m)$。步骤 8 开始需要进行子算法的调用过程。

算法 1 是递归过程。算法首先从出口任务开始计算，然后逐步计算工作流的其他任务。在步骤 1－步骤 13 的循环内部，算法需要计算每个任务的进入边，即包括所有工作流的边 e。进一步，算法需要计算局部决定路径 P，其时间复杂度为 $O(h)$，h 为工作流结构的高度。然后，算法 1 调用算法 2 为路径 P 上的任务分配子截止时间，其时间复杂度为 $O(h)$。最后，在步骤 9－步骤 10 中，算法更新所有未分配子节点和父节点的 EST 和 LFT。在最差情况下，一个任务最多拥有 $n-1$ 个子任务和父任务。因此，更新所有 n 个节点的参数的时间复杂度为 $O(n^2)$。综上，算法 1 的总体时间复杂度为 $O(e\times h+e\times O(h)+n^2)=O(n^2\times h)$。

算法 3 通过自顶向下分级的方式计算任务调度序列。步骤 3－步骤 5 通过考虑任务的子任务计算每个任务的自顶向下分级值，在最差情况下，一个节点的子节点的最多个数为 $n-1$，其时间复杂度为 $O(n^2)$。然后，任务按自顶向下分级值的降序进行排列，时间复杂度为 $O(n\log n)$。因此，算法 3 的时间复杂度为 $O(n^2+n\times\log n)$。

算法 4 中步骤 2－步骤 6 的循环用于产生满足子截止时间约束并带有相应电压等级的任务与资源映射解，其时间复杂度为 $O(n\times m)$。然后，为了降低处理器的空闲/通信时间的能耗，步骤 7－步骤 10 执行于每个资源和相应的每个时间槽，其时间复杂度为 $O(m\times s)$，s 为时间槽的最大数量。因此，算法 4 的时间复杂度为 $O(n\times m+m\times s)$。

结合 3 个子算法的时间复杂度，算法 QCWES 的时间复杂度为 $O(n^2\times m+n^2\times h+n^2+n\times\log n+n\times m+m\times s)=O(n^2\times m+n^2\times h+m\times s)$。

4 仿真实验

4.1 仿真模型

为了验证算法的性能，通过 WorkFlowSim[16] 下的仿真实验对工作流能效调度算法进行模拟仿真。为了模拟工作流中的任务依赖性，以工作流产生器 Pegasus[17] 产生的随机工作流结构和基于高斯消元法(Gaussian Elimination，GE)的现实工作流结构进行实验。为了验证性能，实验设置了不同规模的任务数量，对于不同的工作流结构，用户定义的截止时间约束也设置为不同，具体方式为：基于工作流的关键路径长度为 CPL，截止时间分别设置为 CPL 的 5%(相对紧密的截止时间)，10%，20%和 30%(相对宽松的截止时间)。

每个云资源的 CPU 均具有 DVFS 能力，即可以运行于不同的电压/频率等级。对于资源 r_j，电压缩放等级集合 V_j 随机均匀分布在如表 2 所列的 3 个集合中。设置每个处理器在最大电压等级的活动功耗(执行任务时)取值服从均值为 2 倍平均功耗的均衡分布，且平均功耗设置为 200。不同电压下处理器的功耗通过式(7)计算。

表 2　资源处理能力参数

等级	类型 1			类型 2			类型 3		
	电压/V	频率/GHz	相对速率/%	电压/V	频率/GHz	相对速率/%	电压/V	频率/GHz	相对速率/%
1	1.20	1.8	100.00	1.35	0.60	100.00	1.48	1.4	100.00
2	1.15	1.6	88.88	1.27	0.55	91.66	1.44	1.2	96.76
3	1.10	1.4	77.77	1.20	0.50	83.33	1.31	1.0	88.14
4	1.05	1.2	66.66	1.00	0.25	41.66	1.18	0.8	79.51
5	1.00	1.0	55.55	0.90	0.13	20.83	0.96	0.6	64.42
6	0.90	0.8	44.44						

为了进行能效比较，选择相关工作中的能量感知调度算法 EAS[9]、基于复制与能量平衡调度算法 EDB[10] 及能量感知 DAG 调度算法 EADAG[11] 3 种算法作为比较算法；同时，在实验结果中，以 4 种算法在经典工作流调度算法 HEFT[4] 上得到的增益作为结果进行比较。

4.2　仿真结果

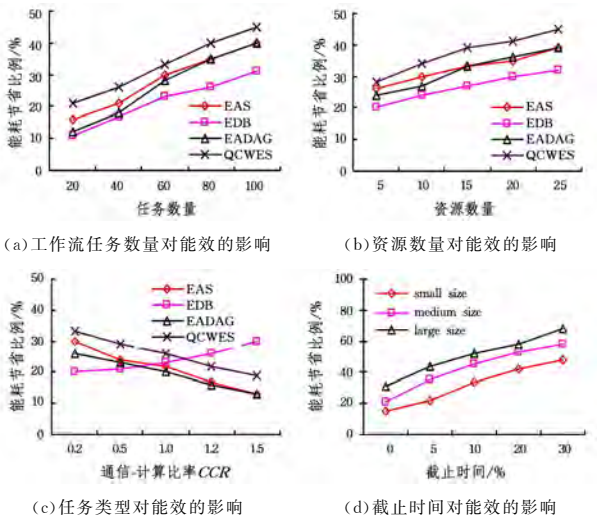
实验 1 通过随机工作流结构对算法进行能效分析。随机工作流的规模设置为不同的任务数量，具体为：20（small size）、40 和 60（medium size）、80 和 100（large size）。任务 t_i 在资源 r_j 上的执行时间服从均值为 2 倍的平均执行时间（实验中设置为 20）的均匀分布，每条边的权值服从均值为平均执行时间和任务通信-计算比 *CCR* 之积的均匀分布，*CCR* 表示工作流任务中通信密集型任务与计算密集型任务的比例。

图 2(a)给出工作流任务数量对能效的影响，此时，截止时间设置为 *CPL* 的 5%，资源数量为 10。可以看出，QCWES 的能效最高，在不同的任务规模下可以节省约 20%～45%的能耗。同时，能耗节省比例会随着工作流规模的增加而增加，这主要是由于任务依赖性导致处理器存在空闲时间槽。EDB 利用前驱任务复制机制调度工作流，可以降低调度长度，但缺少利用 DVFS 的能效技术。而 EAS 利用相对优越函数为就绪任务选择适当电压等级的资源，却没有考虑用户的截止时间约束。与其他算法相比，QCWES 通过有效的截止时间再分配来实现 QoS，而资源空闲和通信阶段的能耗降低也极大地提高了工作流的调度能效。

图 2(b)给出任务规模为 60 个时资源数量对能效的影响。可以看出，能耗节省会随着资源数量的增加而增加，这是由于资源数量的增加可以增加最优资源的选择机会。进一步，资源数量越多，单个资源上的任务越少，资源可以更好地进行电压等级缩放，从而带来更高的能效。然而，当资源增加到一定数量时，空闲资源的能耗开销会趋近于缓和。

图 2(c)给出拥有 60 个任务和 10 个资源规模时 *CCR* 值对能效的影响。*CCR* 为通信密集型任务与计算密集型任务的比例，当 $CCR<1$ 时，即以计算密集型任务为主的工作流中，最多可节省 33%的能耗，这是由于 DVFS 对于执行计算密集型任务更加有利。当 $CCR>1$，即以通信密集型任务为主的工作流中，能耗节省虽然相对降低，但仍可达到 20%左右，这是由于在通信阶段，处理器已经降低至最小电压等级，导致仍可降低能耗。对于 EDB，能效会随着 *CCR* 值的增加而增加，这是由于使用了任务副本机制，从而降低了通信能耗，因此，对于算法而言，通信密集型工作流会得到更高的能效。而 EAS 的能效则是随着 *CCR* 值的增加而降低，这是由于算法并未考虑降低通信能耗所致。

图 2(d)给出不同截止时间下 QCWES 能效的变化。实验观察了 3 种任务规模工作流的执行结果。可以看出，QCWES 的能效会随着截止时间的增加而单调增加；同时，图中结果表明约 70%的能耗节省仅会增加约 30%的工作流执行时间，这说明在能效与执行效率性能之间相互关联与均衡。

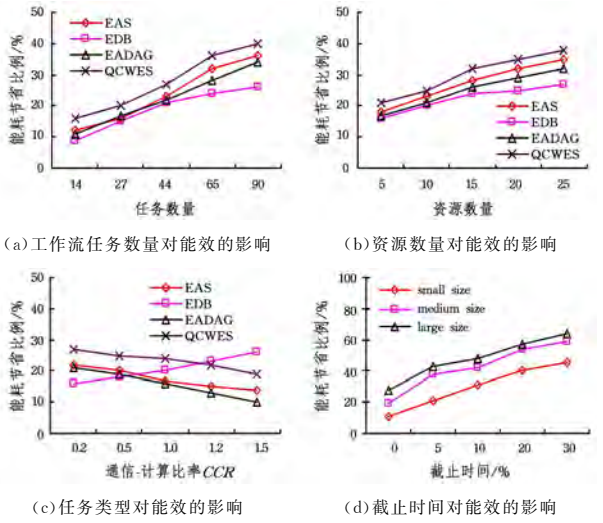


(a)工作流任务数量对能效的影响　(b)资源数量对能效的影响

(c)任务类型对能效的影响　(d)截止时间对能效的影响

图 2　随机工作流结构的能效

实验 2 通过一种基于高斯消元法 GE 的现实工作流结构对算法进行能效分析。在 GE 工作流中，任务结构是相对固定的，任务数量等于 $(m^2+m-2)/2$，其中，m 表示矩阵规模，因此任务数量可选择 14，27，44，65 和 90。任务的执行时间与边的权值取值方法与实验 1 相同。从图 3 可以看出，GE 工作流得到的结构与随机工作流的结果是类似的，这表明 QCWES 算法可以适应于不同的工作流结构以进行能效调度。



(a)工作流任务数量对能效的影响　(b)资源数量对能效的影响

(c)任务类型对能效的影响　(d)截止时间对能效的影响

图 3　GE 工作流结构的能效

结束语　能耗是云计算面临的新课题，尤其对于云环境下的工作流调度，能耗更为突出。为了在满足工作流截止时间 QoS 约束的同时最小化工作流执行能耗，设计了一种 QoS 约束工作流能效调度算法。所提算法将工作流调度划分为

（下转第 327 页）

IEEE，2015：432-438.

[9] RUI L，HONGLIANG Z，YANG X，et al. Remote NAT detect algorithm based on support vector machine[C]// 2009 International Conference on Information Engineering and Computer Science. 2009.

[10] ABT S，DLETZ C，BAIER H，et al. Passive remote source nat detection using behavior statistics derived from netflow[C]// IFIP International Conference on Autonomous Infrastructure，Management and Security. Springer，Berlin，Heidelberg，2013：148-159.

[11] GOKCEN Y，FOROUSHANI V A，HEYWOOD A N Z. Can we identify NAT behavior by analyzing Traffic Flows? [C]// Security and Privacy Workshops (SPW)，2014 IEEE. IEEE，2014：132-139.

[12] KOMÁREK T，GRILL M，PEVNY T. Passive NAT detection using HTTP access logs[C]// 2016 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE，2016：1-6.

[13] BI J，ZHAO L，ZHANG M. Application presence fingerprinting for NAT-aware router[C]// Knowledge-Based Intelligent Information and Engineering Systems. Springer Berlin/Heidelberg，2006：678-685.

[14] The Bro Network Security Monitor[OL]. http://www.bro.org.

[15] HOLMES G，DONKIN A，WITTEN I H. Weka：A machine learning workbench[C]// Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems. IEEE，1994：357-361.

（上接第 309 页）

3 个阶段：截止时间重分配、任务调度选择排序和基于电压/频率等级的资源选择。通过这种三阶段方式，算法最终可以得到满足工作流 QoS 约束且带有合适电压/频率等级的最优工作流调度方案，使得工作流的总体能耗达到最小。通过随机工作流结构和一种现实科学工作流结构的仿真实验证明了算法的有效性和可行性。进一步的研究将考虑更多维度的 QoS 约束，如预算约束、安全约束等，以研究多 QoS 约束下的能效调度问题。

参 考 文 献

[1] BUYYA R，YEO S，VENUGOPAL S，et al. Cloud computing and emerging IT platforms：Vision，hype，and reality for delivering computing as the 5th utility[J]. Future Generation Computer Systems，2011，25(6)：599-616.

[2] LIU L，ZHANG M，LIN Y，et al. A survey on workflow management and scheduling in cloud computing[C]// 14th IEEE/ACM International Symposium on Cluster，Cloud and Grid Computing. USA：IEEE Press，2014：837-846.

[3] ARROBA P，RISCO-MART J L，ZAPATER M，et al. Server power modeling for run-time energy optimization of cloud computing facilities[C]// International Conference on Sustainabiility and building. 2014：401-410.

[4] TOPCUOGLU H，HARIRI S，WU M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE Transactions Parallel Distributed Systems，2012，13(3)：260-274.

[5] SELVI S，MANIMEGALAI D. Task Scheduling Using Two-Phase Variable Neighborhood Search Algorithm on Heterogeneous Computing and Grid Environments[J]. Arabian Journal for Science and Engineering，2015，40(3)：817-844.

[6] LEE Y C，HAN H，ZOMAYA A Y，et al. Resource-efficient workflow scheduling in clouds[J]. Knowledge-Based Systems，2015，80(C)：153-162.

[7] ABUDHAGIR U S，SHANMUGAVEL S. A Novel Dynamic Reliability Optimized Resource Scheduling Algorithm for Grid Computing System[J]. Arabian Journal for Science and Engineering，2014，39(10)：7087-7096.

[8] ARABNEJAD H，BARBOSA J G. A Budget Constrained Scheduling Algorithm for Workflow Applications[J]. Journal of Grid Computing，2014，12(4)：665-679.

[9] LEE Y C，ZOMAYA A Y. Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling[C]// IEEE/ACM International Symposium on CLUSTER Computing and the Grid. IEEE Computer Society，2009：92-99.

[10] ZONG Z，MANZANARES A，RUAN X，et al. EAD and PEBD：Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters[J]. IEEE Transactions on Computers，2011，60(3)：360-374.

[11] BASKIYAR S，ABDEL-KADER R. Energy aware DAG scheduling on heterogeneous systems[J]. Cluster Computing，2013，13(4)：373-383.

[12] 景维鹏，吴智博，刘宏伟，等. 多 DAG 工作流在云计算环境下的可靠性调度方法[J]. 西安电子科技大学学报(自然科学版)，2016，43(2)：83-88.

[13] 王润平，陈旺虎，段菊. 一种科学工作流的云数据布局与任务调度策略[J]. 计算机仿真，2015，32(3)：421-426.

[14] 曹斌，王小统，熊丽荣，等. 时间约束云工作流调度的粒子群搜索方法[J]. 计算机集成制造系统，2016，22(2)：372-380.

[15] 杨玉丽，彭新光，黄名选，等. 基于离散粒子群优化的云工作流调度[J]. 计算机应用研究，2014，31(12)：3677-3681.

[16] CHEN W，DEELMAN E. WorkflowSim：a toolkit for simulating scientific workflows in distributed environments[C]// IEEE 8th International Conference on E-Science (e-Science). IEEE，2012：1-8.

[17] JUVE G，CHERVENAK A，DEELMAN E，et al. Characterizing and profiling scientific workflows[J]. Future Generation Computer Systems，2013，29(3)：682-692.