云环境下计算资源动态能耗感知的并行任务调度方法

曹 洁 曾国荪
(同济大学计算机科学与技术系 上海 200092)
(国家高性能计算机工程技术中心同济分中心 上海 200092)

摘 要 云计算是一种新兴的计算模式,倡导一切皆服务。要实现低成本、高效、安全、易用的云计算系统,依然面临诸多挑战,其中,高能耗已成为云计算不可忽视的问题。在计算资源电压可动态调整的环境下,为截止完成时间有要求的并行任务,提出两种满足并行任务截止时间要求的降低并行任务执行能耗的调度方法 Ssef 和 Egsa。模拟实验表明,提出的算法在保证并行任务截止完成时间要求的条件下能够有效降低并行任务的执行能耗,从而大幅度降低云计算系统的能耗开销。

关键词 云计算,绿色计算,子截止时间,节能调度
中图法分类号 TP316 **文献标识码** A

Scheduling Method for Parallel Task of Dynamic Energy-aware of Computing Resources in Cloud Environment

CAO Jie ZENG Guo-sun
(Department of Computer Science and Technology, Tongji University, Shanghai 200092, China)
(Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai 200092, China)

Abstract Cloud computing becomes more and more popular in large scale computing and data store recently because it enables the sharing of computing resources that are distributed all over the world. Cloud computing system is still facing many challenges in achieving low-cost, efficient, safe, easy-to-use computing. Saving computing resources energy has become a significant research topic which needs to be solved urgently. We proposed a subdeadline distribution approach to satisfy the deadline requirements of parallel tasks deadline. To achieve the goal of saving energy in the environment of the computing resources supply voltage to be dynamically adjusted, we proposed two energy-efficient scheduling algorithms——energy first scheduling algorithm(Ssef) and energy genetic scheduling algorithm(Egsa) to satisfy subdeadline. Repeated experiments show that this two energy-efficient scheduling strategies can reduce the energy consumption considerably while meeting deadline constraints.

Keywords Cloud computing, Green computing, Subdeadline, Energy efficient scheduling

1 引言

计算、存储和通讯技术的快速发展,以及虚拟化技术的日趋完善,使得计算资源、平台、软件可以以服务的形式向客户提供。云计算是分布计算、网格计算、普适计算、虚拟化技术、平台即服务、软件即服务等概念混合演进而来的新兴计算机科学概念,云将构筑在大量计算机、存储设备、通信网络等构成的虚拟资源池上,通过互联网向终端用户按需提供计算服务。云计算作为一种新型的计算模式,以其高可扩展性和高可用性等优点迅速成为学术界和产业界的研究热点。但是,要实现低成本、高效、安全、易用的云计算依然面临诸多挑战,其中,高能耗是云计算系统最为严重的问题之一。

信息和通信技术行业作为全球增长最快的行业之一,其碳排放也随着行业的增长而不断增长。据统计[1],目前信息和通信技术领域的碳排放占全球的2%。2008年服务于互联网的路由器、服务器、交换机、冷却设施、数据中心等各种设备总共消耗8680亿度电,占全球总耗电量的5.3%。文献[2]指出,按照目前的增长趋势,到2025年,IT行业平均能耗会达到2006年的5倍,网络领域能耗更会达到13倍。这意味着到2025年网络领域占IT业的总能耗将增长到43%。耗电问题已成为网络和信息系统持续发展的重大障碍[3]。

云计算系统中通常包含不同类型的计算机,实验结果表明,不同计算机对不同计算任务的执行功率和响应时间一般不同。例如,同一图像处理任务分别在CPU和GPU上的执行功率和响应时间不同,任务执行完成后,产生的总能耗也不同。因此,当未考虑能耗因素时,不匹配的调度方式会造成:

到稿日期:2012-12-17 返修日期:2013-03-26 本文受国家863高技术研究发展计划(2009AA012201),国家自然基金项目(61272107,61103068),NSFC-微软亚洲研究院联合资助项目(60970155),上海市优秀学科带头人计划项目(10XD1404400),教育部博士点基金项目(20090072110035),教育部网络时代的科技论文快速共享专项研究课题(20110740001)资助。

曹 洁 博士生,主要研究领域为并行分布处理、云计算,E-mail:cjjiecao@126.com;曾国荪 博士,教授,博士生导师,主要研究领域为并行分布计算、可信网络软件。

本来用较低能耗就能解决的问题却用了较高的能耗。同时，处理器中的数字电路大多采用 CMOS 集成电路，动态功耗在 CMOS 电路功耗中占主要部分，是节能研究的主要对象，其动态功率与电压的平方成正比[5]。

目前，分布式并行计算系统的能耗优化管理技术主要包括 3 类：关闭/休眠技术(resource hibernation)、电压动态调整技术(dynamic voltage scaling，简称 DVS)和虚拟化技术(virtualization)。其中，关闭/休眠技术主要用来降低空闲能耗，电压动态调整技术和虚拟化技术主要用来降低执行能耗。同一个任务在执行过程中，其执行功率会随着运行阶段、执行特征的变化而变化[6]。

绿色计算作为一个新的热点，引起了国内外很多人对能耗调度方面的关注。2003 年，Zhu 等人[7]针对同构计算环境，利用能够动态调节电压(DVS，dynamic voltage scaling)的处理机共享空闲时间的方法，提出了多个不同的调度算法，这些算法的基本原理就是利用处理机的空闲时间降低电源电压。2005 年，Ge 等人[8]同样也是针对同构计算环境，对周期任务提出了控制电源电压的两个算法；同时对非周期任务提出了具有动态优先级的单处理机调度算法。2009 年，Khan 等人[9]研究了能耗感知的任务分配问题，即将任务集分配到具有 DVS 特点的计算网格上，它的目标是降低能耗和响应时间。Mezmaz 等人[10]在云环境下，提出了基于遗传算法的混合调度算法，以最小完成时间和降低能耗为调度目标，通过基于 DVS 的方法来最小化能耗。Bradley 等人[11]提出了最小能耗的算法，即在保障一定可用性的前提下，利用负载历史去预计未来负载情况，从而调节能耗。Simiri[12]通过动态调节集群中的 CPU 数目，在使用率低的情况下将 CPU 置为睡眠模式，达到降低能耗的目的。在国内，山东大学的李新等人[13]对复制调度算法能够减少后继任务等待延时但耗费更多能量的问题，提出一种启发式处理器合并优化方法 PRO。东南大学的罗军舟等人[14]对当前服务组合方法只考虑组合服务 QoS 的优化而不考虑组合服务的能耗优化问题，提出了一种能耗感知的多路径服务组合方法 EAMSC。中国科学院计算技术研究所的张法等人[15]从网络全局角度研究网络的能耗模型和算法问题，构建了网络能耗系统优化模型，提出了能耗优化的网络数据包路由算法。

上述文献虽然从不同侧面考虑了降低能耗的调度问题，但是在计算资源可动态调整电压环境下，对如何保证具有截止完成时间要求的并行任务按规定时间完成且尽可能降低能耗的调度问题的研究并不多。为此，本文将研究如何为有截止完成时间要求的并行任务的子任务合理分配子截止时间的方法，使每个子任务在其子截止时间完成的条件下尽可能降低执行该子任务的功耗，从而降低整个并行任务执行的功耗。

2 任务、功耗和系统建模和假设

2.1 并行任务的建模

在云计算环境下，任务之间可能存在优先约束关系，表现为一个节点不能在获得它的父节点所有信息之前启动执行。考虑到任务间的优先约束，可以采用有向无环图 DAG(Directed Acyclic Graph)表示一组具有相互依赖关系和数据交换的任务，定义如下。

定义 1(并行任务) 一个并行任务可抽象表示为一个 DAG 图，即一个四元组 $DAG=(T,E,W,D)$，其中 $T=\{t_1,t_2,\cdots,t_n\}$ 表示任务的集合，n 表示任务的个数；$E=\{e_{ij}\mid t_i,t_j\in T\}\subseteq T\times T$ 表示有向边的集合，即任务之间依赖关系的集合；$W=\{w_1,w_2,\cdots,w_n\}$ 是子任务的计算量集合，$w_i\in W$ 表示子任务 t_i 的串行计算量；D 是子任务间的通信量集合 $\{d_{ij}\}$，$d_{ij}\in D$ 表示有向边 e_{ij} 两端点间需要传输的数据量。

图 1 是一个包含 8 个子任务的并行计算任务图，圆圈内的 t_i 表示节点的编号，圆圈旁边的数字表示任务节点的计算量，有向边旁边的数字表示节点间的通信量。通过处理，一般可假设并行任务 DAG 图只有一个入口节点和一个出口节点。

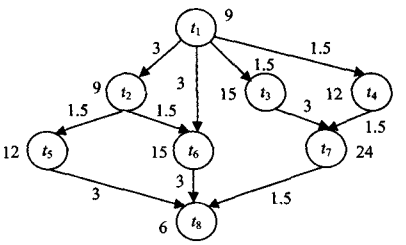

图 1 并行任务 DAG 图

2.2 云系统的建模

云环境中的计算节点的连接方式复杂多样，可能是总线、环、星形等具有规律的连接方式，也可能是没有任何规律的复杂网络，本文采用一般的“图”来刻画云资源的系统组成。本文研究的计算资源节点类型相同，计算资源在不同电压下计算速度、功耗都不相同。

定义 2(云平台) 一个现实的云系统可抽象描述为图型结构，即一个四元组 $Cloud=(R,C,B,VSE)$，其中 $R=\{r_1,r_2,\cdots,r_m\}$ 代表资源节点集，m 为资源节点的总数，每个计算资源节点类型相同，都是可动态调整电压的；$C=\{c_{ij}\mid r_i,r_j\in R\}$ 是代表云平台的通信链路集合；$B=\{b_{ij}\mid r_i,r_j\in R,c_{ij}\in C\}$ 是 C 中边的通信带宽的集合，b_{ij} 是通信边 $c_{ij}=(r_i,r_j)$ 两端点间单位时间内传输的数据量；$VSE=\{vse_1,vse_2,\cdots,vse_k\}$ 是由资源节点的电压、计算速度、功耗构成的三元组组成的集合，每个 vse_i 是一个三元组，即 $vse_i=(v_i,s_i,e_i)$，其中 v_i 表示资源的供应电压，s_i 表示计算资源在电压 v_i 下的计算速度，计算资源的计算速度指的是单位时间内完成的单位计算量，e_i 表示计算资源在电压 v_i 下的计算功率。

图 2 是一个包含 7 个资源节点的云平台拓扑图，圆圈内的 r_i 表示系统中的资源编号，边上的数字表示通信边的通信带宽。事实上，计算资源在不同电压下的计算速度和功率可通过实际运行任务的方式测得。为了简化问题的讨论，结合文献[16]，我们给出一个较理想化的电压、相对计算速度和相对功率表，如表 1 所列。

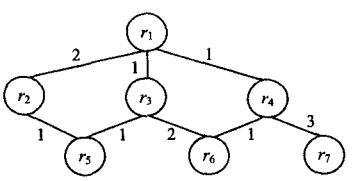

图 2 云平台的结构图

表1 电压相对速度相对功率

电压级(v_k)	计算速度(s_k)	功率 Power(v_k)
1.5(v_1)	3	47
1.3(v_2)	2	25
1.1(v_3)	1	11

2.3 能耗模型

处理器中的数字电路大多采用CMOS集成电路，CMOS集成电路的功耗主要由静态功耗和动态功耗构成。一般来说，内部短路功耗不会超过动态功耗的10%，通常忽略不计。动态功耗在CMOS电路功耗中占主要部分，其功耗模型为$P_d=ACV^2f$，其中A为电路翻转频率，C是负载电容，V是供应电压，f为时钟频率。由于电压V与时钟频率f成正比，f又和系统服务速率成正比，因此可以得出动态功耗和服务速率的n次方成正比。对于给定的处理器，通常认为$A\cdot C$是一个常量，因此，不同的供应电压对应不同的功耗。降低计算资源的供应电压，则会降低计算资源的计算速度和功耗。AMD Mobile Athlon4 DVS处理器可运行在5种电压模式下，分别是1.2V-500MHz，1.25V-600MHz，1.3V-700MHz，1.35V-800MHz和1.4V-1000MHz，电容系数为12.75nF[17]，由公式求得每种电压模式所对应的功耗分别是9.2W，12W，15.1W，18.6W和25W。

计算资源r_j在电压v_k下完成任务t_i的计算能耗$E_{jki}=ACv_k^2f_kw_i/s_k$，其中$f_k$表示资源在$v_k$下对应的时钟频率，$s_k$表示在电压$v_k$下计算资源$r_j$的计算速度。

2.4 并行任务调度执行的基本假设

所谓云平台下的并行任务调度，就是在充分考虑任务间的依赖关系的基础上，将图中的并行任务DAG各子任务分配到资源节点上进行并行协同计算的过程。假定子任务具有原子性，不可再细分，任务的执行是非抢占的。任务统一由中心调度器管理，按照某种策略将每个子任务分配到合适的计算资源上，调度器与各计算资源节点独立运行。并行任务的总能耗即为各个子任务消耗的能耗之和。通信操作可以并发执行，暂不考虑通信冲突情况。若具有依赖关系的两个任务分配到同一个计算资源节点内执行，则它们之间的通信时间忽略。计算资源在一个任务执行期间不改变计算资源的供应电压。

由于本文云平台中每个计算节点在不同供应电压下的计算速度和功率不同，因此每个子任务在不同电压下的计算时间和功耗也不同；不同计算节点间的通信带宽不同，两个子任务在不同计算资源间的通信时间也不相同。为了计算并行任务中每个子任务的最早开始时间、最早完成时间，我们以计算资源在最大供应电压下执行每个任务所用的时间作为任务的执行时间，以计算资源间的平均带宽来计算任务之间数据的传输时间。下面给出本文后面要用到的一些定义和概念。

并行任务图中每个任务的最早开始时间定义为：$Est(t_{entry})=0$，$Est(t_i)=\max\limits_{t_p\in pred(t_i)}\{Est(t_p)+Met(t_p)+Act_{pi}\}$，其中，$t_{entry}$表示没有前驱节点的入口子任务，$Met(t_p)$表示任务$t_p$在最高电压下的最小执行时间，$Act_{pi}$表示任务$t_p$通过云平台向任务$t_i$传输数据的平均通信时间，其时间通过平均带宽来求解。每个任务t_i的最早完成时间$Eft(t_i)$定义为$Eft(t_i)=Est(t_i)+Met(t_i)$。任务$t_i$的最迟完成时间定义为$Lft(t_{exit})=DL$，$DL$为截止完成时间，$Lft(t_i)=\min\limits_{t_c\in Succ(t_i)}\{Lft(t_c)-Met(t_c)-Act_{ic}\}$，其中，$t_{exit}$表示没有后继节点的出口子任务。

3 并行任务子任务截止时间分配方法及合理性分析

3.1 子任务截止时间分配方法

该方法把任务图中的初始任务t_1的最早开始时间$Est(t_1)$和出口任务t_n的最迟完成时间$Lft(t_n)$作为输入，然后给任务图中的每个子任务分配一个子截止时间，只要每个子任务在其子截止时间之前完成，就不会影响整个并行任务在截止时间之前完成。本文采取公平的策略为并行任务中的每个子任务分配子截止时间，子截止时间的分配与任务计算量的大小成正比，即任务图中处于同一层次的有着相同最早开始时间的两个任务中，子任务的计算量越大，分配的子截止时间越长。并行任务的执行跨度Elt定义为该出口任务的最迟完成时间与入口任务的最早开始时间的差，即$Elt=Lft(t_n)-Est(t_1)$。

定义3(子截止时间) 并行任务的子任务t_i的子截止时间$Sdl(t_i)$计算如下：

$$Sdl(t_i)=\frac{Eft(t_i)-Est(t_1)}{Eft(t_n)-Est(t_1)}\times Elt$$

3.2 子任务截止时间分配的合理性分析

定理1 给定一个有依赖关系约束的并行任务，满足整个并行任务完成截止时间要求的子任务截止时间分配方法所分配的子任务t_i截止时间$Sdl(t_i)$满足$Eft(t_i)\leqslant Sdl(t_i)\leqslant Lft(t_i)$。

证明：任给从入口任务到出口任务的一条路径$L=t_1\rightarrow t_2\rightarrow\cdots\rightarrow t_n$，设$t_n$的最迟完成时间为$Lft(t_n)$，其中，每个任务$t_i$的执行时间表示为$Ext(t_i)$，任务$t_{i-1}$与$t_i$间的通信时间表示为$c_{i-1}$，令$Lft(t_n)\geqslant Eft(t_n)$，即出口任务的最迟完成时间大于等于其最早完成时间。为简化问题证明，设任务t_1的最早开始时间为0。

首先，证明$Eft(t_i)\leqslant Sdl(t_i)$。对$\forall t_i\in L$，$Eft(t_i)=\sum\limits_{k=1}^{i}Ext(t_k)+\sum\limits_{k=1}^{i-1}c_k$，$Lft(t_i)=Lft(t_n)-\sum\limits_{k=i+1}^{n}Ext(t_k)-\sum\limits_{k=i}^{n-1}c_k$。由于$Lft(t_i)-Eft(t_i)=Lft(t_n)-\sum\limits_{k=i+1}^{n}Ext(t_k)-\sum\limits_{k=i}^{n-1}c_k-(\sum\limits_{k=1}^{i}Ext(t_k)+\sum\limits_{k=1}^{i-1}c_k)=Lft(t_n)-(\sum\limits_{k=1}^{i}Ext(t_k)+\sum\limits_{k=i+1}^{n}Ext(t_k))-(\sum\limits_{k=1}^{i-1}c_k+\sum\limits_{k=i}^{n-1}c_k)=Lft(t_n)-Eft(t_n)\geqslant 0$，因此$Eft(t_i)\leqslant Sdl(t_i)$。

下面证明$Sdl(t_i)\leqslant Lft(t_i)$。

由于$Lft(t_i)-Sdl(t_i)=Lft(t_n)-\sum\limits_{k=i+1}^{n}Ext(t_k)-\sum\limits_{k=i}^{n-1}c_k-\frac{Eft(t_i)}{Eft(t_n)}\times Lft(t_n)=Lft(t_n)-(Eft(t_n)-Eft(t_i))-\frac{Eft(t_i)}{Eft(t_n)}\times Lft(t_n)=\frac{[Lft(t_n)-Eft(t_n)][Eft(t_n)-Eft(t_i)]}{Eft(t_n)}\geqslant 0$，因此$Sdl(t_i)\leqslant Lft(t_i)$。

综上可知，定理得证。

从定理1可知，每个子任务的子截止时间都大于最早完

成时间且小于最迟完成时间，若每个子任务在计算资源较低的功率下能在其截止时间之前完成该任务，整个并行任务就能在其截止时间之前完成，从而能降低整个并行任务执行的能耗。

4 云环境下能耗感知的并行任务调度问题的公式化描述

假设 Ct_{ij} 表示任务 t_i 通过云平台向任务 t_j 传输数据的通信时间。令 $St(t_i,r_j,v_k)$ 表示任务 t_i 在计算资源 r_j、电压为 v_k 的开始时刻，$Et(t_i,r_j,v_k)$ 表示任务 t_i 在计算资源 r_j、电压为 v_k 的完成时刻，$T_{Comm}(t_i)=\max\{Ct_{ki}\mid t_k\in Pred(t_i)\}$ 表示任务 t_i 的所有父任务的数据都到达的时间，$Et(t_i,r_j,v_k)=St(t_i,r_j,v_k)+T_{Comm}(t_i)+w_i/s_k$，其中，$s_k$ 表示计算资源在电压 v_k 时的计算速度。令 $Reso(t_i)$ 表示执行任务 t_i 的计算资源，$Pred(t_i)$ 表示 t_i 的直接前驱节点集。令 $Power(v_k)$ 表示计算资源在供应电压 v_k 下的功率。

在云平台下，最小化并行任务的完成能耗，并且满足并行任务截止时间的调度问题可以公式化地描述如下：

$$\begin{cases}\min\sum_{i=1}^{n}(w_iPower(v_k)/s_k)\\ \max\{Et(t_1),Et(t_2),\cdots,Et(t_n)\}\leqslant Deadline\\ St(t_i,r_j,v_k)\geqslant 0\\ Et(t_i,r_j,v_k)=St(t_i,r_j,v_k)+T_{Comm}(t_i)+w_i/s_k\\ St(t_j,r_n,v_h)-St(t_i,r_m,v_k)\geqslant w_i/s_k+Ct_{ij}\\ St(t_j,r_m,v_h)-St(t_i,r_m,v_k)\geqslant w_i/s_k\end{cases}\tag{1}$$

上述公式可通俗地解释为：(1)起始任务的开始执行时刻为 0，其它任务的开始执行时刻大于等于 0；(2)对于相邻两任务，后继任务的开始执行时刻不先于前驱任务的开始执行时刻、前驱任务的执行时间和前驱任务的数据传输时间三者之和；(3)同一计算资源上任务的执行时间不能重叠；(4)并行任务的结束时刻小于等于规定的截止时刻；(5)最小化并行任务的计算总能耗。

5 能耗感知的调度方法

式(1)的调度问题是一个 NP 完全的问题，难以获得最优解，一般只能用启发式方法来求解近优解，本文采用能耗优先和基于遗传智能的调度算法。

5.1 满足子截止时间能耗优先的调度算法 Ssef(satisfy swb-deadline energy first scheduling algorithm)

算法 1 满足子截止时间能耗优先的调度算法 SSEF()

输入：任务图 DAG=$\{t_1,t_2,\cdots,t_n\}$，云计算资源系统 Cloud，入口任务 t_1 的最早开始时间 Est(t_1)和出口任务 t_n 的最迟完成时间 Lft(t_n)

输出：子任务、计算资源、电压分配序列 Assign=$\{(t_i,r_j,v_k)\}$，整个任务完成时间 T_{total}，整个任务的能耗和 E_{total}

```
Ssef()
{
L1. SDL[]=CountSdl(DAG);//计算任务图 G 的各个子任务的子截止时间
L2. L←{t_i|indegree(t_i)=0,1≤i≤n};//DAG 图中，入度为零的子任务进入子任务准备队列 L
L3. ρ←{r_1,r_2,…,r_m};//初始化空闲计算资源集合
L4. Assign←Φ;//子任务、计算资源、电压分配序列
L5. ε←L;//子任务执行队列初始化
L6. St(t_i)=0,Et(t_i)=0,1≤i≤n //所有子任务的开始和结束执行时刻初始化为 0
L7. τ_idle(r_j)=0,1≤j≤m;//计算资源空闲等待时刻初始化为 0
L8. T_total=0,E_total=0;
L9. do until ε=Φ
L10. {for each t_i∈ε
{//基于贪婪算法思想，在空闲资源集合中选择一个资源 r_j 和相应电压 v_k 使得满足任务子截止时间要求的条件下计算能耗最小
L11. (t_i,r_j,v_k)←select(t_i,ρ,v_k) ∧min{w_iPower(v_k)/s_k,…}) ∧Et(t_i,r_j)≤Sdl(t_i);
L12. Assign←Assign +{(t_i,r_j,v_k)};
L13. ε←ε−{t_i};
L14. ρ←ρ−{r_j};//将处理任务的计算资源从空闲队列中去除
L15. St(t_i)=max(St(t_i),τ_idle(r_j));
L16. T_Comm(t_i)=max{Ct_ki|t_k∈Pred(t_i)};
     //计算 t_i 的所有父任务的数据通信时间
L17. Et(t_i)=St(t_i)+T_Comm(t_i)+w_i/s_k;//计算任务 t_i 在所在资源上的完成时间
L18. τ_idle(r_j)=Et(t_i);
L19. E_total=E_total+w_iPower(v_k)/s_k;
L20. for each immediate successor v_x of task v_i
L21. {St(t_x)=max(St(t_x),Et(t_i));
L22.    indegree(t_x)=indegree(t_x)−1;
L23. if indegree(t_x=0)  ε←ε+{t_x};
     }
L24.  ρ←ρ+{r_j};
   }//end for each t_i∈ε
 } //end do
L25. T_total=max{Et(t_i),1≤i≤n};
}
```

上面 L10－L24 中的 for each 循环是针对并行任务的子任务，要循环 n 次。L11 为子任务选择资源基于贪婪算法思想，在空闲资源集合中选择一个资源 r_j 和相应电压 v_k，使得在满足任务子截止时间要求的条件下计算能耗最小，平均需要 $m/2$ 次。所以，满足子截止时间能耗优先的调度算法 Ssef()的时间复杂性为 $O(|T|m)$。利用上述满足子截止时间能耗优先的调度算法 Ssef()，求得图 1 所示的并行任务在执行完成截止时间为 35 的约束条件下，在图 2 所示云计算资源平台上执行的一个子任务、资源、电压分配序列为 Assign=$\{(t_1,r_3,v_2),(t_2,r_1,v_2),(t_3,r_3,v_2),(t_4,r_6,v_2),(t_5,r_1,v_2),(t_6,r_3,v_2),(t_7,r_6,v_2),(t_8,r_3,v_1)\}$，整个任务执行时间为 32.25，计算能耗和为 1266，比在最高电压下执行整个并行任务所消耗的能耗 1598 节约了 20.7%。

5.2 基于遗传智能的调度算法

5.2.1 遗传算法设计

(1)编码方案：本文遗传算法所采用的染色体 X 由两部分组成：任务串 T_s 和资源串 R_s，即 $X=T_s+R_s$。T_s 是并行任务图 DAG 满足任务之间优先关系的一个拓扑序列；R_s 由执行任务的资源标号组成。图 3 是一个染色体示例，$t_{x_1}t_{x_2}\cdots t_{x_n}$

表示任务图 DAG 的一个拓扑序列，$r_{Reso(t_i)k_i} \in R$ 表示执行子任务 $t_{x_i} \in T$ 的相应计算资源序号 $Reso(t_i)$ 及其对应的电压级 k_i。不妨假设 $n>m$，那么 $r_{Reso(t_1)}\ r_{Reso(t_2)} \cdots r_{Reso(t_n)}$ 之中有相同资源，相同资源的供应电压不一定相同，即存在同一资源不同电压下执行两个以上的任务。

$$\underbrace{t_{x_1}\ t_{x_2} \cdots t_{x_n}}_{Ts} \mid \underbrace{r_{Reso(t_1)k_1}\ r_{Reso(t_2)k_2} \cdots r_{Reso(t_n)k_n}}_{Rs}$$

图 3　调度方案染色体编码

(2)适应度函数：本文云平台中的并行任务调度目标是在满足并行任务截止时间要求的条件下最小化消耗的能耗。令 X 表示一个有效的染色体，定义适应度函数为：$f(X)=1/\sum_{i=1}^{i=n}(w_{X.Ts[i]}Power(v_k)/s_k)$，其中 $w_{X.Ts[i]}$ 表示任务 $X.Ts[i]$ 的计算量，s_k 表示执行任务 $X.Ts[i]$ 的计算资源在其供应电压 v_k 下的计算速度，$Power(v_k)$ 表示计算资源在供应电压 v_k 下的功率。不满足截止时间要求的染色体适应度值定为 0。

(3)选择算子：在当前群体中，选择出一些比较优良的个体，并将其复制到下一代群体中。本文采用比例选择算子，即个体被选中并遗传到下一代群体中的概率与该个体的适应度大小成正比。

(4)交叉算子：在当前群体中，就一对染色体 X_1，X_2，按规定的概率对任务串随机产生一个断点，将两个任务串分成左右两部分，第一个染色体断点的左部分直接作为后代的第一部分，根据第二个染色体断点的右部分任务顺序构造后代的第二部分；资源串的交叉是对两个资源串随机产生一个断点，将两个资源串分成左右两部分，将断点右边部分交换完成资源串的交叉操作。任务串的交叉操作如图 4 所示。

交叉前

父个体 1　$t_1\ t_4 \mid t_3\ t_5\ t_2\ t_6\ t_7\ t_8$

父个体 2　$t_1\ t_2 \mid t_3\ t_4\ t_6\ t_5\ t_7\ t_8$

交叉后

子个体 1　$t_1\ t_4 \mid t_3\ t_2\ t_6\ t_5\ t_7\ t_8$

子个体 2　$t_1\ t_2 \mid t_3\ t_5\ t_4\ t_6\ t_7\ t_8$

图 4　任务串的交叉操作

(5)变异算子：为了产生新的个体，在当前群体中，针对单个染色体 $X=T_s+R_s$，按规定的概率，随机选择任务串 T_s 上的一位进行突变，合法的突变必须保证整个 DAG 任务图的依赖关系不变；同时随机选择资源串 R_s 上的一位进行突变。

5.2.2　能耗遗传调度算法 Egsa(enery genetic schedu ling algorithm)

算法 2　可用性遗传调度算法 Egsa()

输入：任务图 DAG=$\{v_1, v_2, \cdots, v_n\}$，云计算资源 Cloud

输出：子任务、计算资源、供应电压分配序列 Assign=$\{(t_i, r_j, v_k)\}$，整个任务执行时间 T_{total}，整个任务的能耗和 E_{total}

```
Egsa()
{X^0←t^0_x1 t^0_x2,…,t^0_xn | r^0_Reso(t1)k1 r^0_Reso(t2)k2,…,r^0_Reso(tn)kn;//生成初始种群
 i=0;//演化代数
 while(fitness(X^i)>=Fit0 ∨ i>=Gen_max)//满足适应度阈值，超过容许的最多的演化代数
 {i++;
  X^i←select(X^(i-1),ρs) //选择优秀个体复制到一代种群，ρg 为选择和复制概率
  X^i←cross(X1^(i-1),X2^(i-1),ρc)//对 i-1 代中的两个染色体执行交叉操作，ρc 为交叉概率
  X^i←mutate(X^(i-1),ρm) //对 i-1 代中的染色体执行变异操作，ρm 为变异概率
 fitness(X^i)=Σ_{i=1}^{i=n}(w_X.Ts[i]/s_ki) //计算种群的适应度
 }
 get Assign={(t_i,r_j,v_k)} from the best chromosome X;//由染色体求解调度方案 Assign={(t_i,r_j,v_k)}
 get T_total,E_total like Ssef();//类似满足子截止时间能耗优先的调度算法 Ssef()求解任务的完成时间和能耗和
}
```

6　模拟实验及结果分析

为了测试本文提出的算法对并行任务调度问题的求解效果，考虑在不同截止时间和不同计算资源数下用下面两组实验来验证。本文用 CloudSim[18] 来模拟云计算环境，CloudSim 是由澳大利亚墨尔本大学的网格实验室和 Gridbus 项目推出的云计算仿真软件，支持云计算的资源管理和调度模拟，并提供了以下新的特点：(1)支持大型云计算的基础设施的建模与仿真；(2)一个自足的支持数据中心、服务代理人、调度和分配策略的平台。本文使用的计算资源节点个数在 20 到 100 之间变化，计算资源间的通信带宽等都在预定范围内以规定的概率产生，计算资源的性能情况如表 1 所列。并行任务 DAG 由并行仿真任务的自动生成软件随机生成，每个子任务的计算量大小、任务间的通信量等都在预定范围内以规定的概率产生。实验过程中每种情况执行多次调度算法，并行任务执行总能耗取其平均值。

6.1　不同截止时间的影响实验

这组实验比较了不同截止时间取值对算法的影响。从图 5 可以看出，当截止时间从 35 到 55 逐渐变化时，相对于计算资源在最快速度下完成并行任务，各算法的能耗节约百分数不断增加，这是由于在整个并行任务完成截止时间增加的条件下各子任务可选择速度更小、功率更小的计算资源执行任务，从而降低了整个并行任务的执行能耗；在各种截止时间条件下，Egsa 算法都好于 Ssef 算法，这是因为 Egsa 算法是基于全局的，而 Ssef 算法是基于局部的。

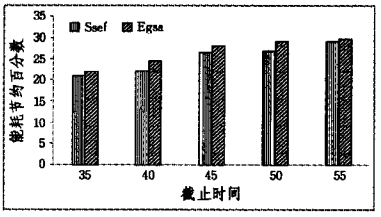

图 5　不同截止时间下的能耗节约百分数

6.2　不同计算资源数下的能耗影响实验

本实验分析不同系统计算资源数对并行任务平均完成所需能耗的影响。为评价本文所提算法的性能，将本文算法与 HEFT[19] 算法和 TDS[20] 算法在不同系统资源数情况下进行了比较。本实验使用的子任务数为 60，完成截止时间是其最快完成时间的 1.5 倍。图 6 所示为不同计算资源数下的并行

调度算法的性能比。

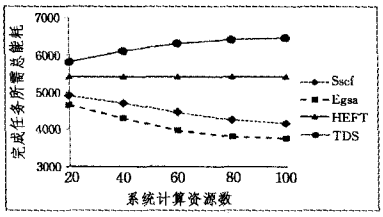

图 6　不同计算资源数下的调度算法的性能比较

由图 6 可知，在不同计算资源数下，Egsa 和 Ssef 表现较好，TDS 表现最差，其原因如下。

HEFT 算法是表调度算法的典型代表，HEFT 算法尽可能将任务调度到具有最早完成时间的资源节点，而不考虑完成任务所需的能耗，每个任务都是在最快计算资源上执行，所以其能耗和是个常数。TDS 是另一个基于关键路径的调度算法，TDS 算法分配关键路径上的任务到同一计算节点上，此外，如果任务已经被分配到其它计算节点上，TDS 算法将复制能缩短调度长度的任务，也就是说 TDS 算法可能复制任务多次，由于额外执行多个任务，其消耗的能耗最多。对于 Egsa 和 Ssef，随着计算资源数的增加，有更多的计算资源参与计算，增加了任务执行并发的程度，此时，更多的子任务有更大的选择余地以较低的执行能耗在其子截止时间之前完成该任务，从而能达到进一步降低能耗的目的；但随着资源数的进一步增加，能耗节约的趋势趋于稳定，这是因为每个子任务都可以在最低功率下执行，这时再增加计算资源已经没有太大的意义；总的来说，Egsa 算法都好于 Ssef 算法，这是因为 Egsa 算法是基于全局的，而 Ssef 算法是基于局部的。

结束语　本文在计算资源可动态调整电压的情况下，提出了两种满足截止时间需求的降低并行任务执行能耗的调度方法。对截止时间约束的并行任务的子任务截止时间分配问题进行了讨论，提出一种按任务计算量的大小分配任务子截止时间的分配方法。该方法将整个并行任务的截止时间划分为每个子任务的子截止时间，使每个任务在计算资源尽可能低的计算功率下执行，减少每个子任务的执行能耗，从而降低完成并行任务的总能耗。但是，本文的截止时间分配方法对非关键路径上子任务截止时间的分配不是很理想，因为非关键任务对并行任务的完成影响不大，理应对非关键任务分配较多的子截止时间以更大程度地降低子任务的执行能耗，因此，需要进一步研究非关键子任务更合理的子时间分配方法。此外，进一步研究通信链路可动态调整电压、能耗可变情况下的并行任务调度方法。

参 考 文 献

[1] Plan G A. An efficient Truth[R]. Global Action Plan Report. http://global action plan. org. uk, Dec. 2007

[2] Yun D, Lee J. Research in green network f or future Internet [J]. Journal of KIISE, 2010, 28(1): 41-51

[3] Lin Chuang, Tian Yuan, Yao Min. Green network and green evaluation: Mechanism, modeling and evaluation [J]. Chinese Journal of Computers, 2011, 34(4): 593-612

[4] 谭一鸣，曾国荪，王伟. 随机任务在云计算平台中能耗的优化管理方法[J]. 软件学报，2012，23(2)：266-278

[5] Venkatachalam V, Franz M. Power reduction techniques for microprocessor systems [J]. ACM Computing Surveys, 2005, 37 (3): 195-237

[6] Blume H, Livonius J V, Rotenberg L, et al. OpenMP-Based parallelization on an MPcore multiprocessor platform—A performance and power analysis[J]. Journal of Systems Architecture, 2008, 54(11): 1019-1029

[7] Zhu D, Melhem R, Childers B R. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2003, 14(7): 686-700

[8] Ge R, Feng X, Cameron K W. Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters[C]// Proceedings of the ACM/IEEE Conference on Supercomputing. November 2005: 34-44

[9] Khan S U, Ahmad I. A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids [J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(3): 346-360

[10] Mezmaza M, Melabb N. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems [J]. J. Parallel Distrib. Comput., 2011, 71: 1497-1508

[11] Bradley D, Harper R, Hunter S. Workload-based power management for parallel computer systems[J]. IBM Journal of Research and Development, 2003, 47(5): 703-718

[12] Lawson B, Smirni E. Power-aware resource allocation in high-end systems via online simulation[C]// Proceedings of the 19th Annual International Conference on Supercomputing. Cambridge, USA, 2005

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

[14] 朱勇，罗军舟，李伟. 一种工作流环境下能耗感知的多路径服务组合方法[J]. 计算机学报，2012，35(3)：627-638

[15] 张法，Anta A F，王林，等. 网络能耗系统模型及能效算法[J]. 计算机学报，2012，35(3)：603-615

[16] Mezmaz M, Melab N, Kessaci Y, et al. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems [J]. J. Parallel Distrib. Comput., 2011, 71: 1497-1508

[17] Iqbal M A, Javed D M, Qayyum U. Curvelet-based Image Compression with SPIHT [C]// 2007 International Conference on Convergence Information Technology. Washington: IEEE Computer Society, 2007: 961-965

[18] Calheiros R N, Ranjan R, Rose C A F D, et al. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services [R]. GRIDS-TR-2009-1. Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, March 2009

[19] Topcuoglu H, Hariri S, Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3): 260-274

[20] Ranaweera an S, Agrawal D P. A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems[C]// Proc. Parallel and Distributed Processing Symp. May 2000: 445-450