doi:10.3969/j.issn.1673-9833.2023.06.001

能耗约束下基于预算等级的调度长度最小化算法

刘丹彤，张龙信，杨　佳，暴子豪，艾明慧

（湖南工业大学 计算机学院，湖南 株洲 412007）

摘　要：为了解决异构云系统中能耗约束条件下的调度长度最小化问题，提出了一种新颖的预算等级（BL）能耗预分配策略，并设计了一种能耗约束下最小化调度长度算法（BLMSL），BLMSL 算法包含任务优先级队列建立、任务能耗约束预分配、最佳处理器与频率组合选择 3 个阶段。在 Epigenomics 和 LIGO 两类科学工作流上进行实验，结果表明，在满足能耗约束的前提下，BLMSL 算法相比当前最先进的启发式算法能获得更小的调度长度，具有明显优势。

关键词：异构云系统；能耗；调度长度；并行应用

中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1673-9833(2023)06-0001-08

引文格式：刘丹彤，张龙信，杨　佳，等 . 能耗约束下基于预算等级的调度长度最小化算法 [J]. 湖南工业大学学报，2023，37(6)：1-8.

Scheduling Length Minimization Algorithm Based on Budget Level Under Energy Consumption Constraint

LIU Dantong，ZHANG Longxin，YANG Jia，BAO Zihao，AI Minghui

（College of Computer Science，Hunan University of Technology，Zhuzhou Hunan 412007，China）

Abstract：In view of a solution of the minimizing scheduling length under energy constraints in heterogeneous cloud systems, a novel budget level (BL) energy preallocation strategy has thus been proposed, with a scheduling length minimization algorithm designed under energy constraints (BLMSL). The BLMSL algorithm consists of three stages: establishment of task priority queues, preallocation of task energy consumption constraints, and selection of the optimal processor and frequency combination. The results of experiments conducted on two types of scientific workflows, Epigenomics and LIGO, show that under the premise of meeting energy consumption constraints, the BLMSL algorithm is characterized with significant advantages in achieving a smaller scheduling length compared to the most advanced heuristic algorithms.

Keywords：heterogeneous cloud system；energy consumption；scheduling length；parallel application

收稿日期：2022-09-11

基金项目：国家重点研发计划“云计算和大数据”重点专项基金资助子课题（2018YFB1003401）；国家自然科学基金资助项目（61702178）；湖南省自然科学基金资助项目（2023JJ50204）；湖南省教育厅科研基金资助项目（20C0625）；湖南省大学生创新训练基金资助项目（S202111535056）

作者简介：刘丹彤（1997-），女，陕西咸阳人，湖南工业大学硕士生，主要研究方向为云计算和边缘计算，E-mail：dantong_83@163.com

通信作者：张龙信（1983-），男，湖南长沙人，湖南工业大学副教授，博士，硕士生导师，主要研究方向为大数据处理，高性能计算，E-mail：longxin.zhang@163.com

1 研究背景

异构云系统（heterogeneous cloud system，HCS）由多个计算单元连接组成，其可扩展性强，在分析处理海量数据和进行大规模科学计算时发挥着重要作用[1−2]。随着人工智能、大数据和云计算等技术的飞速发展，计算的应用需求和系统性能都趋向多样化和复杂化，系统架构集成度不断提高，导致计算功耗显著增加，产生了更高昂的电力成本，甚至对环境造成了污染[3]。降低计算系统的能耗已成为当前 HCS 开发和使用所面临的主要瓶颈，也是影响经济和生态技术发展的重要因素。推进节能能够提高计算设备的利用率，促进经济可持续发展[4]。

近年来，在能耗约束下进行科学工作流调度已经引起了广泛的研究。在电压和频率可变的虚拟机（virtual machine，VM）上进行能耗感知任务调度一直被学者们广泛研究[5]。动态电压和频率缩放（dynamic voltage and frequency scaling，DVFS）作为一种被广泛使用的降低系统能源消耗的经典技术，主要通过调整 VM 上的电源电压和频率来进行功率感知任务调度，提高能量效率比，从而实现对系统能耗的控制[6−7]。

HCS 中基于功率感知的任务调度长度最小化一直是一个研究热点。Tang Z. 等[8]提出了利用空闲时间的回收来合并效率较低的 VM 以实现对空闲时间的有效利用，减少能量浪费。Song J. 等[9]提出了一种有效的调度算法，在能耗受限的情况下，通过对松弛能量的相对平均分配以最小化应用程序的调度长度。王兰等[10]将节点根据其通信开销和计算开销划分为单独的队列，再将关键节点划分到适合的队列中，减少了工作流的完工时间并提高了处理器的调度效率。Li K. Q.[11]将关注重点放在静态功耗上，通过解析方法建立了具有给定能耗约束的并行任务集的最小调度长度，同时建立了给定调度长度约束下任务集最小能耗的下界。该解析思想为具有动态和静态功耗工作流调度提供了分析方法，以评估启发式算法性能。A. K. Maurya 等[12]在基于能量感知服务的水平协议和不改变最大完工时间的前提下降低频率，同时利用预测最早完成时间算法以计算工作流的完工时间。文献 [13] 提出了两种经典的算法，其中异构的最早完成时间优先算法（heterogeneous earliest-finish-time，HEFT）是一种基于插入思想的最小完成时间算法，而关键任务置于同一处理器算法（critical-path-on-a-processor，CPOP）则在任务优先级阶段进行了改进，并最小化关键任务的执行时间。M. Sulaiman 等[14]将启发式算法和元启发式算法的思想相结合，利用列表和任务复制思想提出了两种基于遗传算法的任务调度算法，能够实现较低的复杂度前提下优化调度长度和平均调度长度比等多项指标。

近年来，Xiao X. R. 等[15]在异构系统上探究了能耗约束条件下并行应用的调度长度最小化问题，并提出了能耗约束下最小化调度长度（minimum schedule length with energy consumption constraint，MSLECC）算法。其主要思想是将工作流的总能耗约束传递给每个任务，从而使得调度过程中的关注点转移到每个任务，然后以启发式的调度方法最小化工作流的调度长度。MSLECC 算法具有良好的性能，但 MSLECC 算法在能耗预分配阶段直接为每个未调度任务分配了其最小能耗，这种预分配方法虽能满足总能耗约束条件，但公平性欠佳，任务集的调度长度并不乐观。因此，需要设计一种高效的调度算法以减少调度长度。

基于此，本文提出预算等级（budget level，BL）分配机制这一全新的概念，将应用程序的能耗约束合理地传递给每个任务。BL 分配机制不再使用最小能量预分配方法分配任务的能耗约束，而是根据任务自身能耗水平作为一种智能权重分摊能耗，避免因任务优先级敏感而使能耗分配不公。本文基于这一分配策略提出了能耗约束下最小化调度长度算法（minimizing scheduling length under energy constraints based on budget level，BLMSL），能在能耗约束条件下取得较小的调度长度，减少任务优先级对能耗分配的影响。

2 基本模型

2.1 应用模型

本文应用的一个典型工作流图如图 1 所示。

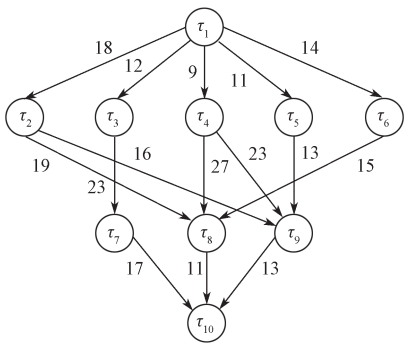

图 1 一个典型的工作流图

Fig. 1 A typical workflow diagram

本文的目标计算平台由一组计算、存储能力各异的

VM组成，VM资源池用$VM=\{vm_1, vm_2, \cdots, vm_{|VM|}\}$表示，其中$|VM|$为虚拟机的数量。并行应用任务集（directed acyclic graph，DAG）通常使用有向无循环图$G=(N, E, C, \boldsymbol{W})$表示，其中$N$为$G$中任务$\tau_i$集合，即$\tau_i \in N$。$E$表示$G$中直接相连两任务节点的通信边集合，每条边$e_{i,j} \in E$表示连接任务$\tau_i$和任务$\tau_j$的通信边。$C$为直接相连两节点间通信边的时间开销集合。若$\tau_i$和$\tau_j$分配给不同$VM$，$c_{i,j}$为$\tau_i$和$\tau_j$间的通信时间。$\boldsymbol{W}$为任务在不同$VM$上执行的通信成本集合的矩阵，$w_{i,k}$表示任务$\tau_i$分配在$vm_k$上执行时所需的计算开销。

对于G中任意直接相连的两个任务节点τ_i和τ_j，$e_{i,j} \in E$表示只有当任务τ_i执行完毕，并且任务τ_j执行所依赖的所有数据准备就绪，τ_j才可被执行。在给定的DAG中，没有前置任务的任务为入口节点，记做τ_{entry}；没有后续任务的任务为出口节点，记做τ_{exit}。表1显示图1所示DAG中各任务在不同VM上的计算开销。

表 1　任务的计算开销

Table 1　Task calculation cost

$task_i$	vm_1	vm_2	vm_3
τ_1	14	16	9
τ_2	13	19	18
τ_3	11	13	19
τ_4	13	8	17
τ_5	12	13	10
τ_6	13	16	9
τ_7	7	15	11
τ_8	5	11	14
τ_9	18	12	20
τ_{10}	21	7	16

2.2　功率和能耗模型

DVFS基于电源电压与工作频率的线性关系，依靠降低频率减小电源电压，达到节能的目的。本文假设一个支持DVFS的系统，采用广泛使用的系统级功率模型为HCS建模，能耗表达式为

$$\varphi(\phi)=\varphi_{\text{s}}+h(\varphi_{\text{ind}}+\varphi_{\text{d}})。\tag{1}$$

式中：φ_{s}为静态功率，其主要包括维持电路工作的基本消耗及存储器休眠状态时的能量开销，系统在打开或关闭时，相关的开销较大，使得φ_{s}始终处于消耗状态；$\varphi_{\text{ind}}+\varphi_{\text{d}}$表示动态功率，其中$\varphi_{\text{ind}}$为与频率无关的动态功率，只有通过关闭整个系统的电源才能消除；h为系统状态，指示当前系统是否消耗动态功率（系统处于激活状态时，h=1；系统处于关闭状态时，h=0）；φ_{d}为系统动态能量消耗，为与频率相关的动态功耗，通常表示为

$$\varphi_{\text{d}}=C_{\text{eff}}\times V_{\text{dd}}^2\times\phi\,,\tag{2}$$

式中：C_{eff}为有效负载电容；V_{dd}为供应电压；ϕ为时钟频率。

由$\phi \propto V_{\text{dd}}$（$0<\gamma<1$），易得$V_{\text{dd}} \propto \phi^{1/\gamma}$，则动态功耗与频率间的关系为$\varphi_{\text{d}}=C_{\text{eff}}\cdot\phi^{\epsilon}$，其中$\epsilon=1+2/\gamma \geqslant 3$代表与特定$VM$相关的动态幂指数。

当系统以较小的频率运行时，由于φ_{ind}的存在，系统总能耗的减少并不明显，因此存在最低节能频率，其表达式为

$$\phi_{\text{ee}}=\sqrt[\epsilon]{\frac{\varphi_{\text{ind}}}{(\epsilon-1)C_{\text{eff}}}}。\tag{3}$$

VM的频率范围是从最小值$\phi_{\min}$到最大值$\phi_{\max}$，其实际频率ϕ_h应是从$\phi_{\min}$到$\phi_{\max}$不等。因此，执行每个任务的最低能效频率ϕ_{low}为

$$\phi_{\text{low}}=\max\left(\phi_{\min},\phi_{\text{ee}}\right)。\tag{4}$$

由于系统中VM的异构性，本文定义以下集合：与频率相关的动态功率φ_{d}集合，

$$\left\{\varphi_{1,\text{d}},\varphi_{2,\text{d}},\cdots,\varphi_{|VM|,\text{d}}\right\};$$

与频率无关的动态功率φ_{ind}集合，

$$\left\{\varphi_{1,\text{ind}},\varphi_{2,\text{ind}},\cdots,\varphi_{|VM|,\text{ind}}\right\};$$

有效电容C_{eff}的集合，

$$\left\{C_{1,\text{eff}},C_{2,\text{eff}},\cdots,C_{|VM|,\text{eff}}\right\};$$

动态幂指数ϵ的集合，

$$\epsilon=\left\{\epsilon_1,\epsilon_2,\cdots,\epsilon_{|VM|}\right\};$$

实际有效频率的集合，

$$\left\{\begin{array}{c}\left\{\phi_{1,\text{low}},\phi_{1,\alpha},\cdots,\phi_{1,\max}\right\}\\ \left\{\phi_{2,\text{low}},\phi_{2,\alpha},\cdots,\phi_{2,\max}\right\}\\ \vdots\\ \left\{\phi_{|VM|,\text{low}},\phi_{|VM|,\alpha},\cdots,\phi_{|VM|,\max}\right\}\end{array}\right\}。$$

因此，任务τ_i以频率$\phi_{k,h}$在vm_k上执行时所消耗的功率为

$$E\left(\tau_i,vm_k,\phi_{k,h}\right)=\varphi_{k,h}\times w_{i,k}\times\phi_{k,\max}/\phi_{k,h}\,,\tag{5}$$

其中，

$$\varphi_{k,h}=\varphi_{k,\text{ind}}+C_{k,\text{eff}}\times\left(\phi_{k,h}\right)^{\epsilon_k}。\tag{6}$$

3　BLMSL 算法

3.1　相关知识

为了便于理解，本文进行如下定义。

定义 1　给定在vm_k上以频率$\phi_{k,h}$执行的任务

τ_i，其最早开始时间表示为 $EST(\tau_{\text{entry}}, vm_k, \phi_{k,h})$，其计算表达式如下：

$$\begin{cases} EST\left(\tau_{\text{entry}}, vm_k, \phi_{k,h}\right)=0, \\ EST\left(\tau_i, vm_k, \phi_{k,h}\right)= \\ \max_{\tau_j\in \text{pred}(\tau_i)}\left\{AFT\left(\tau_j\right)+c'_{i,j},\ avail[k]\right\}, \end{cases} \tag{7}$$

式中：$AFT(\tau_j)$ 为任务 τ_j 实际执行完成时刻；$c'_{i,j}$ 为任务 τ_i 与 τ_j 之间的通信时间（当两任务被分配至同一 *VM* 时通信时间为 0）；$avail[k]$ 为 *VM* 的可用时间。

定义 2 在 vm_k 上以频率 $\phi_{k,h}$ 执行的任务 τ_i 的最早完成时间为 $EST(\tau_{\text{entry}}, vm_k, \phi_{k,h})$，其计算公式如下：

$$EFT\left(\tau_i, vm_k, \phi_{k,h}\right)=EST\left(\tau_i, vm_k, \phi_{k,h}\right)+w_{i,k}\times\phi_{k,\max}/\phi_{k,h}, \tag{8}$$

式中：$EST(\tau_{\text{entry}}, vm_k, \phi_{k,h})$ 为在 vm_k 上执行的任务 τ_i 的最早开始时间；$w_{i,k}$ 为任务 τ_i 以最大频率在 vm_k 上的运行时间。

定义 3 调度长度（*SL*）是并行应用集从入口任务开始执行，到出口任务执行完毕所花费的总时间，即出口任务的实际完成时间与入口任务的实际开始时间的差值，可表示为

$$SL(G)=AFT(\tau_{\text{exit}})-AST(\tau_{\text{entry}}), \tag{9}$$

式中 $AST(\tau_{\text{entry}})$ 为入口任务的实际开始时间。

3.2 问题描述

本文所研究的问题是在满足用户能耗约束的条件下，针对用户提交的工作流任务图，以最少的时间完成调度，提高用户服务质量。即在不超过能耗约束的条件下，为每个任务分配具有适当频率的可用 *VM*，同时使得 DAG 应用程序的调度长度最小。

能耗约束下并行应用的调度长度最小化问题可以形式化表示如下：

$$\text{Min: } makespan(G), \tag{10}$$

$$\text{s.t.: } E(G)=\sum_{i=1}^{|N|}E\left(\tau_i, vm_k, \phi_{k,h}\right)\leqslant E_{\text{bdgt}}(G)。 \tag{11}$$

式中：$E(G)$ 为工作流所消耗的实际能量；$E_{\text{bdgt}}(G)$ 为工作流的预算能耗约束。

由于工作流总能量是所有任务能量之和，工作流的最小能量值 $E_{\min}(G)$ 和最大能量值 $E_{\max}(G)$ 可表示为

$$E_{\min}(G)=\sum_{i=1}^{|N|}E_{\min}(\tau_i), \tag{12}$$

$$E_{\max}(G)=\sum_{i=1}^{|N|}E_{\max}(\tau_i)。 \tag{13}$$

任务 τ_i 的最小能耗 $E_{\min}(\tau_i)$ 和最大能耗 $E_{\max}(\tau_i)$ 分别以最小和最大频率遍历所有的 *VM* 获得，其表达式如下：

$$E_{\min}(\tau_i)=\min_{vm_k\in VM}E(\tau_i, vm_k, \phi_{k,\max}), \tag{14}$$

$$E_{\max}(\tau_i)=\max_{vm_k\in VM}E(\tau_i, vm_k, \phi_{k,\text{low}})。 \tag{15}$$

3.3 任务优先级

在将任务分配给 *VM* 之前，首先需要确定任务调度的顺序。

定义 4 向上优先等级。在任务的优先依赖关系条件下，并行应用任务集的向上优先等级自 DAG 任务集的出口节点开始递归向上进行计算，记为 ζ。任务 τ_i 的 ζ 值等于其后继任务与 τ_i 之间的通信成本之和的最大值加上其在所有 *VMs* 上的平均计算成本，其计算表达式如下：

$$\zeta\left(\tau_i\right)=\max_{\tau_j\in child(\tau_j)}\left\{\zeta\left(\tau_j\right)+c_{i,j}\right\}+\overline{w}_i。 \tag{16}$$

式中：w_i 为任务 τ_i 的平均执行时间，可通过式（17）计算：

$$\overline{w}_i=\left(\sum_{k=1}^{|VM|}w_{i,k}\right)\Big/|VM|。 \tag{17}$$

同理，任务 τ_i 的向下优先等级 $\varpi(\tau_i)$ 的计算表达式如下：

$$\varpi\left(\tau_i\right)=\max_{\tau_j\in parent(\tau_j)}\{\varpi\left(\tau_j\right)+c_{i,j}+\overline{w}_j\}。 \tag{18}$$

3.4 能耗预分配

定义 5 可改进能量。在用户给定能耗约束且当前工作流的最小能耗已知时，工作流的可改进能量 $E_{\text{adv}}(G)$ 表达式如下：

$$E_{\text{adv}}(G)=E_{\text{bdgt}}(G)-E_{\min}(G)。 \tag{19}$$

定义 6 预算等级（*BL*）。工作流能耗的可支配部分将按照每个任务不同的 *BL* 等级进行分配，任务的 *BL* 计算如下：

$$BL\left(\tau_i\right)=\frac{\zeta\left(\tau_i\right)+\varpi\left(\tau_i\right)}{\sum_{j=1}^{|N|}\left\{\zeta\left(\tau_j\right)+\varpi\left(\tau_j\right)\right\}}。 \tag{20}$$

工作流中任务 τ_i 初始给定的 *BL* 预算等级能耗 $E_{BL}(\tau_i)$ 由其预算等级 *BL* 和最小能耗等因素决定，其计算式如下：

$$E_{BL}\left(\tau_i\right)=E_{\text{adv}}(G)\times BL\left(\tau_i\right)+E_{\min}\left(\tau_i\right)。 \tag{21}$$

在极端情况下，任务的预分配能耗也始终不能超过其能耗上限，因此任务的预分配能耗 $E_{\text{bdgt}}(\tau_i)$ 应满足：

$$E_{\mathrm{bdgt}}(\tau_i)=\min\left\{E_{\mathrm{BL}}(\tau_i),E_{\max}(\tau_i)\right\}。\quad(22)$$

设定一个等待调度的并行应用任务集 $Seq(G)$，该任务集按照 ζ 值降序排列的调度序列为 $\{\tau_{s(1)}, \tau_{s(2)}, \cdots, \tau_{s(|N|)}\}$，设定当前的待调度任务为 $\tau_{\mathrm{seq}(j)}$，则 $\{\tau_{s(1)}, \tau_{s(2)}, \cdots, \tau_{s(j-1)}\}$ 表示已调度完成的任务集合；$\{\tau_{s(j+1)}, \tau_{s(j+2)}, \cdots, \tau_{s(|N|)}\}$ 为未被调度的任务集合，此时任务集的总能耗为：

$$E(G)=\sum_{x=1}^{j-1}E(\tau_{\mathrm{seq}(x)},vm_{pr(\mathrm{seq}(x))},\phi_{pr(\mathrm{seq}(x)),hz(\mathrm{seq}(x))})+E\left(\tau_{\mathrm{seq}(i)},vm_k,\phi_{k,h}\right)+\sum_{y=j+1}^{|N|}E_{\mathrm{bdgt}}\left(\tau_{\mathrm{seq}(y)}\right)。\quad(23)$$

工作流的实际总能耗必须小于等于其能耗约束，即 $E_{\mathrm{seq}(j)}(G)\leqslant E_{\mathrm{bdgt}}(G)$。可以得到：

$$E\left(\tau_{\mathrm{seq}(i)},vm_k,\phi_{k,h}\right)=E_{\mathrm{bdgt}}(G)-\sum_{x=1}^{j-1}E(\tau_{\mathrm{seq}(x)},vm_{pr(\mathrm{seq}(x))},\phi_{pr(\mathrm{seq}(x)),hz(\mathrm{seq}(x))})-\sum_{y=j+1}^{|N|}E_{\mathrm{bdgt}}\left(\tau_{\mathrm{seq}(y)}\right)。\quad(24)$$

根据式（24）可得出将任务集总能耗约束分摊给每个单独任务的方法，则序列中当前待调度任务的能耗约束为：

$$E_{\mathrm{given}}\left(\tau_{\mathrm{seq}(j)}\right)=E_{\mathrm{bdgt}}(G)-\sum_{x=1}^{j-1}E_{\mathrm{act}}(\tau_{\mathrm{seq}(x)},vm_{pr(\mathrm{seq}(x))},\phi_{pr(\mathrm{seq}(x)),hz(\mathrm{seq}(x))})-\sum_{y=j+1}^{|N|}E_{\mathrm{bdgt}}\left(\tau_{\mathrm{seq}(y)}\right)。\quad(25)$$

式（25）使任务集总能耗约束传递给每个任务，并降低了算法时间及复杂度。

3.5　调度长度最小化算法

具有能耗约束的工作流调度长度最小化算法（BLMSL）的伪代码如算法 1 所示。

步骤 1 计算任务集中所有任务的 ζ 值和 ϖ 值，这是 *BL* 预分配策略的基础，并根据 ζ 值降序排列得到任务的调度队列。步骤 3 计算每个任务的 *BL* 值和所能消耗的最小和最大能量。步骤 4~5 计算工作流的最小最大能耗以及可改进能耗 $E_{\mathrm{adv}}(G)$，并根据可改进能耗计算出任务的 $E_{\mathrm{bdgt}}(\tau_i)$。步骤 10~27 将任务依次出队进行调度，确定每个任务的能耗约束 $E_{\mathrm{given}}(\tau_i)$，并选择该能耗约束下具有最小调度长度的 *VM* 和频率组合。其中步骤 13~26 为队列中的任务选择合适的 *VM* 与频率，直至队列中所有任务均完成调度。步骤 29 和 30 分别统计工作流的实际总消耗能量 $E_{\mathrm{act}}(G)$ 和调度长度 $SL(G)$。

遍历任务队列中的所有任务的时间消耗为 $O(N)$，对于就绪任务队列中的每个任务，遍历所有 *VM* 和频率组合为其选择满足能耗约束且具有最小 *EFT* 的 *VM* 的时间复杂度为 $O(|N|\times|VM|\times|F|)$，其中 $|F|$ 表示从 $\phi_{k,\,\mathrm{low}}$ 到 $\phi_{k,\,\max}$ 的离散频率数量。因此，可以得出 BLMSL 算法的时间复杂度为 $O(|N|^2\times|VM|\times|F|)$。

算法 1　BLMSL 调度算法

输入：$G=(N, E, C, W)$，VM，$E_{\mathrm{given}}(G)$；

输出：$E(G)$，$SL(G)$。

1: 递归计算工作流图中每个节点的 ζ 和 ϖ 值；
2: for $i\leftarrow 1$ to $|N|$ do；
3: 计算 $E_{\min}(\tau_i)$ 、$E_{\max}(\tau_i)$ 和 $BL(\tau_i)$；
4: 计算工作流图的 $E_{\min}(G)$、$E_{\max}(G)$；
5: 计算工作流图的 $E_{\mathrm{adv}}(G)$；
6: end for
7: for $i\leftarrow 1$ to $|N|$ do
8: 计算 $E_{\mathrm{BL}}(\tau_i)$ 和 $E_{\mathrm{bdgt}}(\tau_i)$；
9: 工作流图 G 根据节点的 ζ 值降序排列得到任务优先队列 $Seq(G)$
10: while 队列 $Seq(G)$ 非空
11: τ_{tmp} = 队头节点
12: 计算 $E_{\mathrm{given}}(\tau_i)$；
13: for each $vm_k\in VM$ do
14: for each $\phi_{k,\,h}$ from $\phi_{k,\,\max}$ to $\phi_{k,\,\mathrm{low}}$ do
15: 计算 $E_{\mathrm{act}}(\tau_i, vm_k, \phi_{k,\,h})$；
16: if $E_{\mathrm{act}}(\tau_i, vm_k, \phi_{k,\,h})>E_{\mathrm{given}}(\tau_i)$ then
17: continue；
18: end if
19: 计算 $E_{\mathrm{act}}(\tau_i, vm_k, \phi_{k,\,h})$；
20: if $EFT(\tau_i, vm_k, \phi_{k,\,h})<AFT(\tau_i)$
21: $vm_{pr(i)}\leftarrow vm_k$ and $\phi_{pr(i),\,hz(i)}\leftarrow\phi_{k,\,h}$
22: $E_{\mathrm{act}}(\tau_i, vm_{pr(i)}, \phi_{pr(i),\,hz(i)})\leftarrow E(\tau_i, vm_k, \phi_{k,\,h})$
23: $AFT(\tau_i)\leftarrow EFT(\tau_i, vm_k, \phi_{k,\,h})$
24: end if
25: end for
26: end for
27: end while
28: end for
29: 计算 $E_{\mathrm{act}}(G)$；
30: 计算 $SL(G)\leftarrow AFT(\tau_{\mathrm{exit}})$；
31: return $E(G)$，$SL(G)$

3.6 调度实例分析

本节使用图 1 中的 DAG 图来进行本文实验，表 2 列出了 *VM* 的相关参数，如动态功率、有效开关电容和动态功率指数等。能耗约束值 $E_{\text{bdgt}}(G)$ 设置为 $E_{\max}(G)\times\lambda$，λ 的取值为 0.5。当 MSLECC 算法使用表 2 所示参数的 *VM* 调度图 1 中的任务集时，其总调度长度是 129.525 6。

表 2 *VM* 的功率参数

Table 2 *VM* power parameters

vm_k	$\phi_{k,\text{ind}}$	$C_{k,\text{eff}}$	ϵ_k	$\phi_{k,\text{low}}$	$\phi_{k,\max}$
vm_1	0.03	0.8	2.9	0.26	1.0

在相同配置的情况下，表 3 显示了 BLMSL 算法在图 1 所示的任务集中总调度长度为 85.852 4，所消耗能耗为 71.739 2。相比 MSLECC 算法，BLMSL 算法的调度长度减少了 33.73%，能量消耗减少了 9.04%，使得工作流的调度长度和能耗都得到优化。

表 3 使用 BLMSL 算法在图 1 所示的并行应用分配结果

Table 3 Parallel application allocation results as shown in Fig. 1 using BLMSL algorithm

τ_i	$E_{\text{given}}(\tau_i)$	$E_{\text{act}}(\tau_i)$	$vm(\tau_i)$	$AST(\tau_i)$	$AFT(\tau_i)$	$\phi(\tau_i)$
τ_1	8.624 3	8.505 1	vm_3	0.000 0	9.890 1	0.91
τ_4	7.931 4	5.920 0	vm_2	18.890 1	26.890 1	1.00
τ_3	9.973 9	9.130 0	vm_1	21.890 1	32.890 1	1.00
τ_2	9.333 3	9.322 6	vm_3	9.890 1	43.852 4	0.53
τ_5	7.464 0	7.391 3	vm_2	26.890 1	42.743 8	0.82
τ_6	7.550 3	7.450 2	vm_1	32.890 1	48.934 5	0.81
τ_9	9.241 3	8.880 0	vm_2	59.852 4	71.852 4	1.00
τ_7	7.614 8	5.810 0	vm_1	48.939 5	55.939 5	1.00
τ_8	8.551 4	4.150 0	vm_1	62.852 4	67.852 4	1.00
τ_{10}	12.310 9	5.180 0	vm_2	78.852 4	85.852 4	1.00

使用 BLMSL 算法调度图 1 中任务集的结果如图 2 所示，调度长度为 85.852 4。

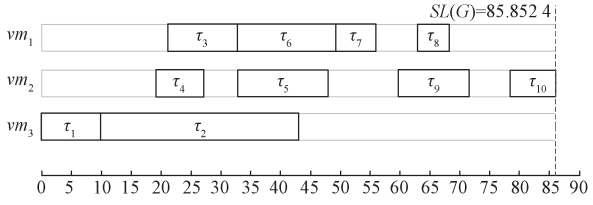

图 2 BLMSL 算法的调度效果图

Fig. 2 Scheduling effect diagram of BLMSL algorithm

4 实验结果及分析

本节进行了多组实验来对比观察 BLMSL 算法相较于其它先进算法的性能。首先介绍实验设置，然后给出实验结果及分析。

4.1 实验设置

实验中工作站配置了 Intel Core(TM) i5-9300H@2.40GHz 八核处理器，16 GB 内存，系统运行版本为 Windows10，采用 Python 作为实验的编程语言。模拟的 HCS 具有 64 个计算能力和单价各异的 *VM*。*VM* 的主要参数如下：10 ms $\leqslant w_{i,k} \leqslant$ 100 ms，10 ms $\leqslant c_{i,j} \leqslant$ 100 ms，0.03 $\leqslant P_{k,\text{ind}} \leqslant$ 0.07，2.5 $\leqslant m_k \leqslant$ 3.0，$\phi_{k,\max}$=1.0 GHz。所有的频率均为离散值，精度为 0.01 GHz。

HEFT 算法是异构系统上以减少调度长度为目标的经典算法，MSLECC 算法与本文的工作均为在考虑能耗约束的前提下最小化调度长度，因此选择这两种算法作为实验中的对比算法。这 3 种算法的研究目标均是最小化工作流的调度长度。

本文的实验使用实际的能耗 $E(G)$ 和最终的调度长度 $SL(G)$ 作为算法性能的评价指标，使用高性能计算领域中广泛使用的真实应用并行任务集进行测试，通过改变应用规模的大小和能耗约束的大小来更直观地说明 BLMSL 算法的有效性。

4.2 Epigenomics 工作流应用

本节选择生物遗传学领域的表观基因组学 Epigenomics 作为测试的工作流结构，下文中简称 EP，其科学工作流应用程序的结构如图 3 所示。

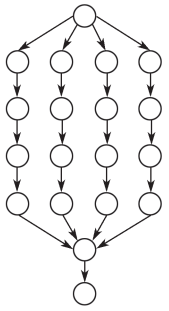

图 3 Epigenomics 工作流结构图

Fig. 3 Epigenomics workflow structure diagram

实验一。本组实验工作流的能耗约束固定为 0.5（即 $E_{\text{given}}(G)=E_{\max}(G)\times 0.5$），将 EP 应用的规模分别设置为 19, 51, 103, 523, 1 003，随后比较在不同任务数量时不同算法的实际能量消耗和调度长度。MSLECC 算法和 BLMSL 算法在所有的实验案例中都能在满足能耗约束的前提下成功进行调度，而由于 HEFT 算法不考虑功耗约束，虽然实现了较小的调度长度，但其能耗未能满足能量约束条件。

表 4 展示了 3 种算法在不同任务数量下的实际能耗和调度长度。随着工作流应用程序规模扩大，能耗和调度长度随之增大。相比 HEFT 和 MSLECC 算法，BLMSL 能够保证在满足能耗约束的条件下，获得较小的调度长度，在应用规模为 N=1 003 且消耗能量基本相同的情况下，BLMSL 算法相比 MSLECC 算法调度长度减少了约 63.22%，体现了其在调度长度最小化方面的巨大优越性。

图 4 展示了 3 种算法在不同任务数量的 EP 应用下的调度长度结果，发现 BLMSL 算法在满足能耗约束条件下，实现了较其他算法更小的调度长度。

表 4　不同应用规模下的 EP 工作流调度结果

Table 4　EP workflow scheduling results under different application scales

$\|N\|$	$E_{bdgt}(G)$	HEFT		MSLECC		BLMSL	
		$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
19	147.41	182.46	118	147.41	259.68	144.48	128.42
51	377.52	461.38	236	377.52	667.65	377.42	265.98
103	784.70	1 058.22	469	784.69	1 384.65	793.07	529.79
523	4 006.83	5 202.69	2 256	4 006.83	6 668.60	4 006.00	2 605.50
1 003	7 461.20	9 799.87	4 238	7 461.20	13 624.05	7 460.90	5 011.47

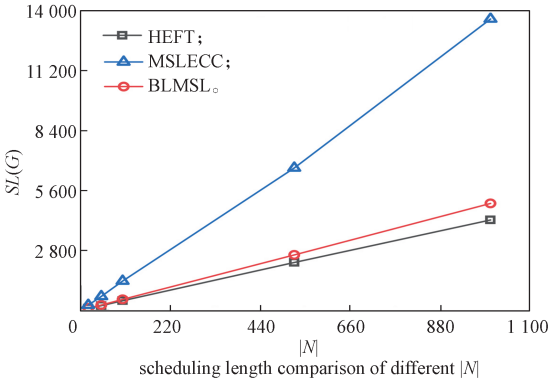

图 4　不同任务数的 *SL* 对比

Fig. 4　*SL* comparison of with different task numbers

4.3　Inspiral 工作流应用

本组实验采用重力物理学领域的 LIGO 科学工作流，其结构如图 5 所示。

实验二。将实验的应用规模固定为 94，能耗约束值 $E_{given}(G)$ 计算为 $E_{min}(G)\times\lambda$，λ 的取值从 2 到 4，取值间隔为 0.5。

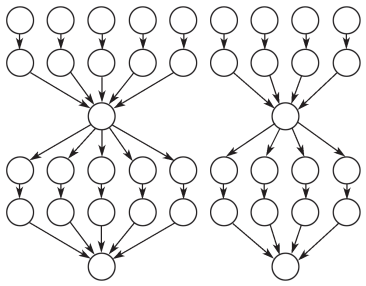

图 5　Inspiral 工作流结构图

Fig. 5　Inspiral workflow structure digram

详细调度结果见表 5，当给定能耗较宽松时，工作流中任务能利用更宽松能量实现更小调度长度，λ=4.0，BLMSL 算法在满足能耗约束条件下，实际调度长度和 HEFT 算法仅相差 3.72%，而能量消耗却比 HEFT 算法节省 13.47%。当并行应用任务集给定能耗较小时，BLMSL 算法的优越性更突出，如 λ=2 时，BLMSL 算法实际调度长度比 MSLECC 算法提升了 76.49%。

表 5　不同能耗约束条件下的 LIGO 工作流调度结果

Table 5　LIGO workflow scheduling results under different energy consumption constraints

λ	$E_{bdgt}(G)$	HEFT		MSLECC		BLMSL	
		$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
2.0	401.02	1 060.81	388	401.02	2 388.93	400.98	561.74
2.5	501.27	1 057.22	386	501.27	1 917.61	467.98	501.27
3.0	601.52	967.30	369	601.52	1 589.40	601.45	431.60
3.5	701.78	908.23	346	701.78	1 242.51	701.74	357.85
4.0	802.03	925.91	339	802.03	683.44	801.20	326.38

图 6 显示了不同能耗约束的 LIGO 应用下 3 种算法的调度长度结果。从图中可以观察出，BLMSL 算法在给定能耗条件变化较大的环境下依然能保持调度长度的相对稳定。且在不同能耗约束条件下，BLMSL 算法的最终调度长度值始终小于 MSLECC 算法的调度长度值。

由以上结果可看出，BLMSL 算法能够同时兼顾工作流的能耗和调度长度两方面，实现性能平衡。通过观察在 LIGO 和 EP 并行应用科学工作流上进行的诸多实验，充分说明 BLMSL 算法在两个科学工作流上的实验中始终保持着优越的性能。

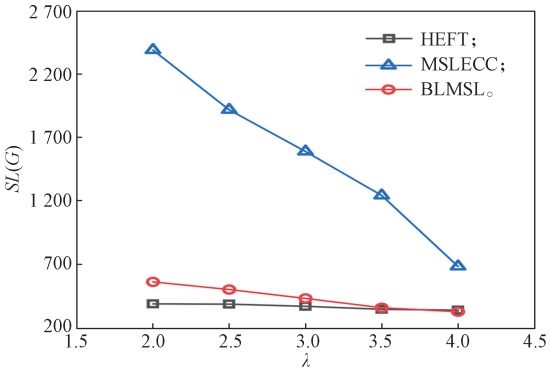

图 6　不同能耗约束 LIGO 的 *SL* 对比

Fig. 6　*SL* comparison of LIGO with different energy consumption constraints

5 结语

针对 HCS 中能量受限并行应用的调度长度最小化问题，本文设计了 *BL* 等级预分配机制，并基于 *BL* 等级预分配策略提出了 BLMSL 算法来实现算法的调度长度最小化，提升了能耗预分配阶段的合理性和科学性，避免了因对任务优先级敏感而影响调度长度问题。在多个科学工作流上进行实验的结果表明，BLMSL 算法能在能耗约束的条件下实现更小的调度长度。未来，课题组将致力于探索云环境中工作流集在预算约束下的高可靠性研究方法。

参考文献：

[1] 孙德洋，娄嘉鹏，李建鹏，等 . 异构云环境下的密码服务调度方法 [J]. 计算机应用与软件，2019，36(6)：302−307，333.
SUN Deyang，LOU Jiapeng，LI Jianpeng，et al. Cryptographic Service Scheduling Method in Heterogeneous Cloud Environment[J]. Computer Applications and Software，2019，36(6)：302−307，333.

[2] 王 凌，吴楚格，范文慧 . 边缘计算资源分配与任务调度优化综述 [J]. 系统仿真学报，2021，33(3)：509−520.
WANG Ling，WU Chuge，FAN Wenhui. A Survey of Edge Computing Resource Allocation and Task Scheduling Optimization[J]. Journal of System Simulation，2021，33(3)：509−520.

[3] YUAN H T，BI J，ZHOU M C，et al. Biobjective Task Scheduling for Distributed Green Data Centers[J]. IEEE Transactions on Automation Science and Engineering，2021，18(2)：731−742.

[4] WANG X J，NING Z L，GUO S，et al. Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing[J]. IEEE Transactions on Mobile Computing，2022，21(2)：598−611.

[5] GARG R，MITTAL M，SON L H. Reliability and Energy Efficient Workflow Scheduling in Cloud Environment[J]. Cluster Computing，2019，22(4)：1283−1297.

[6] CHENG D Z，ZHOU X B，LAMA P，et al. Energy Efficiency Aware Task Assignment with DVFS in Heterogeneous Hadoop Clusters[J]. IEEE Transactions on Parallel and Distributed Systems，2018，29(1)：70−82.

[7] UL ISLAM F M M，LIN M，YANG L T，et al. Task Aware Hybrid DVFS for Multi-Core Real-Time Systems Using Machine Learning[J]. Information Sciences，2018，433/434：315−332.

[8] TANG Z，QI L，CHENG Z Z，et al. An Energy-Efficient Task Scheduling Algorithm in DVFS-Enabled Cloud Environment[J]. Journal of Grid Computing，2016，14(1)：55−74.

[9] SONG J L，XIE G Q，LI R F，et al. An Efficient Scheduling Algorithm for Energy Consumption Constrained Parallel Applications on Heterogeneous Distributed Systems[C]//2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). Guangzhou：IEEE，2018：32−39.

[10] 王 兰，张龙信，满君丰，等 . 异构计算环境下基于优先队列划分的调度算法 [J]. 小型微型计算机系统，2020，41(2)：303−309.
WANG Lan，ZHANG Longxin，MAN Junfeng，et al. Scheduling Algorithm Based on Priority Queue Dividing in Heterogeneous Computing Environment[J]. Journal of Chinese Computer Systems，2020，41(2)：303−309.

[11] LI K Q. Optimal Task Execution Speed Setting and Lower Bound for Delay and Energy Minimization[J]. Journal of Parallel and Distributed Computing，2019，123：13−25.

[12] MAURYA A K，MODI K，KUMAR V，et al. Energy-Aware Scheduling Using Slack Reclamation for Cluster Systems[J]. Cluster Computing，2020，23(2)：911−923.

[13] TOPCUOGLU H，HARIRI S，WU M Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing[J]. IEEE Transactions on Parallel and Distributed Systems，2002，13(3)：260−274.

[14] SULAIMAN M，HALIM Z，LEBBAH M，et al. An Evolutionary Computing-Based Efficient Hybrid Task Scheduling Approach for Heterogeneous Computing Environment[J]. Journal of Grid Computing，2021，19(1)：1−31.

[15] XIAO X R，XIE G Q，LI R F，et al. Minimizing Schedule Length of Energy Consumption Constrained Parallel Applications on Heterogeneous Distributed Systems[C]//2016 IEEE Trustcom/BigDataSE/ISPA. Tianjin：IEEE，2017：1471−1476.

（责任编辑：申 剑）