Journal of Computer Applications
计算机应用,2014,34(2):351 -356
ISSN 1001-9081
CODEN JYIIDU
2014-02-10
http://www.joca.cn

文章编号:1001-9081(2014)02-0351-06
doi:10.11772/j.issn.1001-9081.2014.02.0351

云平台下动态任务调度人工免疫算法

杨 镜[1*],吴 磊[1],武德安[1],王晓敏[2],刘念伯[2]
(1.电子科技大学 数学科学学院,成都 611731; 2.电子科技大学 计算机科学与工程学院,成都 611731)
(*通信作者电子邮箱 yj547931@163.com)

摘 要:针对云计算领域的任务调度问题,提出了一种基于人工免疫(AI)理论的云计算平台动态任务调度算法。该算法首先利用排队论迅速、粗略地确定云计算平台保持稳态的条件,并为后面的计算提供基础数据;然后利用人工免疫理论中的免疫克隆选择算法,搜索出为集群中各节点上的不同虚拟机分配计算资源的近似最优配置;算法中还加入了适当的负载平衡处理,它使抗体基因更加优良。模拟实验结果表明,该调度算法能有效提高收敛速度和精度,快速搜索到合理配置,提高了集群资源利用率。

关键词:云计算;人工免疫;动态任务调度;虚拟机;资源配置;排队论
中图分类号:TP18 **文献标志码**:A

Artificial immune algorithm for dynamic task scheduling on cloud computing platform

YANG Jing[1*], WU Lei[1], WU De'an[1], WANG Xiaomin[2], LIU Nianbo[2]
(1. *School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu Sichuan* 611731, *China*;
2. *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 611731, *China*)

Abstract: In the field of cloud computing, it is a key problem that how task schedules. This paper presented an artificial immune algorithm for dynamic task scheduling on cloud computing platform. Firstly, the algorithm used the queuing theory to determine the conditions of cloud computing platform to maintain steady-state, and provided the basic data for the following algorithm. Then, this paper used the clone selection algorithm to search out the approximate optimal configuration which calculated resources for different virtual machines of different nodes in the cluster. Finally, proper load balancing processing algorithm joined with immune theory improved the antibody genes. The results of simulation experiment show that, this algorithm can effectively improve the convergence speed and accuracy, search reasonable allocation quickly and improve the cluster resource utilization.

Key words: cloud computing; Artificial Immune (AI); dynamic task scheduling; virtual machine; resource configuration; queuing theory

0 引言

云计算[1]是在并行分布式计算技术基础上的更高层次的"集中式"计算处理模式。存储管理、数据管理、虚拟化技术、大规模并行任务调度技术,以及面向数据密集型计算的并行编程模型等技术构成了支撑云计算的关键技术[2]。如何有效地进行任务调度来满足用户对于云计算服务的需求,同时使得云计算资源利用率和系统收益最大化,是当前云计算领域中一个重要而又复杂的研究课题。

目前,国内外已有许多针对不同计算环境的任务调度算法[3],如 Min-Min、Max-Min、快速贪吃算法(fast-greedy)、模拟退火算法、蚁群算法、遗传调度算法、混合遗传调度算法等[4]。Tang 等[5]给出了根据资源需求变化动态地将应用分配到集群节点的在线算法,但这些工作没有引入虚拟机技术,而是以应用作为放置单位。文献[6]先利用遗传算法处理虚拟机放置的组合优化问题,然后结合模糊逻辑进行多目标优化,包括总体资源利用率、能耗,但是只考虑了静态的虚拟机配置,忽视了虚拟机迁移的开销。

Bobroff 等[7]提出一种基于虚拟机动态迁移的方法来关闭无需启动的节点。该方法首先采用线性时序预测方法来预测虚拟机对资源的需求,并根据需求对虚拟机降序排列,然后采用首次满足(first-fit)背包算法将虚拟机部署在合适的节点上。然而,该研究仅仅考虑为每个应用分配一个虚拟机的场景,并不适用数据中心中一种虚拟机有多个副本的场景。Agrawal 等[8]提出的算法的前提是负载是事先确定的,默认的任务输入是基于应用的负载峰值,这样的结果是:在大部分时间,系统提供的资源都过多。

谭一鸣等[9]针对云计算系统运行时由于节点空闲和任务调度不合理而造成的能耗浪费问题,结合任务排队模型,提出了一种通过任务调度方式的能耗优化管理方法,在云计算

收稿日期:2013-07-16;**修回日期**:2013-09-28。 **基金项目**:国家自然科学基金资助项目(61103226,61272526);中央高校基本科研业务专项资金资助项目(ZYGX2010J111,ZYGX2010J074,ZYGX2011J102)。

作者简介:杨镜(1988 -),男,四川宜宾人,硕士研究生,主要研究方向:网络计算、分布式计算; 吴磊(1978 -),男,四川成都人,副教授,博士,主要研究方向:网络计算、分布式计算; 武德安(1972 -),男,四川成都人,副教授,博士,主要研究方向:无线网络、随机过程; 王晓敏(1978 -),女,四川成都人,副教授,博士,主要研究方向:无线自组网络; 刘念伯(1975 -),男,四川成都人,讲师,博士,主要研究方向:无线自组网络。

平台任务调度和能耗管理的研究方面做了有益的尝试。

李强等[10]提出了带应用服务级目标约束的虚拟机放置多目标优化遗传算法。该算法基于长期负载性能模型,采用组方式和三空间分割方法分别对染色体进行编码和译码,可根据染色体长度的变化设计交叉和变异遗传算子,能有效减少虚拟机迁移次数和物理节点的使用数量;但是也存在没有将资源控制与能耗控制结合起来的不足。

经过研究发现,使用人工智能的理论和方法可以找到更好的解,现在已经有了一些基于遗传算法、模拟退火算法、蚁群算法的调度算法,同时也有研究人员将多种调度算法结合使用,取得了较好的结果。混合遗传调度算法[11]是一种解决异构分布式环境下动态任务调度问题的有效方法。该算法加入了任务分组策略和适当的负载平衡处理,提高了算法的收敛速度。但是其主要的理论基础仍然没有摆脱传统的遗传算法天生的缺点:收敛速度缓慢、早熟收敛、缺乏爬山能力,其中早熟收敛最为严重[12]。

人工免疫(Artificial Immune, AI)系统是继神经网络、模糊逻辑和进化算法后人工智能的又一个研究热点[13-14],它主要借鉴生物免疫系统的信息处理机制发展新的算法。而云计算环境本身的软硬件环境很复杂,并且任务到达又时时刻刻都在变化。由于人工免疫系统提供噪声忍耐、无教师学习、自组织、记忆等进化学习机理,结合了分类器、神经网络、机器推理等系统的一些优点,因此具有提供新颖的解决问题方法的潜力[15]。所以,人工免疫系统在处理云平台下动态任务调度算法问题中具有巨大的潜力和优势。

针对云计算环境下系统高可用性、高可靠性要求,以及虚拟机动态配置的特点,本文将动态任务调度问题建模为对应虚拟机资源配置问题,是一个类似于装箱问题的NP-hard问题,并提出了一种启发式算法;同时以算法迭代速度、精度、资源有效利用率等作为性能评价指标,以实际系统的计算性能数据和任务流数据作为输入,验证了该算法在解决动态任务调度问题时的有效性。

本文针对动态任务调度问题的特点,引入了排队论对任务分配、执行前的状态进行快速建模、结构化处理;针对现有以任务高度值(粒度)来作为主要参考因素的任务调度算法所存在的严重不足,提出了任务—计算资源分配矩阵,用以描述任务(或虚拟机)与计算资源(节点)之间的关系,并结合人工免疫理论给出了运算模式。

1 基于人工免疫的云平台动态任务调度算法

1.1 任务排队

云计算平台运行时,大量任务涌入系统,会首先进行排队。通过对任务流排队现象的概率特性进行周期性监测、分析,确定任务到达的随机分布,并估计其参数值。

为便于研究,本文假设共有 m 类不同的任务,不同类型的任务(如任务 r_i)到达系统服从不同的概率分布,相同类型任务的到达间隔相互独立,且服从同一参数(λ_i)的负指数分布,λ_i 由系统统计监测数据得到,则可以得到所有 m 类任务的总到达率:

$$\lambda = \sum_{i=1}^{m} \lambda_i \tag{1}$$

本文统计单位时间内到达的任务 r_i 总数为 λ_i。

同时,云计算平台中的服务器配置彼此之间存在差异,所以要求所有服务器都有统一的计算能力是不现实的,并且这也是云计算异构分布式的主要特征,而对于虚拟机层也同样适用,因此,在对任务排队建模阶段,本文暂时将云计算平台看作只有唯一服务台,采用具有可变服务率的 M/M/1/∞ 型排队模型对云计算系统进行建模,用以求解在各任务 $r_i(i = 1,2,\cdots,m)$ 的到达率分别为 $\lambda_i(i = 1,2,\cdots,m)$ 的情况下,云计算系统需要为各类任务提供的最小服务率 μ_i ($i = 1,2,\cdots,m$) 分别是多少。

所以系统对于第 i 类任务 r_i 的服务强度可以设为:

$$E_{\rho i} = \lambda_i/\mu_i \tag{2}$$

此外,云计算系统对第 i 类任务的平均响应(逗留)时间(响应时间等于等待时间加上服务时间)设为:

$$E_{ti} = 1/(\mu_i - \lambda_i) \tag{3}$$

云计算系统对所有 m 类任务的期望平均响应时间设为:

$$E_{\text{time}} = \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} E_{ti} \tag{4}$$

为实现对系统的最优调控,本文将在系统中事先设定一些必要的参数 $T_i > 0$ ($i = 0,1,2,\cdots,m$),T_i 代表系统对于第 i 类任务的响应时间上限(其中 T_0 表示云计算系统对所有 m 类任务的平均响应时间上限),并有约定规则:

$$\text{Min } \mu_i \tag{5}$$

$$\text{s.t.} \quad 0 < E_{ti} < T_i;\ i = 1,2,\cdots,m;\ 0 < E_{\text{time}} < T_0$$

由式(5)结合式(3)、(4),本文便可以得到在理论上所需的云计算系统对任务 $r_i(i = 1,2,\cdots,m)$ 的最小服务率 $\mu_i(i = 1,2,\cdots,m)$,能够保证云计算系统期望平均响应时间 E_{time} 满足要求,系统分别对各类任务的平均响应时间 $E_{ti}(i = 1,2,\cdots,m)$ 也满足要求。

1.2 虚拟机资源配置框架

在云计算平台中,假设集群节点数为 C,系统共有 m 种任务类型 $r_i(i = 1,2,\cdots,m)$,每种任务对应一种虚拟机 $V_i(i = 1,2,\cdots,m)$,所以虚拟机类型总数也为 m。虚拟机 V_i 在第 j 个节点上的副本表示为 F_{ij},当 $F_{ij} = 0$ 时,表示 j 节点未启动 V_i;当 $F_{ij} = 1$ 时,表示 j 节点启动 V_i;当 $F_{ij} = 0(i = 1,2,\cdots,m)$ 时,表示节点 j 没有任何任务等待执行,处于待机状态。云计算系统将根据 $V_i(i = 1,2,\cdots,m)$ 对节点资源的需求变化情况动态调整节点运行数量和其上运行的虚拟机类型。

此外,本文还需要事先知道当节点 j 仅部署虚拟机 V_i 时,每秒处理(响应)任务 r_i 的最大次数 R_{ij}。然后本文假设节点 j 分配给虚拟机 V_i 的CPU资源为 W_{ij}(所占比例,用一位小数表示),故节点 j 提供给任务 r_i 的响应能力是每秒 U_{ij} 次,记:

$$U_{ij} = R_{ij} \times W_{ij} \tag{6}$$

由此,本文将以上述的参数为基础,对集群进行合理配置,在保证云计算系统服务质量的前提下,搜索出能提高云计算资源利用率的最优配置,集群配置的准则定义为:

$$\begin{cases} \text{Max } f \\ \text{Min } C_{\text{in}} \end{cases} \tag{7}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} W_{ij} \leqslant W_{\text{cpu}},\ \forall j \in C$$

其中:W_{ij} 表示节点 j 为虚拟机 V_i(对应任务为 r_i)分配的CPU资源,所以节点 j 为各类虚拟机分配到CPU资源总和应小于节点 j 的CPU资源上限 W_{cpu},当 $W_{ij} = 0(i = 1,2,\cdots,m)$ 时,表

示节点 j 处于待机状态;f 代表云计算系统所能提供的计算能力与任务所需计算能力的吻合度函数(即下文中的抗体—抗原亲和度函数);C_{in} 表示的是处于工作状态的节点数。

综上所述,虚拟机资源配置的目标是:在各节点分配出去的计算资源小于自身资源的情况下,使得云平台提供的计算能力与外界所需计算能力最接近,并且所需要的节点数越少越好。

1.3　配置算法

对于式(7),本文将基于人工免疫算法设计搜索过程。根据人工免疫的理论,把实际求解的问题及条件约束作为抗原,问题的解作为抗体[16],在本文中,每一种配置方案就是一个解,被视为抗体,根据其与抗原的亲和度和设定的抗体克隆规模进行克隆操作、免疫基因操作和克隆选择操作,获得新的抗体种群。

克隆操作　抗体—抗原亲和度越高,克隆的概率越大,另一方面,通过抗体—抗体亲和度调节,抑制过度竞争,保持抗体群的多样性;

免疫基因操作　产生变异的抗体,利用局部搜索增加了提高抗体—抗原亲和度的可能性;

克隆选择操作　通过局部择优,实现了种群的压缩。

经过若干次这样的迭代,最终筛选出最优配置。

1.3.1　抗体基因编码

每一个抗体代表一种配置方案,表明虚拟机运行位置和分配的计算资源数。本文将抗体编码为 $\boldsymbol{A}=\boldsymbol{\alpha}_1\boldsymbol{\alpha}_2\cdots\boldsymbol{\alpha}_C$,抗体编码由多个等位基因编码组成,$\boldsymbol{\alpha}_j$ 被称为等位基因,一个等位基因对应一个节点,每一个等位基因又由多个子段组成,一个子段对应一种虚拟机,具体说来分为 m 段,代表节点 j 上部署的虚拟机类型 i 和分配资源的比例 W_{ij},所以每个等位基因编码长 $l=m$。文中采用十进制编码方式,编码值 $a_{ij}=W_{ij}\times 10$。具体的编码规则为:

1) $\boldsymbol{\alpha}_j$ 长 m,每一位都是 0 到 10 之间的正整数,比如 $\boldsymbol{\alpha}_j$ 的第一位表示节点 j 分配给虚拟机 V_1 的 CPU 资源,如果 $\boldsymbol{\alpha}_j$ 的第一位为 5,则表示节点 j 分配50% 的CPU资源给虚拟机 V_1,其他可按此推理。

2) $\boldsymbol{\alpha}_j$ 中 m 个个位数之和不能超过 10,这意味着 j 节点分配给各虚拟机的资源之和不能超过 j 节点所能提供的有效计算资源,保证了节点的性能。

3) $\boldsymbol{\alpha}_j$ 中 m 个个位数尽量不等于 1,因为 1 代表节点 j 为某一类虚拟机分配了 10% 的计算资源,如果这种情况在集群中出现太多,那么计算资源就会过于分散,同时需要开启和管理的虚拟机又过多,最终导致系统性能的下降。

本文以 5 个节点,3 种虚拟机为例说明:设各节点分配给虚拟机的 CPU 资源比例及编码如表 1。5 个节点构成 5 个等位基因,每个等位基因有 3 个子段,节点 1 分配 100% 的资源给虚拟机 V_1,节点 2 分配给虚拟机 V_1、V_2、V_3 的资源分别为 20%、50%、30%,节点 3 分配 60% 的资源给虚拟机 V_3,其他节点处于待机状态。

抗体编码如下所示:

$$(\boldsymbol{\alpha}_1\quad\boldsymbol{\alpha}_2\quad\boldsymbol{\alpha}_3\quad\boldsymbol{\alpha}_4\quad\boldsymbol{\alpha}_5)=\begin{pmatrix}10&2&0&0&0\\0&5&0&0&0\\0&3&6&0&0\end{pmatrix}$$

抗体 $\boldsymbol{A}$ 编码为一个 $m\times C$ 维的非负整数矩阵,$\boldsymbol{A}=\{a_{ij}\}_{m\cdot C}$。

表 1　节点基于虚拟机的 CPU 资源分配比例

节点	V_1	V_2	V_3
1	100%	0	0
2	20%	50%	30%
3	0	0	60%
4	0	0	0
5	0	0	0

1.3.2　亲和度函数

本文中的亲和度函数分为抗体—抗体亲和度函数、抗体—抗原亲和度函数两类,它们也将对抗体在种群中的克隆、繁殖起到不同的作用。

抗体—抗体亲和度函数定义为:

$$\Theta_x=\min\{D_{xy}\}=\min\{\exp(\|\boldsymbol{A}_x-\boldsymbol{A}_y\|)\};\quad x\neq y,\ x,y=1,2,\cdots,t \tag{8}$$

定义抗体种群空间 $\boldsymbol{I}=\{\boldsymbol{A}\mid\boldsymbol{A}=\{a_{ij}\}_{m\cdot C},(a_{1j}+a_{2j}+\cdots+a_{mj})\leqslant 10,a_{ij}\geqslant 0\}$,$\boldsymbol{A}_x$、$\boldsymbol{A}_y$ 表示 $\boldsymbol{I}$ 中的两个不同的抗体,令 $\boldsymbol{A}_x=\{a_{ij}^{(x)}\}_{m\cdot C}$,$\boldsymbol{A}_y=\{a_{ij}^{(y)}\}_{m\cdot C}$;$D_{xy}$ 表示两抗体之间的差异,本文中由于采用十进制编码,所以 D_{xy} 取为欧几里得距离,并进行归一化处理。Θ_x 表示抗体 $\boldsymbol{A}_x$ 与其他抗体之间差异的最小值,其值越大,表示抗体 $\boldsymbol{A}_x$ 与种群中的其他个体之间的差异越大,则抗体 $\boldsymbol{A}_x$ 在克隆操作时,克隆规模越大,这样可以避免抗体种群陷入局部最优。Θ_x 将在计算抗体的克隆规模(式(13))时用到。

D_{xy} 的具体算法为:

$$D_{xy}=\frac{\sqrt{\sum_{j=1}^{C}\sum_{i=1}^{m}(a_{ij}^{x}-a_{ij}^{y})^2}}{\sum_{z\neq x}^{t}\sqrt{\sum_{j=1}^{C}\sum_{i=1}^{m}(a_{ij}^{x}-a_{ij}^{z})^2}} \tag{9}$$

抗体—抗原亲和度函数在本文中就是指云计算系统所能提供的计算能力与任务所需计算能力的吻合度函数,定义为:

$$f(A_x)=\frac{1}{1+e(\boldsymbol{A}_x)} \tag{10}$$

其中函数 $e(\boldsymbol{A}_x)$ 定义如下:

$$e(\boldsymbol{A}_x)=\frac{\sqrt{\sum_{i=1}^{m}\left(\sum_{j=1}^{C}U_{ij}-(1+\varepsilon)\cdot\mu_i\right)^2}}{\sqrt{\sum_{i=1}^{m}\mu_i^2}} \tag{11}$$

式(11)中 ε 为一正小数,可以按需求取适当值,ε 值能使最终的解更接近比期望值稍大的一侧,保证最终系统能够提供足够的计算能力,$f(\boldsymbol{A}_x)$ 就是抗体 $\boldsymbol{A}_x$ 与抗原之间的亲和度函数,取值在 0 到 1 之间,其值越大,抗原对其刺激作用越大,克隆规模越大;当 $f(\boldsymbol{A}_x)=1$ 时,表示抗体 $\boldsymbol{A}_x$ 就是最优解,抗体 $\boldsymbol{A}_x$ 所代表的虚拟机资源分配策略就是本文所需要的最佳策略。

1.4　免疫克隆选择

免疫克隆选择算子的主要操作步骤有克隆操作、免疫基因操作、克隆选择操作。操作的对象是该问题的具有一定规模 t 的抗体种群 $\boldsymbol{A}=\{\boldsymbol{A}_1\quad\boldsymbol{A}_2\quad\cdots\quad\boldsymbol{A}_t\}$,故本文假设满足约束条件的一组父解为 $\boldsymbol{A}^n=\{\boldsymbol{A}_1\quad\boldsymbol{A}_2\quad\cdots\quad\boldsymbol{A}_t\}$,$\boldsymbol{A}^n\in\boldsymbol{I}$。

1.4.1 克隆操作

在人工免疫中,克隆是指通过复制而形成一个群体,抗体复制的规模与抗体本身的亲和度有关。对抗体种群 $\boldsymbol{A}^n = \{\boldsymbol{A}_1 \quad \boldsymbol{A}_2 \quad \cdots \quad \boldsymbol{A}_t\}$ 的克隆操作 $T(\cdot)$ 可以表示为:

$$\begin{aligned} Y = T(\boldsymbol{A}^n) &= T(\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_t) = \\ & T(\boldsymbol{A}_1) + T(\boldsymbol{A}_2) + \cdots + T(\boldsymbol{A}_t) = \\ & \{A_{1,1}, A_{1,2}, \cdots, A_{1,p1}\} + \{A_{2,1}, A_{2,2}, \cdots, A_{2,p2}\} + \cdots + \\ & \{A_{t,1}, A_{t,2}, \cdots, A_{t,pt}\} \end{aligned} \tag{12}$$

其中:$A_{i,j} = A_i, j = 1,2,\cdots,p_i, i = 1,2,\cdots,t$。$p_i$ 表示抗体 $\boldsymbol{A}_i$ 的克隆规模,由于 p_i 与抗体 $\boldsymbol{A}_i$ 的亲和度有关,经过分析 p_i 分别与抗体—抗体亲和度函数、抗体—抗原亲和度函数之间的关系,得到如下关系式:

$$p_i = \mathrm{Int}\left(\frac{f(\boldsymbol{A}_i)}{\sum_{n=1}^{t} f(\boldsymbol{A}_n)} \cdot \boldsymbol{\Theta}_i \cdot P\right) \tag{13}$$

其中:Int(·)为上取整函数;P 可以是一个与克隆规模有关的设定参数,$P > t$。

1.4.2 免疫基因操作

本文中免疫基因操作将采取克隆重组、基因变异两步。

克隆重组是将同一抗体的不同克隆个体分开,并按一定的重组策略与其他抗体的克隆个体组合在一起,形成抗体种群的新子群。重组是在抗体层面对抗体种群的修改操作,它可以促进不同抗体之间的基因交流,有利于丰富种群多样性。重组操作可表示为 $T^C(\cdot)$。

$$\begin{aligned} Y' = T^C(Y) &= \{A'_{1,1}, A'_{1,2}, \cdots, A'_{1,p1}\} + \\ & \{A'_{2,1}, A'_{2,2}, \cdots, A'_{2,p2}\} + \cdots + \{A'_{t,1}, A'_{t,2}, \cdots, A'_{t,pt}\}; \\ & A'_{i,j} \in \boldsymbol{A}^n, i = 1,2,\cdots,t, j = 1,2,\cdots,p_i \end{aligned} \tag{14}$$

基因变异将是在基因编码的层面对抗体进行变异。设 $A'_{i,j} = \boldsymbol{\alpha}_1\boldsymbol{\alpha}_2\cdots\boldsymbol{\alpha}_C$ 是一个父解,本文按概率 p_m 对分量 $\boldsymbol{\alpha}_i$ 进行变异,已知编码中 $\boldsymbol{\alpha}_i$ 是一个 m 维列向量,本算法采用按位变异的策略 $T^B(\cdot)$,即对抗体 $A'_{i,j} = \boldsymbol{\alpha}_1\boldsymbol{\alpha}_2\cdots\boldsymbol{\alpha}_C$ 的第 i 位编码,用另一个 m 维列向量 $\boldsymbol{\alpha}'_i$ 替换。

$$\begin{aligned} Y'' = T^B(Y') &= \{A''_{1,1}, A''_{1,2}, \cdots, A''_{1,p1}\} + \\ & \{A''_{2,1}, A''_{2,2}, \cdots, A''_{2,p2}\} + \cdots + \{A''_{t,1}, A''_{t,2}, \cdots, A''_{t,pt}\}; \\ & A''_{i,j} \in \boldsymbol{I}, i = 1,2,\cdots,t, j = 1,2,\cdots,p_i \end{aligned} \tag{15}$$

则变异后的解为:$A''_{i,j} = \boldsymbol{\alpha}_1\cdots\boldsymbol{\alpha}_{i-1}\boldsymbol{\alpha}'_i\cdots\boldsymbol{\alpha}_C$。其中,$\boldsymbol{\alpha}'_i$ 也满足一些必要的性质。

定义 m 维列向量 $\boldsymbol{\alpha}'_i$ 中所有值取值为非负整数,且所有值之和也在 0 ~ 10。所以 $\boldsymbol{\alpha}'_i$ 的取值将有一个确定的范围,根据定义知 $\boldsymbol{\alpha}'_i$ 的取值空间为 $\boldsymbol{I}_m$,$\boldsymbol{I}_m$ 的空间规模为:

$$\frac{m^{11} - 1}{m - 1} \tag{16}$$

其中 m 为虚拟机种数。

1.4.3 克隆选择操作

克隆选择操作 $T^S(\cdot)$ 通过局部择优,从经过变异操作得到的抗体种群不同子群中分别提取亲和度相对良好的抗体,实现了种群的压缩和优胜劣汰。

$$\begin{aligned} A^{n+1} = T^S(Y'' \cup A^n) &= T^S(\{A_1, A''_{1,1}, A''_{1,2}, \cdots, A''_{1,p1}\} + \\ & \{A_2, A''_{2,1}, A''_{2,2}, \cdots, A''_{2,p2}\} + \cdots + \\ & \{A_t, A''_{t,1}, A''_{t,2}, \cdots, A''_{t,pt}\}) = \\ & T^S(A_1, A''_{1,1}, A''_{1,2}, \cdots, A''_{1,p1}, \cdots, A_2, A''_{2,1}, A''_{2,2}, \cdots, \\ & A''_{2,p2}, + \cdots +, A_t, A''_{t,1}, A''_{t,2}, \cdots, A''_{t,pt}) = \\ & \{A''_1, A''_2, \cdots, A''_t\} \end{aligned} \tag{17}$$

$\boldsymbol{A}^{n+1}$ 为下一代抗体种群。由于在进行克隆选择操作时,算法将父解也同时并入抗体种群中,这样就保证了抗体群中的最优解不会在算法迭代过程中变差,也保证了算法按照概率 1 收敛到最优种群集。

1.5 算法步骤

令 m 表示虚拟机的种类,C 表示节点数,n 表示迭代代数。

输入 $\{\mu_i\}(i = 1,2,\cdots,m)$ 表示前端接收到的对各类虚拟机的服务率请求;

输出 $\boldsymbol{A} = \{a_{ij}\}_{m \cdot C}$ 表示集群各节点分配给各类虚拟机的计算资源的信息编码。

第 1 步 设 $n = 0$,初始化抗体种群,$\boldsymbol{A}^n = \{\boldsymbol{A}_1 \quad \boldsymbol{A}_2 \quad \cdots \quad \boldsymbol{A}_t\}$。

第 2 步 计算各抗体 $\boldsymbol{A}_i$ 的抗体—抗原亲和力:$f(\boldsymbol{A}^n) = f(\{\boldsymbol{A}_1 \quad \boldsymbol{A}_2 \quad \cdots \quad \boldsymbol{A}_t\})$。

第 3 步 若 $\exists f(\boldsymbol{A}_i)(i = 1,2,\cdots,t)$ 达到理想值 f^*,或 n 达到最大迭代代数,则停止计算;否则继续。

第 4 步 克隆操作:$Y = T(\boldsymbol{A}^n)$;将抗体按照亲和度函数确定的克隆规模进行克隆,得到扩展后的种群 Y。

第 5 步 免疫基因操作:$Y' = T^C(Y) \rightarrow Y'' = T^B(Y')$;先后对种群进行重组、基因变异操作。

第 6 步 克隆选择操作:$\boldsymbol{A}^{n+1} = T^S(Y'' \cup \boldsymbol{A}^n)$;将经过免疫克隆选择算子后得到种群与初始种群合并,并分别从其子群中选取最优抗体,形成与初始种群(父代)$\boldsymbol{A}^n$ 规模相当的下一代子群 $\boldsymbol{A}^{n+1}$。

第 7 步 $n = n + 1$,返回第 2 步。

2 负载均衡处理

当按照上述克隆免疫选择算法计算得到问题的优质解 $\boldsymbol{A} = \{a_{ij}\}_{m \cdot C}$ 之后,通过解码得到集群各节点分配给各类虚拟机的计算资源比例,本文将以此为依据调整节点运行数量和其上运行的虚拟机类型,并按照规则向节点 j 上的虚拟机 V_i 调度任务 r_i,任务的规模为 λ_{ij}:

$$\lambda_{ij} = \frac{R_{ij} \times W_{ij} \times \lambda_i}{\mu_i} \tag{18}$$

由此,本文就大体完成了对于云计算集群的资源分配和任务调度工作。但还需要周期性地对云计算集群进行负载平衡处理,以平衡各节点的负载,保证系统性能和服务质量。负载平衡处理的策略为:

1)找出抗体中得到最多额外资源和资源最紧缺的两种虚拟机 $V_{\max}$、$V_{\min}$。

2)确定了获得最多额外资源的虚拟机 $V_{\max}$ 后,在所有部署了 $V_{\max}$ 的节点中选择对任务 $r_{\max}$ 处理能力最弱的一个 *Comp*。

3)如果在节点 *Comp* 上减少对 $V_{\max}$ 一个单位(1/10)的 CPU 资源分配,并且将这部分 CPU 资源转移给此节点上的虚拟机 $V_{\min}$ 后,抗体更加优良,则如上述调整此节点 *Comp* 的资源分配比例;如果节点 *Comp* 中没有开启相应的虚拟机 $V_{\min}$,则开启相应的虚拟机 $V_{\min}$;如果调整后的抗体不如调整前优良,则节点 *Comp* 上的资源分配比例不变。

4)循环 k 次执行这样的负载均衡处理。

此负载平衡处理属于辅助调节,在具体实行时不宜太频

繁,故本文仅在免疫循环的前几个循环、中间周期性间隔的几个循环和最后一次循环时使用:前几个循环利用负载均衡可以大大提高算法前期的收敛速度;并在每次负载平衡处理时,处理的循环次数也不宜太大。本文令循环次数为 k,k 值可以在实验中观察得到。

3　实验与结果分析

本文设计了基于人工免疫的云平台动态任务调度算法的仿真实验,并在相同实验条件下与混合遗传算法相比较,以验证该算法的正确性和有效性。实验模拟了有 20 个节点的计算机集群,其中 5 台 CPU 分别为 Intel Core2 Duo 2.83 GHz 的 Dell 主机、5 台 CPU 分别为 Intel Core2 Duo 2.33 GHz 的 Dell 主机和 10 台 CPU 为 AMD Athlon64 X2 3600 + 1.9 GHz 的 Lenovo 主机,并且由文献[17]取得这三种主机对 3 类典型应用:Browsing Mix(BM),Shopping Mix(SM)和 Ordering Mix(OM)的响应能力,见表 2。

表 2　节点的应用响应能力　　Req/s

主机	BM	SM	OM
Dell(2.83 GHz)	235	197	130
Dell(2.33 GHz)	190	163	91
Lenovo(1.9 GHz)	132	89	57

实验过程中的应用级任务到达数据由程序自动生成(每类任务到达率为 100 ~ 300 s^{-1})。每类任务 r_i 的期望响应时间上限 T_i 设为 0.05 s,共有 6 类不同的任务,对应 6 类不同的虚拟机,初始抗体种群大小设为 100 个。

从图 1 可以看出,在同等实验条件下,运用人工免疫算法的收敛速度高于运用混合遗传算法,虽然人工免疫算法的收敛速度出现了较大的波动(1 ~ 7 s),但是图中数据显示运用人工免疫算法收敛时间在 2 s 以下的实验占 60% 以上,而混合遗传算法仅为 30%,收敛时间在 3 s 以上的,人工免疫算法为 15%,混合遗传算法为 55%。

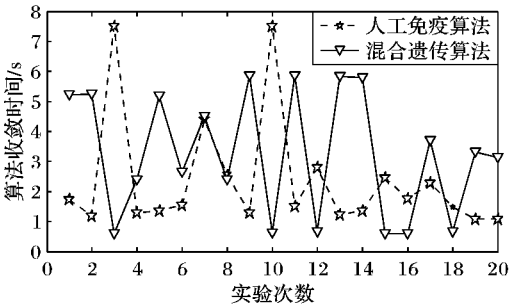

图 1　两种算法的收敛速度情况

图 2 表示,在同等的实验环境下,运用人工免疫算法或混合遗传算法对收敛精度的影响。从图中可以看出,人工免疫算法比混合遗传算法具有更高的收敛精度:混合遗传算法的收敛精度在 95% 左右,并且振动幅度较大(3%);而人工免疫算法的收敛精度基本能维持在 97% 以上,波动范围也缩小到一个百分点以内。所以人工免疫算法的收敛精度明显高于混合遗传算法。

在两种算法下集群所需要开启的物理节点数,如图 3 所示。从图中可以看出,前 7 次实验中,各类任务到达率均为每秒 300 个,第 8 次到 14 次实验中,各类任务到达率均为每秒 350 个,最后 6 次实验的各类任务达到率为每秒 400 个。可以看出利用人工免疫算法得到的配置中,所需物理节点数要少于混合遗传算法的结果。

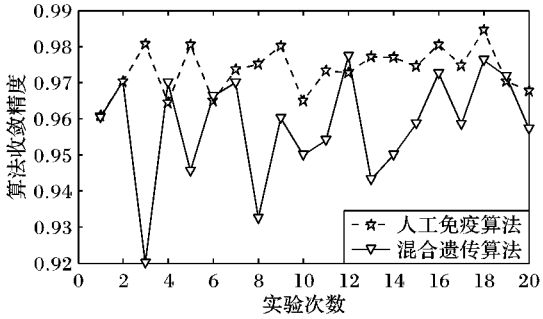

图 2　两种算法的收敛精度情况

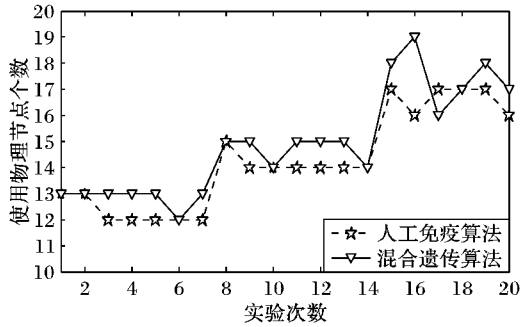

图 3　两种算法的所得配置方案中物理节点需求情况

从图 4 可以看出,当不进行负载均衡处理时,人工免疫算法收敛速度很慢,并且很不稳定,收敛所需循环次数在 10 至 40 之间大幅波动,这种情况显然无法满足云环境下实时动态调度的需求;而当算法加入了负载均衡之后,算法的收敛速度大大提高,所需循环次数基本维持在 6 次以内,而且也更加稳定。

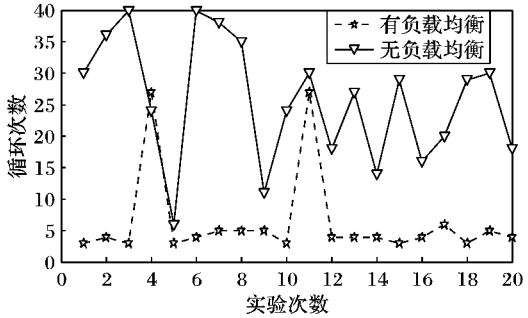

图 4　负载均衡对人工免疫算法收敛速度的影响

图 5 表示在有 20 个节点、6 种虚拟机的情况下,负载均衡对人工免疫算法收敛精度的影响。从图中可以看出,虽然人工免疫算法的收敛精度已经基本达到 95% 以上,但是在加入了负载均衡处理之后,还是明显提高了该算法的收敛精度,基本能维持在 97% 以上,波动范围也减小到一个百分点以内。所以负载均衡处理是优化人工免疫算法的有利工具,起到了加快收敛和提高收敛精度的一举两得的作用。

此外,通过图 6 可以看到,本算法的理论有效资源(这里指 CPU)利用率能达到 90% 以上,这对于当下数据中心资源利用率只有 50% 左右的现状来说,具有重要的应用与研究价值。

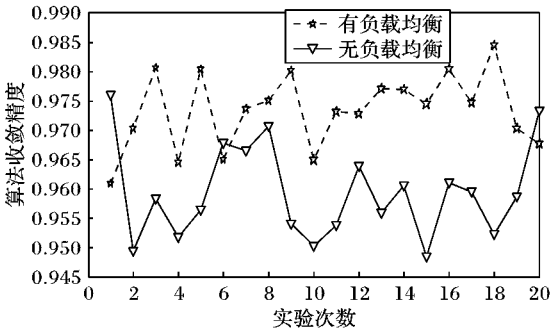

图5 负载均衡对人工免疫算法收敛精度的影响

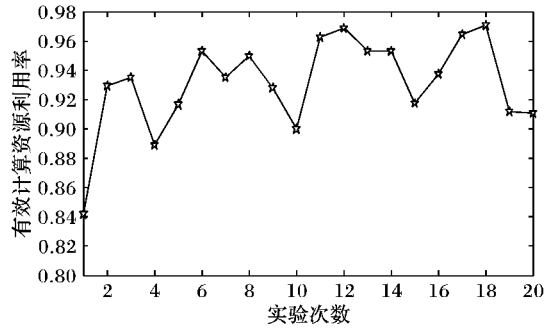

图6 基于人工免疫的任务调度算法理论资源利用率

4 结语

本文在研究了近几年来的各种任务调度算法之后,提出了基于人工免疫的云平台动态任务调度算法。此算法首先利用排队论对任务进行排队处理,然后用人工免疫理论中的免疫克隆选择策略对集群中的计算资源进行合理配置,再利用负载均衡调整抗体基因,使得集群资源的配置更加满足任务处理的需要。实验结果表明,本算法对于处理云平台的动态任务调度问题是正确可行的。

需要注意的是:1)本文中的算法是针对元任务,但现实集群还有着大量存在依赖关系的任务,这导致集群中虚拟机的部署也存在着依赖关系,在这种情况下,调度算法就会变得更加复杂,这是本文下一步需要改进的重点;2)本文所研究算法是对于任务到达情况变化之后,通过监测得到环境变化数据,才能用以实施调度策略,如何在变化之前通过分析历史数据来预测任务下一步的到达情况,也是本文接下来所要关注和改进的方向。

参考文献:

[1] CHEN K, ZHENG W. Cloud computing: System instances and current research[J]. Journal of Software, 2009, 20(5): 1337 - 1348. (陈康,郑纬民. 云计算: 系统实例与研究现状[J]. 软件学报, 2009, 20(5): 1337 - 1348.)

[2] GUO M, XU C. The programming model and adaptive management of cloud computing[J]. Communications of China Computer Federation, 2012, 7(7): 26 - 33. (过敏意, 须成忠. 云计算的编程模型与自适应资源管理[J]. 中国计算机学会通讯, 2012, 7(7): 26 - 33.)

[3] CHEN Z G, YANG B. Task scheduling based on multidimensional performance clustering of grid service resources[J]. Journal of Software, 2009, 20(10): 2766 - 2775. (陈志刚, 杨博. 网格服务资源多维性能聚类任务调度[J]. 软件学报, 2009, 20(10): 2766 - 2775.)

[4] DI MARTINO V. Scheduling in a grid computing environment using genetic algorithms[C]// Proceedings of the 16th and Distributed Processing Symposium. Washington, DC: IEEE Computer Society, 2002: 297.

[5] TANG C, STEINDER M, SPREITZER M, *et al.* A scalable application placement controller for enterprise data centers[C]// Proceedings of the 16th International Conference on World Wide Web. New York: ACM, 2007: 331 - 340.

[6] XU J, FORTES J. Multi-objective virtual machine placement in virtualized data environments[C]// Proceedings of 2010 IEEE/ACM International Conference on Green Computing and Communications. Washington, DC: IEEE Computer Society, 2010: 179 - 188.

[7] BOBROFF N, KOCHUT A, BEATY K. Dynamic placement of virtual machines for managing SLA violations[C]// Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management. Piscataway: IEEE, 2007: 119 - 128.

[8] AGRAWAL S, BOSE S K, SUNDARRAJAN S. Grouping genetic algorithm for solving the server consolidation with conflicts[C]// Proceedings of the 1st ACM/SIGEVO Summit Genetic and Evolutionary Computation. New York: ACM, 2009: 1 - 8.

[9] TAN Y, ZENG G, WANG W. Policy of energy optimal management for cloud computing platform with stochastic tasks[J]. Journal of Software, 2012, 23(2): 266 - 278. (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.)

[10] LI Q, HAO Q, XIAO L, *et al.* Adaptive management and multi-objective optimization for virtual machine placement in cloud computing[J]. Chinese Journal of Computers, 2011, 34(12): 2253 - 2264. (李强, 郝沁汾, 肖利民, 等. 云计算中虚拟机放置的自适应管理与多目标优化[J]. 计算机学报, 2011, 34(12): 2253 - 2264.)

[11] FAN M, HU W. Research of hybrid scheduling strategy in heterogeneous environment based on improved genetic algorithm[J]. Microelectronics & Computer, 2010, 27(8): 119 - 123. (范敏, 胡伟. 基于改进遗传算法的异构环境混合调度策略研究[J]. 微电子学与计算机, 2010, 27(8): 119 - 123.)

[12] SHIVLE S, CASTAIN R, SIEGEL H J, *et al.* Static mapping of subtasks in a heterogeneous Ad Hoc grid environment[C]// Proceedings of the 13th International Parallel and Distributed Processing Symposium. Piscataway: IEEE, 2004: 26 - 30.

[13] LIU F, YANG H. A clone based multicast algorithm with adjustable parameter[J]. Journal of Software, 2005, 16(1): 145 - 150. (刘芳, 杨海潮. 参数可调的克隆多播路由算法[J]. 软件学报, 2005, 16(1): 145 - 150.)

[14] GONG M, DU H, JIAO L. Optimal approximation of linear systems by artificial immune response[J]. Science in China: Series F, 2006, 49(1): 63 - 79.

[15] JIAO L C, DU H F. Artificial immune system: Progress and prospect[J]. Acta Electronica Sinica, 2003, 31(10): 1540 - 1548.

[16] de CASTRO L N, von ZUBEN F J. Learning and optimization using the clonal selection principle[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(3): 239 - 251.

[17] MI H, WANG H, YIN G, *et al.* Resource on-demand reconfiguration method for virtualized data centers[J]. Journal of Software, 2011, 22(9): 2193 - 2205. (米海波, 王怀民, 尹刚, 等. 一种面向虚拟化数字中心资源按需重配置方法[J]. 软件学报, 2011, 22(9): 2193 - 2205.)