Journal of Computer Applications
计算机应用,2013,33(9):2410 - 2415
ISSN 1001-9081
CODEN JYIIDU
2013-09-01
http://www.joca.cn

文章编号:1001-9081(2013)09-2410-06
doi:10.11772/j.issn.1001-9081.2013.09.2410

# 云计算环境下多有向无环图工作流的节能调度算法

刘丹琦[1*],于　炯[2],英昌甜[1]
(1.新疆大学 信息科学与工程学院,乌鲁木齐 834000;　2.新疆大学 软件学院,乌鲁木齐 834000)
(*通信作者电子邮箱 liudq@xju.edu.cn)

**摘　要**:针对多有向无环图(DAG)工作流节能调度算法中存在的节能效果不佳、适用范围较窄和无法兼顾性能优化等问题,提出了一种新的多 DAG 工作流节能调度方法——MREO。MREO 在对计算密集型和通信密集型任务特点进行分析的基础上,通过整合独立任务,减少了处理器的数量,并利用回溯和分支限界算法对任务整合路径进行动态的优化选择,有效降低了整合算法的复杂度。实验结果证明,MREO 在保证多 DAG 工作流性能的前提下,能够有效降低系统的计算和通信能量开销,获得了良好的节能效果。

**关键词**:多有向无环图;整合;节能调度;能耗
**中图分类号**:TP393　　**文献标志码**:A

# Energy efficient scheduling for multiple directed acyclic graph in cloud computing

LIU Danqi[1*], YU Jiong[2], YING Changtian[1]
(1. *School of Information Science and Engineering, Xinjiang University, Urumqi Xinjiang* 834000, *China*;
2. *School of Software, Xinjiang University, Urumqi Xinjiang* 834000, *China*)

**Abstract:** Energy-efficient scheduling algorithms based on multiple Directed Acyclic Graph (DAG) fail to save energy efficiently, have a narrow application scope and cannot take performance optimization into account. In order to solve these problems, Multiple Relation Energy Optimizing (MREO) was proposed for multiple DAG workflows. MREO integrated independent tasks to reduce the number of processors used, on the basis of analyzing the characteristics of computation-intensive and communication-intensive tasks. Backtracking and branch-and-bound algorithm were employed to select the best integration path dynamically and reduce the complexity of the algorithm at the same time. The experimental results demonstrate that MREO can reduce the computation and communication energy cost efficiently and get a good energy saving effect on the premise of guaranteeing the performance of multiple DAG workflows.

**Key words:** multiple Directed Acyclic Graph (DAG); integration; Energy-Efficient Scheduling (EES); energy consumption



## 0　引言

云计算是近几年兴起的一种网络应用模式,使用者可以按需动态地改变访问的资源和服务的种类及数量[1]。随着众多行业和应用对大规模计算能力及海量数据处理能力不断增长的需求,越来越多的企业选择将工作负载传输到云端处理,因此大规模的云计算数据中心应运而生。然而,由大规模的云计算数据中心带来的能耗也快速增长。例如:2011 年美国的数据中心全年的耗能大概为 1 000 亿 kWh,相当于 74 亿美元[2]。云计算数据中心的大量能耗带来的不仅是高额的资金消耗,同时也增加了二氧化碳的排放,对环境造成了一定影响。因此,如何构建绿色的云计算数据中心已成为研究人员关注的热点问题。

云计算数据中心节能调度问题中针对并行任务节能调度的研究日益受到众多研究者的关注。现有的针对并行任务调度的能耗优化机制,其主要目标是尽量减少占用资源数目和占用时间,提高资源利用率,以达到整个并行任务集群系统性能和能耗之间的平衡。其调度基础是将所有任务按照依赖关系图的拓扑结构分成多条路径,每个路径上的任务分为一组,分配到一个处理器上执行。而针对于分配后负载轻的处理器,如何合理地在不影响其他处理器工作的前提下将其合并,以节省能耗也日渐成为研究的重点。

由于 CPU 在 Idle 状态下产生的能耗大约为满负载时能耗的 70%,因此如何利用并行任务间的松弛时间(即任务最早开始时间和最晚结束时间的处理器时间空隙,在这段时间内 CPU 处于 Idle 状态)也成为对于单台处理器能耗优化的研究重点。文献[3]中提到 CPU 运行频率在长时间内保持一致的情况下能耗较低,可以通过动态电压频率调整(Dynamic Voltage and Frequency Scaling,DVFS)机制降低在松弛时间内 CPU 的运行频率以达到节省能耗的目的。节能调度(Energy-Efficient Scheduling, EES)算法通过 DVFS 机制,在任务容忍限度下,合理地调整 CPU 运行频率,降低了处理器能耗开销[4-5]。对于通信密集型任务,由于传输能耗在总能耗中的比重较大,基于 DVFS 机制的 EES 算法无法获得良好的节能效果[3]。除了 DVFS 机制外,在处理器松弛时间内,通过对先驱任务的复制执行,可以减少任务间的传输能耗、缩短任务的总体执行时间,达到整体能耗优化的效果[6-7],能量感知的处理器合并优化(Energy Aware Duplication-Processor Reduction

**收稿日期**:2013-03-12;**修回日期**:2013-04-26。
**基金项目**:国家自然科学基金资助项目(61063042,61262088);新疆维吾尔自治区自然科学基金资助项目(2011211A011)。
**作者简介**:刘丹琦(1988 - ),女,新疆克拉玛依人,硕士研究生,主要研究方向:云计算、绿色计算;　于炯(1964 - ),男,北京人,教授,博士生导师,博士,主要研究方向:分布式与网格计算、网络安全;　英昌甜(1989 - ),女,新疆乌鲁木齐人,博士研究生,主要研究方向:数据库、网格调度算法。

Optimizing,EAD-PRO)算法通过基于能量感知的任务复制方法,启发式地对处理器进行合并,从而实现节省系统能耗的目的[8]。

然而大多数的研究只考虑一种任务依赖关系,但在实际情况中,一个处理器集群中存在多种任务依赖关系,若用有向无环图(Directed Acyclic Graph, DAG)来表示,则意味着在整个数据中心中存在多种 DAG 工作流。

本文在以上研究基础上,设计了一种针对多 DAG 工作流的能耗优化方法——MREO (Multiple Relation Energy Optimizing)。该方法通过合理利用任务间的松弛时间,减少服务器使用数目和传输能耗来降低系统总能量,减少服务器使用数目和传输能耗来降低系统总能量开销。对于合并后残留的松弛时间从通信密集型和计算密集型两种不同的任务类型出发,采用最优的处理方式再对局部能耗进行优化。该方法可以分为两大模块:1)任务合并模块。依照任务的依赖关系确定如何合并处理器,减少处理器使用数目和传输能耗。2)残留松弛时间优化模块。针对合并后残留的松弛时间采用最优的处理方式。

与已有的研究工作相比,本文的创新之处在于:

1)考虑了多 DAG 工作流的合并,以降低资源数目和处理器占用时间,提高资源的利用率,降低总体能量消耗。

2)对于合并后残留的松弛时间,针对任务类型的不同,采用最优的处理方式,从局部最优的角度出发,再次对系统整体能耗进行优化。

## 1　相关工作

### 1.1　DAG 工作流调度策略

由于 DAG 工作流调度具有 NP 难任务的特点,大量的启发式调度算法应运而生,它们大致可以被分为以下几类:1)基于优先级的调度;2)基于聚类的调度;3)基于复制的调度。

基于优先级的调度方法为每一个任务分配一个优先等级,根据优先等级将任务分配到处理器上执行。基于聚类的调度方法将有通信情况的任务划分到一个分组,然后按照分组指派处理机,从而减少处理机间的通信开销。基于复制的调度方法利用处理器的空闲时间,复制执行前驱任务,从而降低前驱任务的通信开销,缩短系统整体的运行时间。文献[9]对不同的 DAG 工作流调度算法进行了对比研究,其中,异构最早完成时间(Heterogeneous Earliest-Finish-Time, HEFT)算法是线性规划启发式算法的典型代表。文献[10]通过对 HEFT 算法的分析研究,设计了性能和公平性更好的调度算法。文献[11]针对到达时间不同的多类型 DAG 工作流,提出了一种兼顾调度需求和资源利用率的混合调度策略。

### 1.2　基于复制的任务调度策略

前驱任务的复制能够减少后继任务的等待时间和调度长度,从而实现系统整体能耗的优化。任务复制调度(Task Duplication Scheduling,TDS)[6]通过复制关键路径上的前驱任务和避免前驱任务的数据通信延迟,提前关键路径上的后继任务的开始时间。EAD(Energy-Aware Duplication)调度[7]将执行前驱任务的能耗与复制后节省的网络传输能耗之差作为参数,若该参数小于给定阈值则复制任务。PEBD(Performance-Energy Balance Duplication)调度[7]在性能和能耗节省之间做出了平衡,用执行前驱任务的能耗减去节省的网络传输能耗,其差值再除以缩短的等待时间,将最终的比值作为参数,只有该参数小于或等于所设定的阈值时,才复制前驱任务。处理器合并优化(Processor Reduction Optimizing, PRO)算法[8]是一种启发式处理器合并的优化方法,该方法按照任务最早开始时间和最早结束时间查找处理器时间空隙,将轻负载处理器上的任务重新分配到其他处理器上,从而减少处理器的使用数目,降低系统总体能耗。

### 1.3　DVFS 高能效策略

近几年有大量的基于 DVFS 机制处的能耗优化任务调度策略。这些方法通过降低 CPU 内部的电压或频率,减小 CPU 芯片的功率,从而降低整个系统的能量开销。文献[4-5,12]中提出了一些对于独立任务,单个处理机系统和实时系统的 DVFS 节能策略。Kimura 等[13]提出了一种通过动态调整 CPU 频率,有效延长任务执行时间从而回收松弛时间,最终达到节能的调度策略。Huang 等[3]提出了一种针对并行任务的能量有效利用的调度策略,该方法通过收集非关键路径上的松弛时间并重新分配任务来达到节能的目的。

本文算法从能耗优化的角度考虑如何合理地合并多 DAG 工作流中无依赖关系的路径,并通过 DVFS 和复制调度两种策略的结合,针对通信密集型和计算密集型两种不同的任务类型,有效地处理合并后残留的无效松弛时间。同时,在通信密集型应用中,由于网络通信的能耗可能在总能耗中所占比重较大。所以,本文的能耗优化策略也引入了对传输能耗的考虑,从而达到总体能耗的优化。

## 2　节能调度模型

本文的节能调度模型由系统模型、任务模型和能量模型组成。

### 2.1　系统模型

由于本文的调度方法针对多 DAG 工作流的能耗优化,因此所有的处理器都可以运行在不同的电压和频率下。本文用 $P_i$ 表示处理器,用 $V_i$ 表示处理器的供电电压,用 $f_i$ 表示处理器的运行频率。其他的系统参数如表 1 所示。

**表 1　系统模型参数**

| 函数 | 含　义 |
|---|---|
| $D$ | 所有 DAG 的集合 |
| $m$ | 集合 $D$ 中 DAG 的总数 |
| $n_{ij}$ | 第 $i$ 个 DAG 中的第 $j$ 个任务 |
| $n_i$ | 第 $i$ 个 DAG 中的任务总数 |
| $e_{ijk}$ | 在第 $i$ 个 DAG 中任务 $j$ 与任务 $k$ 之间的约束关系 |
| $AST(n_{ij})$ | 任务的最早开始时间 |
| $AFT(n_{ij})$ | 任务的最早完成时间 |
| $AST'(n_{ij})$ | 任务的最晚开始时间 |
| $AFT'(n_{ij})$ | 任务的最晚结束时间 |
| $T_{exec}(n_{ij})$ | 任务的执行时间 |
| $Slack(n_{ij})$ | 任务的松弛时间 |

### 2.2　任务模型

对于 $n$ 个具有依赖关系的并行任务用一个 DAG 来表示其依赖关系,多个 DAG 之间不存在依赖关系,相互独立。

**定义**　一个 DAG 图 $G$,$G=(N, E)$,其中顶点集合 $N$ 用来表示要被处理的任务集合,$N=\{n_{ij} \mid 1\leqslant i\leqslant m,\ 1\leqslant j\leqslant n_i\}$;有向边集合 $E$ 代表任务之间的依赖关系,$E=\{e_{ijk} \mid 1\leqslant i\leqslant m,\ 1\leqslant j\leqslant n_i,\ 1\leqslant k\leqslant n_i\}$。

每一个 $e_{ijk}\in E$ 都有一定权值,用 $c_{ijk}$ 表示,代表着任务 $n_{ij}$ 传输到任务 $n_{ik}$ 所需的网络耗时:如果任务 $n_{ij}$ 和任务 $n_{ik}$ 不在

同一台计算机上执行，则需要一定的网络耗时将任务 $n_{ij}$ 的运行结果传输给任务 $n_{ik}$，此时 $c_{ijk}\neq 0$；如果这两个任务在一台计算机上执行，则网络传输耗时为 0，此时 $c_{ijk}=0$。如图 1 所示，如果任务 3 和任务 5 不在同一台计算机上运行则需要 3 个单位的时间才能将任务 3 的相关结果传输给任务 5。

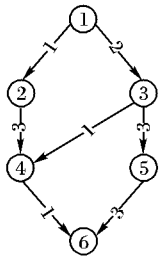


图 1 DAG 示例

### 2.3 能量模型

本文从计算密集型和通信密集型两方面出发，因此统计 CPU 和网卡两部分消耗的能耗作为系统的总能量消耗。

#### 2.3.1 CPU 能耗

CPU 是云计算数据中心中最耗能的部分[14]，其主要由动态能耗 $E_{\text{dynamic}}$ 和静态能耗 $E_{\text{statoc}}$ 组成。其中动态能耗 $E_{\text{dynamic}}$ 是 CPU 总能耗的主要部分[15-16]，同时动态能耗 $E_{\text{dynamic}}$ 中的功率消耗又可以通过 DVFS 机制来降低。

$P_{\text{dynamic}}$ 和供电电压 $V_i$，CPU 频率 $f_i$ 的关系如下所示

$$P_{\text{dynamic}} = k * V_i^2 * f_i \tag{1}$$

其中 $k$ 是一个与动态功率相关的参数。因此，动态能耗 $E_{\text{dynamic}}$ 可表示为：

$$E_{\text{dynamic}} = P_{\text{dynamic}} * \Delta t \tag{2}$$

其中 $\Delta t$ 代表动态功率的持续时间。当处理器处于 Idle 状态时，它的频率并没有降到 0，而是降到了处理器的最低运行频率，同时供电电压也只是降低到最低电压，而不是 0。因此对于总能量消耗 $E_C$ 而言，不仅要考虑任务运行时的能量消耗，也要考虑 CPU 在 Idle 状态下的能量消耗。

$$E_C = E_{\text{dynamic}} + E_{\text{static}} \tag{3}$$

#### 2.3.2 网卡能耗

假设每个节点上只有一块网卡，$PN_{\text{busy}}$ 表示网卡忙碌（收发数据）时的功率，$PN_{\text{idle}}$ 表示网卡空闲时的功率，第 $j$ 个节点中网卡接受数据耗费的能量可以表示为：

$$EN_{\text{busy}}^{j} = PN_{\text{dynamic}} * \sum_{i=1}^{n}\left(x_{ij} * \sum_{v_k\in\Omega}(1-x_{kj}) * c_{ijk}\right) \tag{4}$$

假设发送和接受同一批数据耗时相等，所有网卡忙碌（发送和接受）时间总能耗为：

$$EN_{\text{busy}} = 2 * PN_{\text{dynamic}} * \sum_{i=1}^{n}\left(x_{ij} * \sum_{v_k\in\Omega}(1-x_{kj}) * c_{ijk}\right) \tag{5}$$

则网卡的总能耗为：

$$EN = m * PN_{\text{idle}} * L_{\max} + 2 * (PN_{\text{busy}} - PN_{\text{idle}}) * \sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij} * \sum_{v_k\in\Omega}(1-x_{kj}) * c_{ijk}) \tag{6}$$

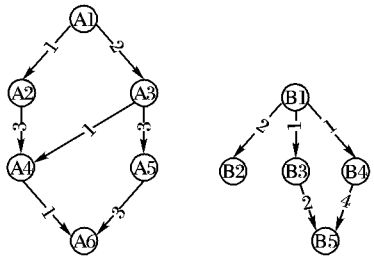
则总能耗为：

$$E_{\text{total}} = E_C + EN \tag{7}$$

## 3 调度算法

### 3.1 多 DAG 合并调度

本文将通过如下方法对多 DAG 工作流进行整合，如图 2 所示。利用 HEFT 算法分别求出 DAG(A) 和 DAG(B) 的最早完成时间，约定较长的时间为全局的最晚完成时间 makespan。




(a) DAG(A) (b) DAG(B)

图 2 两种不同的 DAG 工作流

图 2 中两种不同 DAG 工作流的各个系统参数如表 2 所示。表格中的时间单位均为一个单位时间。松弛时间 $Slack(n_{ij})$ 的计算公式如式(8)所示：

$$Slack(n_{ij}) = AFT'(n_{ij}) - AST(n_{ij}) - T_{\text{excc}}(n_{ij}) \tag{8}$$

表 2 图 2 系统参数

| 任务 | $T_{\text{excc}}(n_{ij})$ | $AST(n_{ij})$ | $AFT(n_{ij})$ | $AST'(n_{ij})$ | $AFT'(n_{ij})$ | $Slack(n_{ij})$ |
|---|---|---|---|---|---|---|
| A1 | 1 | 0 | 1 | 0 | 1 | 0 |
| A2 | 1 | 2 | 3 | 4 | 5 | 3 |
| A3 | 2 | 1 | 3 | 1 | 3 | 0 |
| A4 | 1 | 4 | 5 | 5 | 6 | 1 |
| A5 | 4 | 3 | 7 | 3 | 7 | 0 |
| A6 | 1 | 7 | 8 | 7 | 8 | 0 |
| B2 | 1 | 3 | 4 | 6 | 7 | 3 |
| B3 | 2 | 2 | 4 | 2 | 4 | 0 |
| B4 | 4 | 1 | 5 | 2 | 6 | 10 |
| B5 | 1 | 6 | 7 | 6 | 7 | |

对于多种 DAG 集合，利用 HEFT 算法得到的关键路径单独使用一台处理器，对于不含关键路径的其他 DAG 使每个 DAG 中传输代价最大的路径单独占用一台处理器。图 2 中处理器的分配情况如图 3 所示。

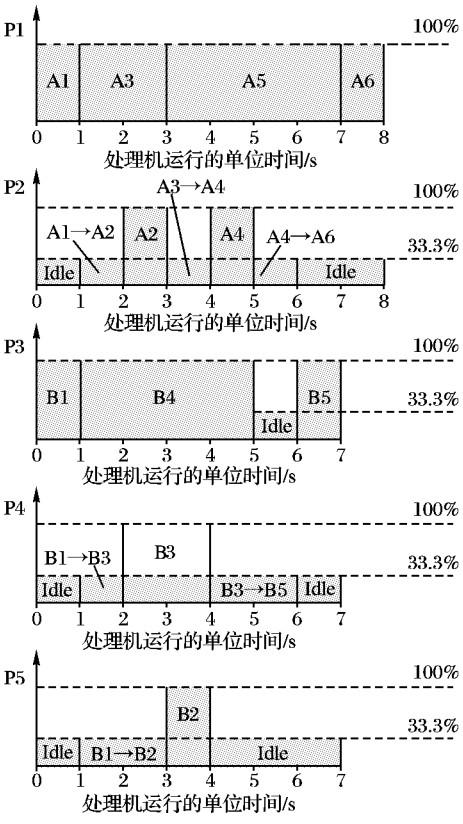



图 3 处理机分配

合并调度采用分支限界与回溯结合的方法,每一步将没有松弛时间和松弛时间不符合合并的处理器分支减去不考虑,利用回溯算法算出每一种合并方式的能耗值,选择最优的能耗值,以其合并路径作为最终的合并路径。合并后的结果如图 4 所示。

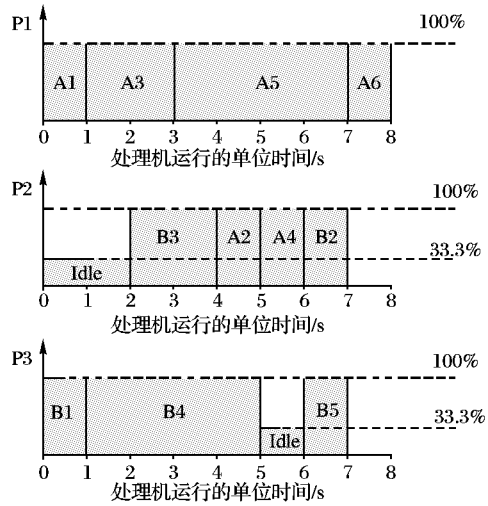



图 4　合并处理后的处理机状态

**算法 1**

```
D = {DAG};                                    //输入多 DAG 的集合 D
makespan = Cmakespan(D);
//makespan = HEFT 算法计算所有 DAG 的最长完成时间
Re = D 中各个 DAG 的每条路径分配一个处理机的结果集合
Ge = Prio(Re);
//确定优先级,作业数量少的处理机优先级高;
//作业数量相同时,处理时间短的优先级高
S′用来保存过程中产生的全部结果集合
MREO(Ge)
{
  将优先级最高的处理机 Ge0 作为树根
  遍历剩余处理机集合 Ge′
  {
    if( ismerge ( root + Ge′) = = true&&time ( root + Ge′) < =
    makepan)                                  //可以合并
    {
      J = {root && Gei 中的所有作业};
      job(J);
      //根据各任务的处理时间,最早开始时间,
      //早晚完成时间,确定处理顺序
      RT = {root + Ge′i} + {Ge′ - {Ge′i}};
      //将合并后的结果放入临时集合 RT 中
      Prio(RT);                               //对 RT 集合排序
      MREO(RT);                               //深度搜索
    }
    else                                      //分支限界,回溯
    {
      if(i >= length(Ge′))                    //无法继续深度搜索
      {
        回溯一层,选择其他孩子节点,继续搜索
        if(RT in S′)                          //重复结果
          return;
        else
        { 保存新的结果集合 RT,求得能耗结果
          return;
  }}}}
  OEnergy(S);
}
```

### 3.2　残留松弛时间的处理

对于合并后残留的松弛时间,本文选用两种处理方式:DVFS 机制和能量感知的任务复制机制。

**算法 2**

```
OEnergy(T)
{
  if(slacking(T) == ture)
  //检查当前合并结果中是否存在无利用的松弛时间
  {
  for(所有满足复制条件的任务)
  { 利用 EAD-PRO 算法复制任务,并保留复制后能耗最少的
    复制结果 T1;
    CEnegy(T1);}
  利用 EES 算法处理 T,将处理结果保留入 T2;
  if(CEnegy(T1) <= CEnegy(T2))
    T = T1;
  else
    T = T2;}
  return T;
}
```

## 4　实验评估

为了评价和分析本文算法的性能,实验通过模拟程序随

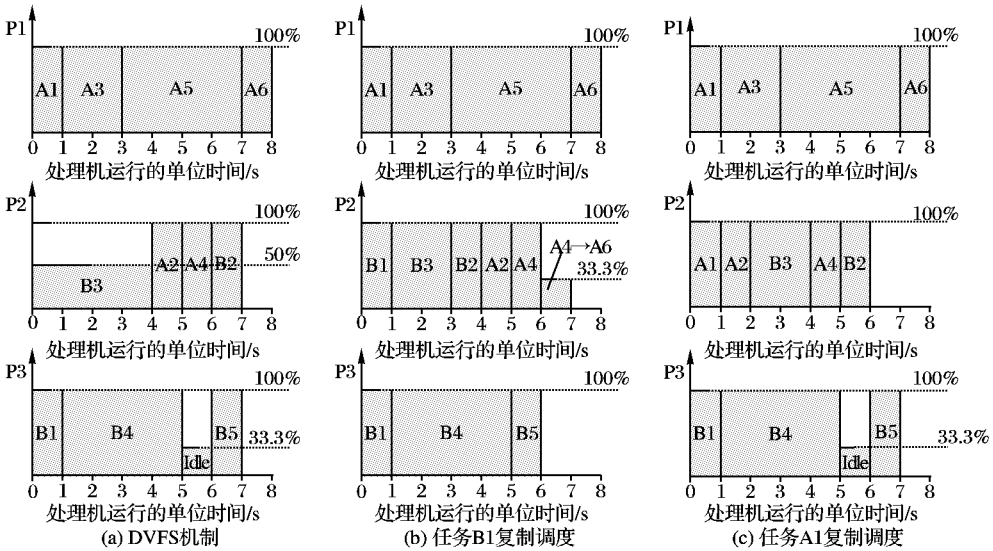



图 5　松弛时间处理后的处理机情况

机生成实验所需多种 DAG。为了便于计算,DAG 中任务的执行时间与传输时间均取整数并限制于[1,10]的闭区间。利用 CloudSim 对 EES[3]、EAD-PRO[8]和 MREO 算法进行云计算环境下的仿真实验并且进行了相应的对比。在实验过程中,通过修改模拟程序的相关参数,分别以计算密集型和通信密集型两种任务集合为实验输入,对各个算法的节能效果进行对比。同时构建了以 AMD Torion MT-34 为处理器和瑞昱 RTL8105E 为网卡的模拟同构集群的实验环境。本实验采用千兆以太网作为网络环境,在这一网络类型下网卡的功率为 5 W。处理器电压和运行频率的关系如表 3 所示。

表 3 电压频率关系

| 电压/V | 频率/MHz | 电压/V | 频率/MHz |
|---|---|---|---|
| 1.50 | 2000 | 1.10 | 1200 |
| 1.40 | 1800 | 1.00 | 1000 |
| 1.30 | 1600 | 0.90 | 800 |
| 1.20 | 1400 | | |

### 4.1 实验 1:处理器利用数目对比

首先通过模拟程序随机生成一张 DAG,分配完处理器后,记录下所占用的处理器数目;然后再随机生成一张 DAG,对于现有的两张 DAG 采用合并与非合并算法分别计算处理器的利用数目;最后记录。以此类推,之后的每次实验都在原有 DAG 的基础上加入一张新生成的 DAG。利用 MREO 算法作为合并算法,对合并前与合并后利用处理机的数目进行分析比较。

从图 6 可看出:随着 DAG 数的增加,利用 MREO 算法合并后的处理器的利用数目大大减少。当 DAG 数量很少时,差别并不明显;但当 DAG 达到一定数量时,使用 MREO 算法优化后利用处理器数目的结果大约降低到无 MREO 算法的一半。由此,可得出 MREO 算法有效地合并多 DAG 工作流中的轻负载处理器,从而减少处理器的使用数量。

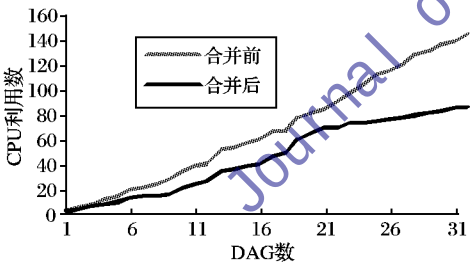



图 6 CPU 利用数与 DAG 数关系

### 4.2 实验 2:MREO 算法与其他算法总执行时间对比

在实验 1 计算处理器利用数目的基础上,针对实验 1 中生成的 DAG 利用不同的节能算法计算其总执行时间(总执行时间为所有处理器完成时间的总和)。

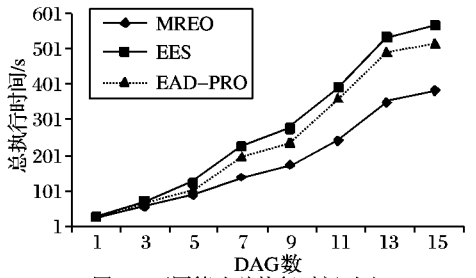



图 7 不同算法总执行时间对比

由图 7 可看出:随着 DAG 数量的增加,总执行时间也在增加。由于 EES 只考虑 DVFS 机制,EAD-PRO 只考虑了有依赖关系的处理器的任务合并,而 MREO 算法针对多种 DAG 集合中轻负载处理器的合并,当 DAG 的数量少时几种算法的总执行时间基本相同,但数量达到一定数值时,MREO 算法的总执行时间明显低于其他两种算法的总执行时间。

### 4.3 实验 3:MREO 算法与其他节能算法资源利用率对比

基于实验 1 与实验 2,利用式(9)计算出在不同不同 DAG 数下各个算法对处理器的平均利用率。

$$\text{平均资源利用率} = \frac{\sum_{\text{CPU数}}\sum_{\text{时间}}\text{CPU 利用率}}{\text{总完成时间}} \times 100\% \qquad (9)$$

由图 8 可知:随着 DAG 数的增加,MREO 算法的平均资源利用率呈上升趋势,明显优于其他两个算法。由此证明 MREO 算法有效地整合了无关 DAG 工作流中存在的轻负载处理器,从而大大提高了资源的利用率。

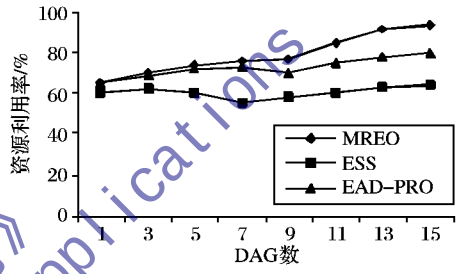



图 8 平均资源利用率对比

### 4.4 实验 4:MREO 算法与其他节能算法平均执行时间对比

实验生成大量的随机任务,在实验过程中记录下生成的任务数,并且参照实验 2 的计算方法统计出相应的总执行时间:

$$\text{平均执行时间} = \text{总执行时间} \div \text{任务总数} \qquad (10)$$

利用式(10)计算出每一个任务的平均执行时间。不同算法下任务的平均执行时间的分析对比如图 9 所示。

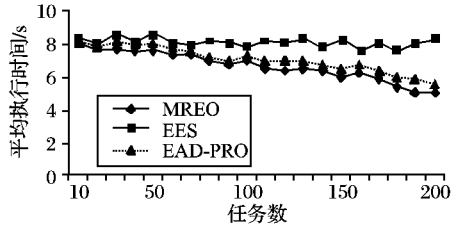



图 9 不同算法任务平均执行时间对比

由图 9 可知:当任务数量少时,三个算法的任务平均执行时间没有出现明显差别。但当任务数超过 70 后,EES 算法的平均执行时间明显高于其他两个算法。这是由于 EES 算法只考虑了能耗的优化,却没有兼顾考虑性能的优化。虽然 EAD-PRO 与 MREO 算法的平均执行时间都呈下降趋势,但其优化效果并不如 MREO 算法明显。

### 4.5 实验 5:MREO 算法与其他算法节能效果对比

实验生成大量的随机任务,在预先设定的模拟同构集群环境下,结合式(1)~(7)计算不同算法下的能耗情况,并对节能效果加以分析比较。

$$\text{节能效果} = \frac{\text{算法下总能耗}}{\text{传统方法下总能耗}} * 100\% \qquad (11)$$

由图 10 可知:当任务数到达一定数量时,MREO 算法的节能效果优于其他两种算法。并且,任务数量越大,MREO 算法节能的优越性越凸显。

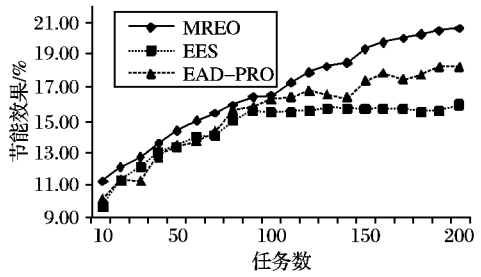
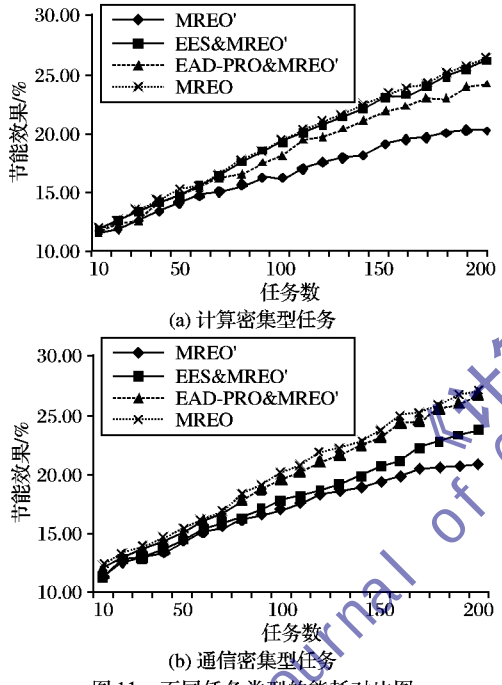



图 10　不同算法节能效果比较

### 4.6　实验 6:不同任务类型的节能效果对比

实验利用模拟程序中参数来调整平均出度,将任务划分为计算密集型任务与通信密集型任务[7]。针对不同任务类型的集群采用不同的优化算法,进行分析与比较。对于 EES 和 EAD-PRO 算法,本文也引入了合并算法。本文单独考虑了合并算法的影响,即不考虑结合 EES 与 EAD-PRO 算法平衡的情况下单纯地对合并算法做一次能耗测量,单纯的合并算法用 MREO′表示。




(a) 计算密集型任务

(b) 通信密集型任务

图 11　不同任务类型的能耗对比图

从图 11 可知:EES 和 EAD-PRO 算法结合了合并算法后,节能效果都有比较明显的提升,并高于单纯的合并算法。但是,具有 EES 和 EAD-PRO 算法平衡选择的 MREO 算法的节能效果最为明显。

## 5　结语

本文提出了一种针对同时到达、无依赖关系的多 DAG 工作流的节能方法,它通过动态地对任务集合进行整合,能够使得全局能耗最优化。对于整合后残留的松弛时间采用 DVFS 和复制调度算法进行处理,以降低局部能耗。在选取整合路径时采用有效的分支限界条件,大大降低了算法的冗余度。通过实验,证明了本文方法能够有效地降低处理器集群的整体能耗。本文的局限性在于未考虑不同时间达到的多 DAG 工作流的能耗优化问题,这是未来工作的重点。

**参考文献:**

[1] CHASE J, ANDERSON D, THAKAT P, *et al*. Managing energy and server resources in hosting centers[C]// Proceedings of the 18th Symposium on Operating Systems Principles. Canada: ACM Press, 2001: 103 – 116.

[2] ELNOZAHY E N, KISTLER M, RAJAMONY R. Energy-efficient server clusters [C]// Proceedings of the 2nd International Conference on Power-aware Computer Systems. New York: ACM Press, 2003: 179 – 197.

[3] HUANG Q J, SU S, LI J, *et al*. Enhanced energy-efficient scheduling for parallel applications in cloud[C]// Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. New York: ACM Press, 2012: 781 – 786.

[4] HSU C, FENG W. A power-aware run-time system for high-performance computing[C]// Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. Washington, DC: IEEE Computer Society, 2006: 258 – 267.

[5] ZHU D, MELHEM R, CHILDERS B. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems[J]. Parallel and Distributed Systems, 2003, 14(7): 686 – 700.

[6] RANAWEERA S, AGRAWAL D P. A task duplication based scheduling algorithm for heterogeneous systems[C]// Proceedings of the 2000 Parallel and Distributed Processing Symposium. Piscataway, NJ: IEEE Press, 2000: 445 – 450.

[7] ZONG Z L, MANZANARES A, RUAN X J, *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.

[8] 李新,贾智平,鞠雷,等.一种面向同构集群系统的任务节能调度优化方法[J].计算机学报,2012,35(3):591 – 602.

[9] BRAUN T, SIEGEL H, BECK N, *et al*. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems[J]. Journal of Parallel and Distributed Computing, 2001, 61(6): 810 – 837.

[10] ZHAO H N, SAKELLARIOU R. Scheduling multiple DAGs onto heterogeneous systems[C]// Proceedings of the 20th International Parallel and Distributed Processing Symposium. Piscataway, NJ: IEEE Press, 2006: 54 – 67.

[11] 田国忠,肖创柏,徐竹胜,等.异构分布式环境下多 DAG 工作流的混合调度策略[J].软件学报,2012,23(10):2720 – 2734.

[12] KIM K, BUYYA R, KIM J. Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters[C]// Proceedings of the 11th IEEE International Symposium on Cluster Computing and the Grid. Piscataway, NJ: IEEE Press, 2011: 1 – 8.

[13] KIMURA H, SATO M, HOTTA Y, *et al*. Emprical study on reducing energy of parallel programs using slack reclamation by DVFS in a power-scalable high performance cluster[C]// Proceedings of 2006 IEEE International Conference on Cluster Computing. Piscataway, NJ: IEEE Press, 2006: 1 – 10.

[15] CHEN G, HE W, LIU J, *et al*. Energy-aware server provisioning and load dispatching for connection-intensive Internet services [C]// Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. San Jose, CA: USENIX, 2008: 337 – 350.

[15] SHEKAR V, IZADI B. Energy aware scheduling for DAG structured applications on heterogeneous and DVS enabled processors [C]// Proceedings of 2010 International Green Computing Conference. Piscataway, NJ: IEEE Press, 2010: 495 – 502.

[16] GE R, FENG X, CAMERON K. Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters[C]// Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. New York: ACM Press, 2005: 34 – 45.

[17] GUNARATNE C, CHRISTENSEN K, NORDMAN B, *et al*. Reducing the energy consumption of Ethernet with Adaptive Link Rate (ALR)[J]. IEEE Transactions on Computers, 2008, 57(4): 448 – 461.