软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(Suppl.(2)):78−89
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于改进 GEP 和资源改变量的局部云任务调度算法*

李昆仑, 王 珺, 宋 健, 董庆运

(河北大学 电子信息工程学院,河北 保定 071000)

通讯作者: 李昆仑, E-mail: likunlun@hbu.edu.cn

摘 要: 针对云计算中一些现有的基于批量调度模式和进化算法的动态云任务调度算法计算量较大,计算时间成本较高的现象,提出了一种基于改进基因表达式编程(GEP)和资源改变量的局部云任务调度算法.首先结合云任务调度的特点对普通 GEP 算法做出了相应的改进,然后采用加权求和的方式构造了一个基于综合利用率和能耗的适应度函数,最后依据物理机综合利用率的差异给出了基于改进 GEP 和资源改变量的局部云任务调度算法.基于资源改变量的局部云任务调度算法,通过对任务运行情况和物理资源使用情况进行监控,合理设定阈值,以减少参与调度物理机的个数,从而降低任务调度算法的时间成本.基于 RH(rolling horizon)模型,通过实验将所提出的算法与普通遗传算法、全局 GEP 算法进行了比较,可知该算法不仅可以降低寻优时间,不易陷入局部最优解,且具有较快的收敛速度.

关键词: 云计算;GEP 算法;ETC 矩阵;任务调度

中文引用格式: 李昆仑,王珺,宋健,董庆运.基于改进 GEP 和资源改变量的局部云任务调度算法.软件学报,2015,26(Suppl.(2)):78−89. http://www.jos.org.cn/1000-9825/15018.htm

英文引用格式: Li KL, Wang J, Song J, Dong QY. Local cloud task scheduling algorithm based on improved GEP and change of resources. Ruan Jian Xue Bao/Journal of Software, 2015,26(Suppl.(2)):78−89 (in Chinese). http://www.jos.org.cn/1000-9825/15018.htm

Local Cloud Task Scheduling Algorithm Based on Improved GEP and Change of Resources

LI Kun-Lun, WANG Jun, SONG Jian, DONG Qing-Yun

(College of Electronic and Information Engineering, Hebei University, Baoding 071000, China)

Abstract: Cloud computing has become the focus information processing and many related fields with its powerful computing and storage capacity. For some of the existing phenomenon about the large calculation and high computing time cost in cloud task scheduling algorithms based on the batch model and evolution algorithm, this paper presents a local cloud task scheduling algorithm based on improved GEP and change of resources. In the process of designing the algorithm, it is first improved the GEP algorithm according to the characteristics of cloud task scheduling. And then, this paper constructs a fitness function considering both the comprehensive utilization and energy consumption. Finally, this paper constructs a local cloud task scheduling algorithm based on improved GEP and comprehensive utilization. The algorithm proposed in this paper reduces the computing time cost by monitoring the physical resource usage and reducing the number of physical machines involved in the task scheduling. The comparison experiments among GEP, genetic algorithm and the algorithm proposed in this paper based on RH (rolling horizon) model has been made. The results show that the proposed algorithm can not only reduce the optimization time, hard to fall into the local optimal solution, but also has the faster convergence speed.

Key words: cloud computing; GEP algorithm; ETC matrix; task scheduling

随着数据采集技术、计算机网络技术的发展,许多应用领域都存储了大量的数据,如何存储与管理这些数

* 基金项目: 国家科技支撑计划(2013BAK07B04); 河北省自然科学基金(F2013201170); 河北省高等教育科学技术研究重点项目(ZD2014008)

收稿时间: 2014-05-02; 定稿时间: 2014-08-22

© 中国科学院软件研究所 http://www.jos.org.cn

据,并对其进行有效的处理与分析,是目前很多领域亟待解决的问题(大数据问题).云计算技术不但可以为这些数据提供存储空间,而且还可以提供强有力的计算资源,为大数据问题的解决提供了一条有效的途径.Google 公司早在 2003 年凭借着 GFS 文件系统、Map/Reduce 分布式编程环境、BigTable 分布式大规模数据库管理系统最早打开了云计算之门,开发出了一系列的云产品,如基于 PaaS 层的 Google App Engine,Google earth,Google 搜索引擎等.Amazon 公司、IBM 公司、中国的新浪网也紧随其后,相继推出了自己的云平台和云产品,形成了云计算百花齐放的局面.

云计算是效用计算和网格计算的总和,是一个可以被用于调度和使用的资源池[1].通过对云中计算资源和存储资源的整合和调度,使得用户能够在任何时间、任何地点都能对海量数据有效地存储和分析.云计算具有可扩展、可伸缩、透明、经济等特点[2],使得开发者和用户能够根据自身需要合理地改变云平台中资源分配策略和虚拟机使用策略,甚至可以构建一个专属的私有云平台.

云计算是利用网络把不同物理位置上的各类资源整合成为一个逻辑整体,从而为用户提供一体化的应用服务,因此,云环境中的资源分配和任务调度关系着云平台的整体使用效率,在云计算中起着重要作用.目前云环境下的任务调度算法主要包括:基于 *M*/*M*/1 队列模型的任务调度算法、基于时间序列分析的任务调度算法和基于遗传算法的任务调度算法.遗传算法以其出色的寻优能力在一定程度上克服了云环境下资源管理和调度的 NP 难题,在云环境下的任务调度领域有了一定的应用.基因表达式编程(gene expression programming,简称 GEP)算法结合了传统的遗传算法和遗传编程的优点,弥补了传统遗传算法收敛速度低和容易陷入局部最优解的不足,在很大程度上克服了基因编程中由于对树结构数据进行操作而造成的效率低下和代码膨胀的问题.GEP 算法规定了染色体编码的长度,使其在进化过程中不需要随时对编码进行调整,同时也能保证解树的复杂度不超过一定的值,提高了计算效率.

本文将 GEP 算法引入到了云计算的任务调度领域,提出了一种基于改进 GEP 和资源改变量的局部云任务调度算法.针对云计算中基于进化理论的寻优算法容易陷入局部最优解的问题,本文对 GEP 的编码和翻译方式做出了改进,将改进的 GEP 算法引入到了云任务调度领域.同时,为了降低批量调度模式中基于进化理论云任务调度算法的运行时间成本,进一步提高云任务调度的效率和实时性,提出了基于资源改变量的局部云任务调度算法,该算法依据物理机综合利用率的阈值按照实际需求将物理机分成了参加调度的物理机和不参加调度的物理机,从而减少了参加调度物理机的个数,降低了基于进化理论云任务调度算法的计算复杂度,节约了运行时间成本,并通过实验验证了本算法的适用性和有效性.

1 云任务调度的现状

早在云计算产生之初,人们就已经对云环境下的负载均衡和任务调度策略给予了高度关注.文献[3,4]概述了云环境下任务调度算法的划分,将云环境下的任务调度算法分成了静态任务调度算法和动态任务调度算法两类.文献[5−7]将任务细分为计算型任务和存储型任务,并运用虚拟机迁移技术提出了云环境下虚拟机调度策略,以实现任务运行时间最短和消耗资源最少的目的.其中,文献[5]从资源消耗最少的角度对虚拟机进行分配,这样虽节约了计算的成本却相对延长了任务的完成时间.

文献[8]根据动态调度云任务方式的不同,进一步将动态任务调度算法分成了实时模式的动态云任务调度算法和批量模式的动态云任务调度算法.其中,实时调度模式的实时性较强,对算法的计算效率要求较高.目前用于实时调度模式的云任务调度算法主要是一些简单的算法,诸如:基于公平原则的先来先服务(FIFO)任务调度、基于负载均衡度考虑的轮询(round robin)任务调度、基于 *M*/*M*/1 队列模型的任务调度规则[9].批量调度模式的实时性相对较弱,调度结果较好,但是该调度算法的计算量较大,运行时间成本较高.目前用于批量调度模式的云任务调度算法以遗传优化算法为主[10,11].

1975 年,Holland 提出了著名的遗传算法(genetic algorithm,简称 GA)[12].该算法因其自身的寻优特性,可用来解决云环境下任务调度和资源配置过程中的 NP 难题,在静态云任务调度算法中已得到了应用[13].1992 年,Dorigo 通过将 GA 算法和蚂蚁觅食方式相结合提出了蚁群算法(ant colony optimization,简称 ACO)[14,15],目前

© 中国科学院软件研究所 http://www.jos.org.cn

蚁群算法也已经应用到云环境下的任务调度和资源配置领域[16,17].蚁群算法继承了 GA 算法的寻优特性,弥补了 GA 算法容易过早收敛和易于陷于局部最优解的不足.

1992 年,Koza 创立了遗传编程(gene programming,简称 GP)算法,与 GA 算法同样基于达尔文的生物进化理论[18].近年来,随着 GP 算法的不断完善,该算法已经应用到了教育、医疗、建筑等领域.2001 年,Ferreira 结合了 GA 和 GP 的优点创立了基因表达式编程(gene expression programming,简称 GEP)算法[19],弥补了 GA 算法中的个体必须线性定长和 GP 算法容易产生无效基因的不足.GEP 算法中的个体虽然被编码成线性定长的形式,但却能够表示不同形状、不同大小的非线性实体.本文针对批量处理模式的动态云任务调度算法,将 GEP 算法引入到云计算中的任务调度领域,以提升云任务调度算法的全局寻优能力.通过设置阈值减少参与任务调度算法的物理机个数,降低批量模式下云任务调度的运行时间成本,希望得到更好的结果.

2 云任务调度模型及描述

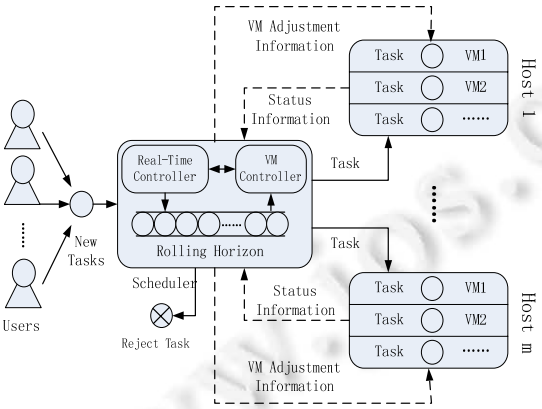


Fig.1 Rolling horizon model

图 1 RH 云任务调度模型

(1) 云任务调度模型(RH 模型)

滚动视野(rolling horizon,简称 RH)优化云任务模型将云任务的调度看作是一种排队模型.如图 1 所示,调度器中的实时监控器会实时地对排队队列中任务的截止期限、资源需求情况、虚拟机的使用情况以及物理资源的可用情况进行监视,并将监视结果传输给虚拟机控制器.虚拟机控制器按照相应的分配策略将任务分配到虚拟机中运行.因此,RH 云任务调度模型具有良好的实时监控能力,能够实时地完成任务分配和调度工作,是一种动态的任务调度模型.文献[20,21]分别对 RH 任务调度模型做出了相应的描述,同时,基于 RH 模型提出了相应的动态云任务调度算法.

(2) 云任务调度的描述

云任务调度可以归结为一个 0-1 背包问题的规约[22],即将物理机视为背包,将运行于虚拟机中的任务视为需要装入“背包”中的物品,且物品不能够重复装入“背包”.云任务调度问题可以被描述为如何在有限的“背包”数量中装入更多的“物品”,或如何用最少的“背包”装入尽可能多的“物品”.

云任务运行于由物理机产生的虚拟机中,即物理机是中资源的主要提供者,物理机可以表示成一个 1×4 维向量的形式.如公式(1)所示.物理机 $i(R_i)$被表示成一个 Cr_i 物理 CPU、Mr_i 物理内存、Sr_i 物理硬盘存储、Br_i 物理链路带宽的属性集合.公式(2)表示所有物理机提供的物理资源的总和,其中,N 为云系统中物理机的个数.

$$R_i=\left(Cr_i,Mr_i,Sr_i,Br_i\right) \tag{1}$$

$$R=\sum_{i=1}^{N}R_i \tag{2}$$

云任务是物理资源的需求者,云任务可以被表示为一个 1×6 维向量的形式.如公式(3)所示.任务 $i(Task_i)$被表示成 tn_i 任务机器指令数,tb_i 为任务开始时间,te_i 为任务结束时间,Ct_i 为任务运行所需的 CPU 资源、Mt_i 为任务运行所需的内存资源、St_i 为任务运行所需的硬盘存储资源的属性集合.公式(4)表示所有物理机提供的物理资源的总和,其中,m 为云系统中需要调度的任务个数.

$$Task_i=\left(tn_i,tb_i,te_i,Ct_i,Mt_i,St_i\right) \tag{3}$$

$$T=\sum_{i=1}^{m}Task_i \tag{4}$$

依据以上对云任务和物理机的数学建模,云任务调度问题可以被描述为:在满足物理机提供的物理资源多

© 中国科学院软件研究所 http://www.jos.org.cn

于云任务运行所需要的资源的条件下,快速地将尽可能多的任务分配到尽可能少的物理机中运行的问题.本文正是在此描述的基础上针对批量动态调度模式下任务的快速和合理分配问题提出了基于改进 GEP 和资源改变量的局部云任务调度算法,在降低调度算法运行时间成本的同时也提高了资源的使用效率.

3 GEP 算法简介

GEP 算法是由 Ferreira 于 2001 年首次提出的,可利用一个或多个基因算子,依照适应度函数对种群进行寻优的基因算法.该算法不仅保留了 GA 算法变异和转换等遗传操作的简单性,而且保留了 GP 算法在寻优求解过程中巨大的空间搜索能力.GEP 算法最突出的特性就是能够通过一种定长的染色体编码方式表达不固定非线性的实体[19].

3.1 GEP算法的表达方式

GEP 算法中的染色体有两种表达方式,分别为染色体表达方式和表达式树表达方式.这两种表达方式之间可以相互转换,其中从染色体到表达式树的过程被称为翻译.因此,GEP 算法也同时对应着两种语言:基因语言和表达式树语言.如图 2 所示.

图 2(a)中的数学表达式可以被编码为图 2(b)中的基因表达式,*Q* 代表开平方.将基因表达式中的元素按从上到下、从左到右的顺序以树的形式表达出来就生成了图 2(c)所示的表达式树,完成了从基因表达式到表达式树的翻译过程.

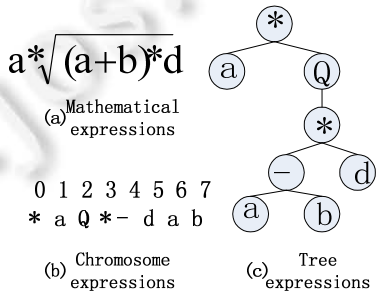


Fig.2 Expressions of GEP algorithm

图 2 GEP 算法的表达方式

3.2 GEP算法的基因结构

GEP 算法的基因是由函数集和终点集组合而成的,如图 3 所示.函数集为可能用到的函数(通常为数学运算符)的集合 $F=\{Q,*,/,-,+\}$,终点集为参与运算的变量的集合 $T=\{a,b\}$.

```
012345678901234567890123456789
*b+a-aQab+//+b+babbabbbababbaaa
```

Fig.3 Genetic structure of GEP algorithm

图 3 GEP 算法的基因结构

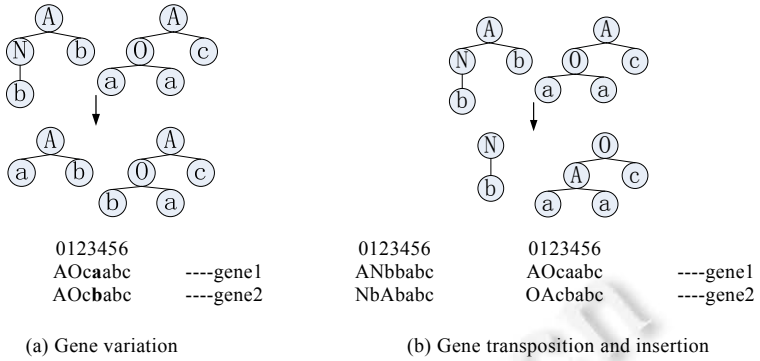
GEP 算法的基因包含头部和尾部两个部分,图 3 中加粗的部分为基因的尾部,没有加粗的部分为基因的头部.头部基因是由函数集和终点集中的元素随机组合而成的,其长度 h 可以人为设定,显然,本例中的头部长度 $h=15$.尾部基因只能由终点集中的元素随机组成,且其长度 t 为 $t=h(n-1)$,其中 n 为所需变量最多函数的参数个数.当尾部长度满足上式时,就可以避免 GEP 基因在变异遗传的过程中产生无效的基因组合,而且尾部多出的看似无用的元素也为 GEP 基因在变异过程中提供了更大的变异空间和基因优化的可能性[21].

3.3 GEP算法基因的操作

基因操作是 GEP 算法的核心,GEP 算法正是通过对基因的变异、转座、复制、重组等操作来对下一代种

© 中国科学院软件研究所 http://www.jos.org.cn

群进行优化,从而达到寻优的目的.图 4 中介绍了两种基本的基因操作:变异和转座.



(a) Gene variation (b) Gene transposition and insertion

Fig.4 Basic operations of gene

图 4 基因的基本操作

如图 4(a)所示,gene1 中 1 号位的元素 *N* 突变成了 *a*,不仅改变了基因的表现形式,还从根本上改变了表达式树的形式,变成了完全不同的数学形式,使基因有了较高的寻优能力.同理,图 4(b)表示的是基因转座操作,将 gene1 中 1,2 号位上的基因提前到了 0,1 号位,而之前的 0 号位的基因被移动到了 2 号位,从而实现了基因的另一种变化方式.基因的转座操作与基因的突变操作一起加快了 GEP 算法的求解过程.

4 基于改进 GEP 和资源改变量的局部云任务调度策略

本文正是基于 GEP 算法在全局寻优能力和表达方式灵活、可移植性强等方面的优点,为了解决其他寻优算法容易过早收敛于局部最优解的问题,将 GEP 算法引入到了云计算领域.根据云计算中任务分配的特点对 GEP 染色体的编码和解码方式作了一定的改进.同时,针对云计算中一些现有的基于批量调度模式和进化算法的动态云任务调度算法计算量较大、计算时间成本较高的现象,提出了一种基于改进 GEP 和资源改变量的局部云任务调度算法,不仅考虑了如何使云任务调度算法的时间成本最低,而且还考虑了如何提高云任务调度的综合利用率.

在一个云任务调度系统中提升某一个物理机的利用率势必会造成该物理机的能耗的增加,但一个云任务调度系统是由多个物理机组成的一个整体,每台物理机的能耗系数都不尽相同,如果在满足用户对服务质量需求的情况下,通过合理的云任务调度策略,减少参与运行任务的物理机个数,或提高能耗系数较低的物理机的利用率,降低能耗系数较高的物理机的利用率,就能在一定程度上降低云任务调度系统的整体能耗.

云任务调度的能耗也分成了静态能耗和动态能耗两个部分,其中静态能耗为物理机运行任务时的能耗,静态能耗与物理机的使用率有关.而动态能耗是云系统中由于虚拟机的迁移和任务之间的联系所产生的能量损耗,该损耗不仅与物理机的使用效率相关,还与需要迁移的虚拟机的大小、虚拟机迁移的次数以及虚拟机迁移的距离有关,文中所提出的云任务调度策略,通过设立阈值的方式减少了参与任务调度的物理机的个数,减少了虚拟机迁移的范围,进而减少了虚拟机迁移的次数,从而能够在一定程度上降低云系统中的动态能耗,是一个综合考虑了物理机的利用率和能耗的云任务调度策略.

4.1 GEP编码方式的改进

(1) 任务编码

本文中采用直接编码的方式对任务进行编码.即如果有 N 个任务,N=1,2,…,N,则顺序地将这些任务编码为 1,2,3,4 等等,如果某个任务又能分成若干个独立的子任务,则子任务的总数(SN)和 T_{ij} 的编码(m)可以通过公式(5)和公式(6)得出,其中,$T(n)$为第 n 个任务所划分出的子任务个数.

$$SN=\sum_{n=1}^{N}T(n) \tag{5}$$

© 中国科学院软件研究所 http://www.jos.org.cn

$$m=\sum_{n=1}^{i-1}T(k)+j \tag{6}$$

例如,已知第 1 个任务可以被分为 3 个子任务,第 2 个任务的第 2 个子任务的编码 m 为 5.本文将这些子任务编码组成的集合看作是染色体的第 1 个基因,即 gene1,其长度为子任务的总个数 SN.

(2) 资源编码

对资源的编码实际上就是对物理机的编码,假设有 $W1$,$W2$ 和 $W3$ 这 3 个物理机,为了方便理解本文,顺序地将这 3 个物理机编码为 A,B,C.资源的总个数 WN=3.本文将资源编码组成的集合看作是染色体的第 2 个基因(gene2).一般情况下 $WN<SN$,即资源的总个数小于子任务的总个数.所以基因长度也为 SN,gene1 和 gene2 的长度要保持一致,其数值取 SN 和 WN 中较大的值.

(3) 染色体编码

本文中的染色体编码由两个基因组成,即 gene1 和 gene2,其中每个基因的长度为 SN 和 WN 中的较大值,一般情况下 $SN>WN$,所以这两个基因的长度都为 SN,染色体长度即为 $2\times SN$.如图 5 所示,此染色体有 t_1,t_2 两个任务,每个任务又可以划分为 3 个子任务,即 6 个子任务:1,2,3,4,5,6 和 3 个物理机:A,B,C 组成,每个基因的长度为 6,染色体的总长度为 12.

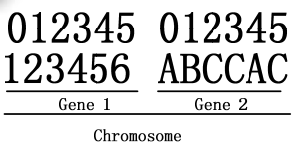


Fig.5 Code of chromosome

图 5 染色体编码

染色体的解码表示为 $A\{1,5\}$,$B\{2\}$,$C\{3,4,6\}$,即将编码为 1 和 5 的子任务交由编码为 A 的执行器执行.同理可知,编码为 2 的子任务被分配到了编码为 B 的执行器,子任务 3,4,6 交由了 C 执行器进行运算.

此染色体的解码也可以表示为 $t_1\{A,B,C\}$,$t_2\{C,A,C\}$.即表示 t_1 任务的 3 个子任务按顺序分由 A,B,C 这 3 个执行器进行运算.同理可知,t_2 任务的 3 个子任务也是按顺序地分配到了 C,A,C 执行器中.通过染色体的编码和解码,可以看到此种染色体的编码方式能够清楚明了地表示出任务的分配情况.

4.2 染色体有效性的判断

对 GEP 算法编码方式的改进,提高了对云任务编码的效率,使得用 GEP 算法求解云任务调度问题成为可能.但上述的编码方式会不可避免地产生一些无效的染色体,尤其是在种群的初始化阶段和对种群中的个体进行变异、转座等操作的时候.如图 5 所示,其染色体的解码表示为 $A\{1,5\}$,$B\{2\}$,$C\{3,4,6\}$,即物理机 C 中运行着编码为 3,4,6 的 3 个任务,如果任务 3,4,6 对物理资源的需求总和超出了物理机 C 所能提供的物理资源的总和,则编码“123456ABCCAC”的染色体为无效的染色体.无效染色体的存在会降低 GEP 算法的寻优效率,甚至会得出错误的任务调度方案.因此,本文对由 GEP 算法生成的染色体进行了有效性的判断,如果一个染色体中的所有物理机都能够同时满足公式(7)中的 3 个条件,则此染色体是有效染色体.由有效染色体构成的种群即为有效种群.

公式(7)表示参与云任务调度的物理机 i 所提供的 CPU,内存和硬盘资源要多于或等于分配到该物理机中任务对物理资源需求的总和,NT 为分配到物理机 i 中任务的个数.

$$\begin{cases} Cr_i-\sum_{k=1}^{NT}Ct_k\geqslant 0 \\ Mr_i-\sum_{k=1}^{NT}Mt_k\geqslant 0 \\ Sr_i-\sum_{k=1}^{NT}St_k\geqslant 0 \end{cases} \tag{7}$$

4.3 适应度函数的设计

本文综合考虑了云任务调度中使物理机利用率最高和整体调度能耗最低两个方面的因素,进一步推导出了基于加权求和形式的云任务调度适应度函数.如公式(8)所示,本文中的适应度函数可以分成两个部分,前一部分为所有参与云任务调度物理机的综合利用率的平均值,后一部分为所有参与调度物理机的总能耗的倒数,这

© 中国科学院软件研究所 http://www.jos.org.cn

两部分通过权重系数 a,b 进行加权求和,其中,$a+b=1$.a 和 b 的选择主要取决于云用户的偏好.由于通常情况下,适应度越高,云任务调度的效果越好,所以适应度函数的后半部分采用的是总能耗的倒数.经过反复的实验可知,总能耗的倒数和综合利用率的平均值不属于同一个数量级,即总能耗倒数的值要远小于综合利用率的均值.因此,本文在权重系数 b 的基础上又增加了参数 c,以使综合利用率和综合利用率的平均值有相同的数量级.

$$F_{fit}=a\times\left(\sum_{i=1}^{NK}l_z(i)\Big/NK\right)+b\times\left(c\Big/\sum_{i=1}^{NK}l_z(i)\times\varphi(i)\right) \tag{8}$$

公式(8)中的 NK 为参与任务调度的物理机数量,云任务调度系统中物理机的能耗是物理机运行一段时间的总能量消耗.物理机的能耗系数$\phi(i)$则是平均功耗,即若ΔE 是某个物理机在Δt 时间内所消耗的能量,则其能耗系数为$\phi(i)=\Delta E/\Delta t$.$\phi(i)$为物理机 i 的能耗系数,$l_z(i)$为物理机 i 的综合利用率,其可由公式(9)、公式(10)得出.

$$l_Z(i)=\alpha\times lC(i)+\beta\times lM(i)+\lambda\times lS(i) \tag{9}$$

$$lC(i)=\sum_{k=1}^{NT}Ct_k/Cr_i \tag{10}$$

公式(9)根据物理机的不同硬件产生的能耗不同,分别为 CPU、内存和硬盘设定了 3 个能耗权重参数,通过一个线性加权多项式的形式而得出.α,β,γ分别是 CPU、内存和硬盘的 3 个能耗权重系数,α,β,γ这 3 个能耗权重系数可以通过 CPU、内存和硬盘的功耗比得出.假设 W_c、W_m 和 W_s 分别是 CPU、内存和硬盘的功耗,则α,β,γ可以分别表示成$\alpha=W_c/(W_c+W_m+W_s)$,$\beta=W_m/(W_c+W_m+W_s)$和$\gamma=W_s/(W_c+W_m+W_s)$的形式.$lC(i)$,$lM(i)$和 $lS(i)$分别表示物理机 i 的 CPU、内存和硬盘的使用率.以 $C(i)$为例对物理机 i 的 CPU 使用率的得出进行说明,如公式(10)所示,$C(i)$为分配到物理机 i 中任务的 CPU 资源需求占物理机 i 所能提供的 CPU 资源的百分比.通过以上叙述可知,公式(8)即为本文中云任务调度的适应度函数.

4.4 基于资源改变量的局部云任务调度算法

基于批量模式和达尔文进化理论的动态云任务调度算法往往将批量到来的任务和所有的物理机进行全局的进化寻优操作,即对所有参与调度的任务和虚拟机进行编码和寻优操作.这种寻优方式能够得到云任务调度的最优解,但其计算较为复杂,运行时间成本较高.本文所提出的基于资源改变量的局部云任务调度算法希望能够通过牺牲一定寻优效果的方式,减少参加调度的物理机数量,从而降低云任务调度算法的计算复杂度和运行时间成本.

本文通过监控云系统中的物理资源改变量,即任务在物理机中的完成情况,对云系统中物理机的综合利用率进行更新操作,同时设定一个物理机综合利用率的阈值(lR),阈值(lR)的选取见第 5 节.在对到来的批量任务进行调度运算开始之前,令云系统中物理机的综合利用率与阈值(lR)进行对比,从而将物理机分成参与调度的物理机和不参与调度的物理机,即如果物理机 i 的综合利用率 $l_z(i)\geqslant lR$,则物理机 i 为不参见调度的物理机.否则,为参加调度的物理机.文中提出的基于资源改变量的局部云任务调度算法,减少了参与调度物理机的个数,降低了全局参加进化寻优的运行时间成本.

4.5 任务调度算法的基本步骤

本文首先对云中物理机的综合利用率进行监控,再依据综合利用率阈值(lR)将物理机分成参与调度和不参与调度两类,对任务和参与调度物理机进行编码,同时确定适应度函数,再利用改进的 GEP 算法对种群进行寻优,从而确定最优染色体.通过对该染色体的解码,可以得出任务与资源的对应关系,实现任务分配的优化.

算法基本步骤如下:

Step 1. 对云中物理机的综合利用率进行监控,同时依据综合利用率阈值(lR)将物理机分成参与调度和不参与调度两类;

Step 2. 以直接编码的方式对任务的每一个任务进行编码,记录所有子任务的总数 SN,确定 gene1 的形式;

Step 3. 对物理机进行编码,确定 gene2 的形式和长度,进而确定整个染色体的形式;

Step 4. 根据公式(8)~公式(10)确定适应度函数;

© 中国科学院软件研究所 http://www.jos.org.cn

Step 5. 确定种群的大小,随机生成第 1 代种群;

Step 6. 对随机生成种群中的每个染色体进行有效性判断.如果均为有效染色体,则顺序执行 Step 7.如果存在无效的染色体,则将该染色体进行重新随机生成,直至生成有效的染色体,进而生成第 1 代有效种群;

Step 7. 判断种群的适应度和迭代次数是否符合标准.如果符合,则对任务进行解码和任务分配.如果不符合,则根据改进的 GEP 对染色体进行变异和选择(具体过程如图 6 所示),生成新的种群;

Step 8. 将符合标准的染色体进行解码;

Step 9. 依照解码后的对应关系将任务分配到相应的执行器中运行.

Algorithm: Population evolutionary algorithm.
Task: Obtaining the population with the better fitness
Input: Initial population $\boldsymbol{P}(\mathbf{1}\times\boldsymbol{InN})$, iteration number ItN, individual number InN, mutation rate Mr, transposition rate Tr, recombination rate Rr, parameter i.
Output: The population with better fitness $\boldsymbol{P'}$.
Begin
Initialize: $\boldsymbol{U_0}=\boldsymbol{P}(\mathbf{1}\times\boldsymbol{InN})$ and i=0.
For i=0 **To** i<=ItN **By** 1 **Do**
  If 50<i<ItN **And** max($F_{fit}(\boldsymbol{P_i})$)==max($F_{fit}(\boldsymbol{P_{i-1}})$)==…==max($F_{fit}(\boldsymbol{P_{i-49}})$) **And** $F_{pan}(\boldsymbol{P_i})$==true
  Then % $F_{fit}(x)$,$F_{pan}(P_i)$ are shown in (8),(7) %
    $\boldsymbol{U_0}=\boldsymbol{P_i}$
    Break
  Else
    Update $\boldsymbol{U_0}$ by the elitist strategy and roulette strategy
    Update $\boldsymbol{U_0}$ by improved GEP algorithm as in section 3.3
    While $F_{pan}(\boldsymbol{U_0})$==false **Do**
      Update $\boldsymbol{U_0}$ by the elitist strategy and roulette strategy
      Update $\boldsymbol{U_0}$ by improved GEP algorithm as in section 3.3
    End While
    $\boldsymbol{P_i}=\boldsymbol{U_0}$
  End If
End For
$\boldsymbol{P'}=\boldsymbol{U_0}$
End

Fig.6 Pseudo-Code of population evolutionary

图 6 种群进化的伪代码

5 仿真实验与结果分析

本文采用 Matlab 对云环境下的任务调度算法进行仿真实验,实验的主要内容包括:基于改进 GEP 和资源改变量的局部云任务调度算法在云任务分配中的参数整定,全局 GEP 云任务调度算法和局部 GEP 云任务调度算法运行时间成本的比较,全局 GEP、局部 GEP 和普通遗传算法适应度和迭代次数的比较.

实验的设定:本文参考现实中应用到的物理机和服务器的类型,采用 Matlab 按照一定的规律在合理范围内分别随机生成 10,20,…,80 个物理机的资源提供向量,如公式(1)所示,以仿真拥有不同物理机个数的云系统.同时,以随机的方式生成每批次到来的任务向量,随机完成部分运行中的任务,如公式(3)所示,以仿真基于批量模式的动态云任务调度环境.

5.1 基于改进GEP和资源改变量的局部云任务调度算法的参数整定

表 1 列举了基于改进 GEP 和资源改变量的局部云任务调度算法在任务分配中的各项参数.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 1 Parameter list in the experiments

表 1 实验中各项参数表

	Items	Parameters
Parameters of improved GEP	Iteration number	200
	Individual number	10
	Mutation rate	0.4
	Transposition rate	0.3
	Recombination rate	0.3
Parameters of physical machine selection	Utilization rate (*lR*)	0.80
Parameters of fitness function	*a*,*b*	0.5
	c	10

由表 1 可知,本文实验中的各项参数主要分成了 3 个部分:改进 GEP 算法相关的参数、综合利用率阈值和适应度函数相关的参数.

(1) 改进 GEP 算法相关参数的整定

改进 GEP 算法相关参数的设定均是通过多次(至少 100 次)的反复实验得到.

现以突变概率参数为例,见表 2,第 1 列为不同的突变概率值,第 2 列为对每个突变概率值进行 100 次实验后的收敛迭代次数均值.通过观察图 7 可以发现,在其他参数不变的情况下,突变概率和平均迭代次数之间呈现出一种二次项系数大于 0 的二次函数关系,且使得平均迭代次数最低的突变概率值为 0.4.

Table 2 Mutation probability and average iterations table

表 2 突变概率和平均迭代次数表

Mutation probability	Average number of iterations
0.1	70
0.2	40
0.3	21
0.4	8
0.5	13
0.6	29
0.7	70

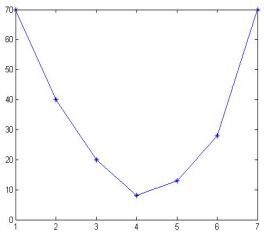


Fig.7 Mutation probability and average iterations figure

图 7 突变概率和平均迭代次数图

(2) 综合利用率阈值和适应度参数的整定

为了较为快捷地对综合利用率和适应度参数进行整定,本文选取一个由 10 个物理机构成的云系统反复(至少运行 20 次)对随机到来的任务进行分配.同时,为了方便理解,本文随机抽取出其中一种任务分配策略的物理资源利用率表,见表 3.

由表 3 所示和反复实验可知,物理机总能耗的倒数(1/19.0670=0.052631579)总是与综合利用率的平均值(0.811282798)相差 10 倍左右.所以本文中的参数 *c* 设定为 10,*a*,*b* 视为同等重要,均设定为 0.5.由表 3 可知,如果综合利用率阈值选取得过高,则将造成参与调度的物理机过多,使局部 GEP 调度和全局 GEP 调度在运行时间成本上的差距不明显.如果综合利用率的阈值选取得过低,则会对云任务调度算法的效果产生较大的影响,造成不必要的资源浪费,甚至会出现恶意拒绝服务的现象.经过 20 次反复实验的对比,本文中选取综合利用率阈值(*lR*)为 0.80.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 3 Example of ten physical machines allocation strategy

表 3 10 物理机的分配策略举例

Number of physical machine	CPU utilization	Storage utilization	Comprehensive utilization	Energy consumption
No.1	0.61	0.984 6	0.797 3	
No.2	0.998	0.282 333 333	0.640 166 667	
No.3	0.979 25	0.750 833 333	0.865 041 667	
No.4	0.638 857 143	0.990 166 667	0.814 511 905	
No.5	0.492 285 714	0.993 5	0.742 892 857	
No.6	0.665 833 333	0.991 2	0.828 516 667	19.067 0
No.7	0.949 571 429	0.999 625	0.974 598 214	
No.8	0.425 5	0.999 5	0.712 5	
No.9	0.979	0.513 6	0.746 3	
No.10	0.983	0.999	0.991	
Average utilization	0.772 129 762	0.850 435 833	0.811 282 798	

5.2 全局GEP云任务调度算法和本文中算法运行时间成本的比较

如表 4 和图 8 所示,全局 GEP 云任务调度算法和本文中所提出的局部 GEP 云任务调度算法在物理机个数相对较少的小型云系统中,在运行时间成本上的差距相对较小,但是,随着参与云任务调度物理机个数的不断增多,全局 GEP 调度算法的运行时间成本也不断地增加,而本文中局部 GEP 调度算法的运行时间成本基本保持不变.这是因为,随着物理机的不断地增加,全局 GEP 调度算法需要将所有的物理机进行编码,同时进化求解,其计算复杂度迅速增加.而本文中局部 GEP 调度算法的计算复杂度并不取决于云系统中物理机的个数,而是综合利用率小于综合利用率阈值(IR)的物理机个数.因此,本文中局部 GEP 云任务调度算法在物理机较多的云系统中会有比较好的表现.

Table 4 Comparison of time cost

表 4 时间运行成本对比

Physical machine number	GEP	Local GEP
10	10.225 433	4.425 594
20	31.848 64	15.390 419
30	36.586 446	22.940 431
40	50.924 288	30.440 591
50	131.685 275	20.363 416
60	75.324 709	16.565 887
70	177.904 282	24.481 523
80	233.732 355 7	14.737 677

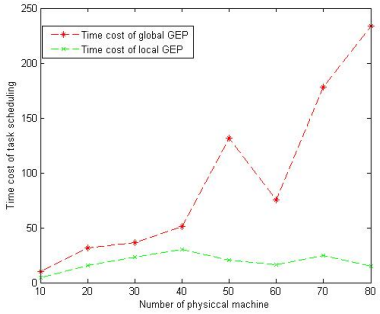


Fig.8 Comparison of time cost figure

图 8 时间运行成本对比图

5.3 全局GEP、局部GEP和普通遗传算法适应度和迭代次数的比较

如图 9 所示,本文中局部 GEP 任务调度算法因为需要调度的物理机个数较少,所以在迭代次数上有着一定的优势,但在适应度值上却略逊于全局 GEP 任务调度算法.这是因为,局部 GEP 算法在降低运行时间成本的同时,舍弃了对一些物理机的编码和调度,从而舍弃了一些可能出现的更好的任务调度策略.

© 中国科学院软件研究所 http://www.jos.org.cn

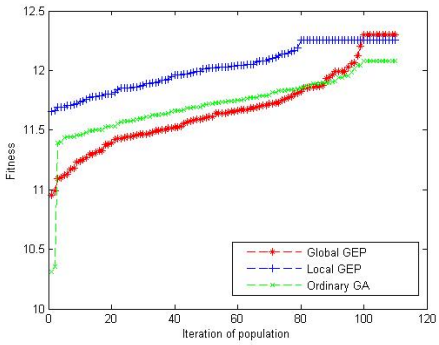


Fig.9 Comparison of the adaption degree and iteration time

图 9 适应度和迭代次数的对比图

从整体上看,本文提出的局部 GEP 云任务调度算法,在运行时间成本、收敛速度和调度策略最优等方面有着一定的优势和提高.

6 结 论

本文针对传统遗传算法收敛速度较慢且易于陷入局部最优解的现象,将 GEP 算法引入到了云计算的任务调度领域,对其编码和翻译方式做出了一定改进.同时,针对如何降低基于批量处理模式动态云任务调度算法中的计算量,降低运行时间成本的问题,提出了一种基于改进 GEP 和资源改变量的局部云任务调度算法.该算法通过设立综合利用率阈值,减少了参加调度物理机的个数,使该算法下的基于批量处理模式的动态云任务调度算法在降低计算复杂度、节约运行时间成本、提高物理机的使用效率等方面得到了一定的改善.

References:

[1] Well P, Grance T. The NIST Definition of Cloud Computing. NIST Special Pubulication, 2011.

[2] Kumar Al, Amandeep V. Workfilow scheduling algorithms in cloud environment—A survey. In: Engineering and Computational Sciences (RAECS). Chandigarh, 2014. 1−4. [doi: 10.1109/RAECS.2014.6799514]

[3] Lin C, Su WB, Meng K, Liu Q, Liu WD. Cloud computing security: Architecture, mechanism and modeling. Chinese Journal of Computers, 2013,36(9):1765−1784 (in Chinese with English abstract).

[4] Zhang Q, Zhani MF, Boutaba R. Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Trans. on Cloud Computing, 2014,2(1):14−28. [doi: 10.1109/TCC.2014.2306427]

[5] Huang Q. Power consumption of virtual machine live migration in clouds. In: Yuan DF, Cao MY, Wang CX, Huang H, eds. Proc. of the 3rd Int'l Conf. on Communications and Mobile Computing (CMC). Qingdao, 2011. 122−125. [doi: 10.1109/CMC.2011.62]

[6] Abdulrahman A, Abdulaziz A, Zhu MM, Che D. Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud. In: Akhgar B, Arabnia HR, eds. Proc. of the 2014 Int'l Conf. on Computational Science and Computational Intelligence (CSCI). Las Vegas, 2014. 69−74. [doi: 10.1109/CSCI.2014.97]

[7] Nejad MM, Mashsyekhy L, Grosu D. Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. IEEE Trans. on Parallel and Distributed Systems, 2014,(99):1−5. [doi: 10.1109/TPDS.2014.2308224]

[8] Mathew T, Sekaran KC, Jose J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In: Proc. of the Int'l Conf. on Advances in Computing, Communications and Informatics (ICACCI). New Delhi: IEEE, 2014. 658−664.

[9] Chen C. A new live virtual machine migration strategy. In: Proc. of the 2012 Int'l Symp. on Information Technology in Medicine and Education. Hokodate: IEEE, 2012. 173−176.

[10] Selvarani S, Sadhasivam GS. Improved cost-based algorithm for task scheduling in cloud computing. In: Proc. of the Computational Intelligence and Computing Research (ICCIC). Coimbatore: IEEE, 2010. 1−5.

© 中国科学院软件研究所 http://www.jos.org.cn

[11] Rakesh M, Sowmya KS. An hybrid bio-inspired task scheduling algorithm in cloud environment. In: Proc. of the Computing, Communication and Networking Technologies (ICCCNT). Hefei: IEEE, 2014. 1−7.

[12] Holland JH. Classifier systems and genetic algorithms. Artificial Intelligence, 1989, 235−282.

[13] Kun HE, Zhao Y, Huang WQ. A clustering and scheduling algorithm based on task duplication. Chinese Journal of Computers, 2008,31(5):733−740 (in Chinese with English abstract).

[14] Dorigo M. Ant systerm: Optimization by a colony of cooperating agents. IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, 1996,26(1):29−41. [doi: 10.1109/3477.484436]

[15] Dorigo M. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evolutionary Computation, 1997,1(1):53−66. [doi: 10.1109/4235.585892]

[16] Tawfeek MA, EI-Sisi A, Keshk AE, Torkey FA. Cloud task scheduling based on ant colony optimization. In: Fahmy HMA, El-Kharashi MW, El-Din AMB, Taher M, Salama CR, Safar M, eds. Proc. of the 8th Int'l Conf., Computer Engineering & Systems (ICCES). Cairo, 2013. 64−69. [doi: 10.1109/ICCES.2013.6707172]

[17] Sun WF, Qin ZQ, Li MC, Hu J, Chen YF. QIACO: An algorithm for grid task scheduling of multiple QoS dimensions. Acta Electronica Sinica, 2011,39(5):1115−1120 (in Chinese with English abstract).

[18] Koza JR. Gene Programming: On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.

[19] Ferreira C. Gene expression programming: A new adaptive algorithm for solving problems. Complex Systems, 2001,13(2):87−129.

[20] Zhu XM, Yang LT, Chen HK, Wang J, Yin S, Liu XC. Real-Time tasks oriented energy-aware scheduling in virtualized clouds. IEEE Trans. on Cloud Computing, 2014,2(2):168−180.

[21] Zhu XM, Chen HK, Yang LT, Yin S. Energy-Aware rolling-horizon scheduling for real-time tasks in virtualized cloud data centers. In: Proc. of the High Performance Computing and Communications & the 2013 IEEE Int'l Conf. on Embedded and Ubiquitous Computing (HPCC_EUC). Zhangjiajie: IEEE, 2013. 1119−1126.

[22] Shang YF, Li D, Xu MW. Energy-Aware routing in data center network. In: Proc. of the 1st ACM SIGCOMM Workshop on Green Networking. ACM, 2010. 1−8.

附中文参考文献:

[3] 林闯,苏文博,孟坤,刘渠,刘卫东.云计算安全:架构、机制与模型评价.计算机学报,2013,36(9):1765−1784.

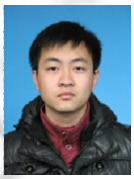
[13] 何琨,赵勇,黄文奇.基于任务复制的分簇与调度算法.计算机学报,2008,31(5):733−740.

[17] 孙伟峰,覃振权,李明楚,胡晶,陈媛芳.QIACO:一种多 QoS 约束网格任务调度算法.电子学报,2011,39(5):1115−1120.

李昆仑(1962－),男,博士,教授,CCF 会员,主要研究领域为通信与信息系统,计算机网络,智能信息处理.

宋健(1989－),男,硕士生,主要研究领域为云计算,智能信息处理.

王珺(1988－),男,硕士生,主要研究领域为云计算,智能信息处理.



董庆运(1990－),男,硕士生,主要研究领域为云计算,智能信息处理.

© 中国科学院软件研究所 http://www.jos.org.cn