基于多目标演化算法的云计算虚拟机分配策略研究

艾浩军[1] 龚素文[2] 袁远明[1]
(武汉大学计算机学院 武汉 430072)[1] (九江职业技术学院 九江 332007)[2]

摘 要 分析云计算虚拟机资源模型,针对模型中虚拟机与物理机的映射关系以及虚拟机多资源因子、多优化目标的特点,将虚拟机分配问题转化成多维装箱问题,引入多目标演化算法进行求解。算法设计了基于组的虚拟机分配链式编码和染色体评估函数,并根据编码设计了两种交叉算子和智能变异算子,通过引入基于超体积的种群更新机制,设计了基于SMS-EMOA的云计算虚拟机分配算法。为验证SMS-EMOA的性能,分别使用优先匹配启发式算法、基于物理节点数量的单目标简单遗传算法、SMS-EMOA进行了模拟。实验结果表明,基于SMS-EMOA的虚拟机分配算法在性能上更优。

关键词 云计算,虚拟机分配,多目标演化算法
中图法分类号 TP393 **文献标识码** A

Research of Cloud Computing Virtual Machine Allocated Strategy on Multi-objective Evolutionary Algorithm

AI Hao-jun[1] GONG Su-wen[2] YUAN Yuan-ming[1]
(School of Computer, Wuhan University, Wuhan 430072, China)[1] (Jiujiang Vocational & Technical College, Jiujiang 332007, China)[2]

Abstract The thesis analyzed cloud computing virtual machine resource model, and aiming at the mapping between virtual machines and physical machines in the model, and the characteristics of virtual machine multi-resource factor and multi-objective optimization, translated the virtual machine allocation problem into multi-dimensional packing problem, introduced multi-objective evolutionary algorithm to solve problem. Firstly, the algorithm designs the growing chain coding and chromosome evaluation function for virtual machine distribution problem, and according to the Encoding, designs two crossover operators and a smart mutation operator, finally, introduces the Hypervolume Measure as population selection Criterion, and designs cloud virtual machine allocation algorithm based on SMS-EMOA. In order to verify the performance of SMS-EMOA, the simulation experiment was made using PMH, CGA, SMS-EMOA, and the experimental results show that SMS-EMOA is better in virtual machine allocation performance.

Keywords Cloud computing, Virtual machine allocated, Multi-objective evolutionary algorithm

1 引言

云计算作为一种新的基础架构和服务模式,目前已成为各国政府、各科研机构和企业界关注的焦点,并成为新一代信息技术领域的研究热点[1]。随着“智慧城市”在全球范围内的兴起,国内各大城市也纷纷进入了智慧城市建设热潮。智慧城市信息系统作为支撑智慧城市运转的平台,涵括了城市的各个方面,是一个应用全面、大数据、大计算量、实时快速的系统,对作为智慧城市信息系统核心枢纽的城市数据中心的服务能力提出了更高要求。基于云技术构建城市云数据中心,能提升智慧城市信息系统的高可用性、高性能以及数据中心硬件资源的利用率,很好地满足智慧城市信息系统的高需求。

云数据中心通过虚拟化技术将计算资源、存储资源和网络资源统一构造成虚拟资源池,使用虚拟资源管理技术实现云计算资源按需自动分配与调度、动态扩展、自动部署;用户按需获取资源。而如何对云计算资源进行合理分配与高效调度,保证云数据中心的均衡负载性能及低能耗已成为当前云计算研究的难点。因此,研究负载均衡、低能耗的云计算资源分配与动态调度算法具有重要的理论意义和应用价值。

2 云计算虚拟机调度模型

虚拟化资源是通过虚拟化技术对物理资源进行抽象后的资源。由于城市数据中心硬件设备间存在差异且兼容性差,进行统一的物理资源管理难以实现。而通过对资源的抽象和虚拟化,屏蔽物理资源间的差异,搭建云计算环境,为实现资源的统一管理提供了科学可行的方案。资源虚拟化结构如图1所示。

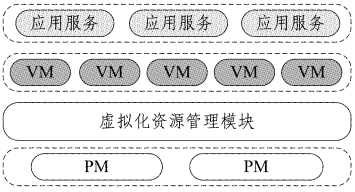

图1 资源虚拟化结构示意图

到稿日期:2013-07-17 返修日期:2013-12-02 本文受国家科技支撑计划(2012BAH35B03)资助。
艾浩军(1972—),男,副教授,主要研究方向为音视频信息处理与云计算应用系统;龚素文(1975—),副教授,主要研究方向为软件工程与计算机网络控制,E-mail:gsw4036@126.com;袁远明(1973—),博士,主要研究方向为智慧城市信息系统应用。

虚拟资源层向应用服务层提供物理资源映射，消除应用服务与硬件资源间的耦合，实现应用服务的快速部署与切换。城市数据中心运用虚拟化技术建立统一的虚拟资源池来实现对大规模基础资源的有效统一管理。依据城市信息系统的应用服务需求，通过虚拟化资源管理模块从动态可扩展的统一虚拟资源池中申请虚拟机资源，虚拟机资源主要包含计算资源、存储资源和网络资源。

虚拟化技术通过将底层的处理器、内存、网络带宽等硬件资源进行抽象，使得底层的差异性和兼容性对上层应用透明，从而实现对底层各种硬件资源的统一管理。虚拟机资源调度有两层：第一层是应用服务的作业任务与虚拟机资源的匹配问题；第二层是虚拟机与物理节点间的映射问题，将虚拟机分配在合适的物理节点上或进行动态迁移。本文注重虚拟机在物理机上的分配问题研究。

虚拟机到物理节点的映射关系为多对一或一对一，映射关系如图 2 所示。

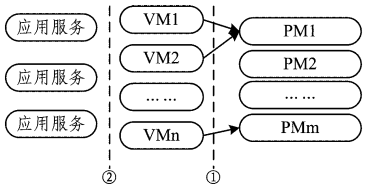

图 2　虚拟机与物理机的映射关系示意图

目前，虚拟器件技术[2]成为快捷部署虚拟机的重要技术，并被广泛应用于云计算。虚拟器件是一个包含预安装、预配置的操作系统、中间件和应用程序的最小化的虚拟机，可实现以虚拟机为基础单元的调度分配与重配置。最小化虚拟机有 3 个模板：CPU 消耗型、内存消耗型、网络带宽消耗型。在多个虚拟机创建后，如何将这些虚拟机首次部署到多个物理机上，实现物理机集群负载均衡效果最佳且节能效果最好，已成为云计算资源调度研究的重点。

近年来，国内外学者对云计算虚拟机资源的分配与调度进行了研究。杨星等基于性能向量完成云环境下的虚拟机部署工作[3]；华夏渝等利用蚁群算法完成云计算资源的分配工作[4]；李强等通过多目标遗传算法完成云计算虚拟机的放置与自适应管理工作[5]；徐星等采用动力学演化算法完成云任务与虚拟机分配工作[6]。从研究情况来看，智能优化算法已被逐步用于云计算资源的调度优化研究，在云计算资源调度领域具有很大的研究潜力。因此，本文针对虚拟机多资源因子、多优化目标的特点，采用模型转换基于多目标演化算法来研究云计算虚拟机的分配问题，使服务器整体负载均衡与低能耗最优。

3　多目标虚拟机分配演化算法

3.1　问题描述

云计算虚拟机分配是多目标的向量装箱问题，同时也是多目标组合优化问题。我们可将多目标虚拟机分配问题看作一个多维装箱问题，每个物理节点的可用资源（箱子）是一个 d 维向量，每一维表示某种资源（CPU、内存、网络带宽等）。每个虚拟机的资源（物品）也是一个 d 维向量。目标是将多个虚拟机（物品）放入多个物理节点（箱子），并使得占用物理节点的个数最少且负载方差最小。多目标虚拟机分配问题的描述如下：

$$f_{PN}=\min \sum_j C_j \text{ 且 } f_{LB}=\min \frac{\sum D\xi_i}{d}$$

其中，f_{PN} 表示占用物理节点的个数，f_{LB} 表示服务器集群的均衡负载方差。

$$C_j=\begin{cases}1\\0\end{cases}\text{，物理节点 } j \text{ 使用时为 1，否则为 0；}$$

$D\xi_i$ 表示 i 维方差，d 表示总维数，$D\xi_i=\frac{\sum_n (p_{ij}-\bar{p}_i)^2}{n}$，$n$ 表示物理节点数，$\bar{p}_i$ 表示所有物理节点第 i 维性能特征的平均值，性能特征为规范化值，即等于物理节点中第 i 维资源的剩余分配量除以 i 维总资源量，p_{ij} 为物理节点 j 的第 i 维性能特征。

$$\sum_i V_i^{cpu} * x_{i\to j} \leqslant P_j^{cpu}$$
$$\sum_i V_{ij}^{mem} * x_j \leqslant P_j^{mem}$$
$$\sum_i V_{ij}^{I/O} * x_j \leqslant P_j^{I/O}$$

其中，$x_{i\to j}=\begin{cases}1\\0\end{cases}$，虚拟机 i 分配到物理节点 j 时 $x_{i\to j}$ 为 1，否则为 0 。VM_i^{cpu}、VM_i^{mem}、$VM_i^{I/O}$ 为虚拟机 i 的 CPU、内存、网络带宽资源，P_j^{cpu}、P_j^{mem}、$P_j^{I/O}$ 为物理节点 j 的 CPU、内存、网络带宽资源。

应用系统 P 可能部署在 cm 个虚拟机上，cm 个虚拟机 VM 提供给应用系统 P 的性能指标应满足每个应用系统 P 的服务等级协议（Service Level Agreement，SLA），则

$$\sum_{cm} VM_{ip} \geqslant SLA_p \tag{1}$$

式(1)用于资源动态调度，通过资源监控器监测性能，当超载时，通过部署新的虚拟机来分担负载，当访问下降时，在满足式(1)的条件下缩减虚拟机。

3.2　染色体编码与评估函数

1. 染色体编码

编码方式是决定演化算法的搜索效率和效果的首要因素，编码体现问题解到染色体的映射关系。在云计算资源的虚拟机分配问题中，将虚拟机分配到物理节点的解的编码作为染色体。目前，装箱问题的染色体编码有基于箱子、基于物品、基于组 3 种编码方式[7]。前两种编码方式是针对单项物品进行的，而装箱问题的目标函数依赖于物品组。因此，论文依据基于组的编码方式，针对组编码方式存在的组合冗余问题[8]，提出基于组的链式编码方法。

将 N 个虚拟机分配到 M 个物理节点上，通常 $N \gg M$；随机生成一个包含 N 个虚拟机编号的序列，应用优先匹配启发式算法将随机序列依次放入物理节点中，得到染色体编码。其中优先匹配启发式算法描述为：从所有已使用的物理节点中选取第一个物理节点，如果该物理节点的三维（CPU、内存、网络带宽）资源量能满足第一个物理节点的三维资源要求，则将第一个虚拟机放入该物理节点，否则与后面已使用的物理节点依次匹配，直到寻找到满足要求的物理节点（假设 M 个物理节点能满足 N 个虚拟机的分配需求）；若在所有已使用的物理节点中未找到合适的物理节点，则从所有未使用的物理节点中选取第一个物理节点，用于该虚拟机的分配（初始化时，所有物理节点为未使用的物理节点）。按照上述描述依次

将序列中的其他虚拟机按照上述算法放入物理节点中。其伪代码如下。

```
Alg1:For i=0 to N
        Flag = 0
        For j=0 to M
            If Pj 已使用 then
                If Rpj (c, m, b) > Rvi (c, m, b)
                    虚拟机VMi放入物理节点Pj
                    Flag = 1; Break;
                Else
                    j++;
                End if
            End if
        Next
        If flag = 0 then
            虚拟机VMi放入第一个未使用的物理节点P中
        End if
        i++
    Next
```

以 9 个虚拟机分配到 5 个物理节点为例对编码进行阐述，基于组的链式编码如图 3 所示。

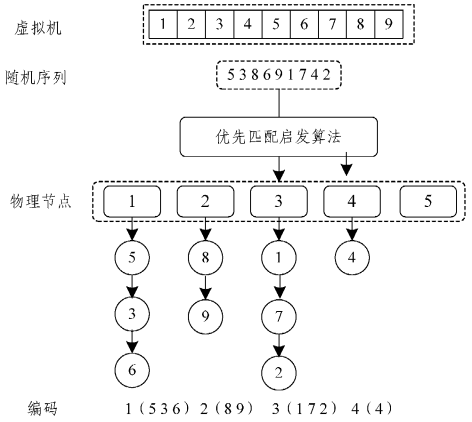

图 3　基于组的虚拟机分配链式编码

2. 评估函数

均衡负载保证应用系统高性能、物理节点低使用率两个目标，通过两个适应度函数(见式(2))来评估个体。第一个是使用物理节点的使用个数来评估染色体的耗能程度，物理节点占用越少，耗能越小；第二个是按照已使用物理节点的负载方差来评估染色体负载性能，方差值越小，负载性能越好。

$$Fitness=\{\min\sum_j C_j,\min\frac{\sum_d D\xi_i}{d}\} \tag{2}$$

3.3　演化算子

1. 交叉算子

针对虚拟机分配到物理节点的问题，基于组的链式编码包含两个部分：虚拟机编码和物理节点编码。用组及内部链式结构表示染色体的基因，组中虚拟机个数不是固定的，所以交叉涉及长度可变的染色体。

本文提出两种交叉因子：最低索引-最大长度交叉、改进单点交叉。

(1)最低索引-最大长度交叉的步骤

①两个父个体交叉产生一个子个体；

②比较两个父个体的最小索引的组中虚拟机分配链式长度，选取较长的组遗传给子个体；

③删除两个父个体中已遗传到子个体中的各虚拟机编号；

④循环步骤②，直到所有虚拟机编号都被遗传给子个体。

假如父个体 1{(1,3,6),(2,4),(5)}和父个体 2{(1,2),(3,4,5,6)}被选取进行交叉。首先父个体 1 的最小索引组(1,3,6)的链式长度相比父个体 2 的最小索引组(1,2)的链式长度要长(如果长度相等，则比较组对应的物理节点的可分配资源量，选取值小的组)，保留并传给子个体；然后将两父个体中的虚拟机编号(1,3,6)删除，得到父个体 1{(2,4),(5)}和父个体 2{(2),(4,5)}。再次比较父个体 1 的最小索引组(2,4)与父个体 2 的最小索引组(2)的链式长度大小，并将较大的组传给子个体；然后将两父个体中虚拟机编号(2,4)删除，得到父个体 1{(5)}和父个体 2{(5)}。最后(5)被传给子个体。删除两父个体的虚拟机编号(5)使得父个体序列为空，则交叉完成。

(2)改进式单点交叉的步骤

①随机选取父个体 1 中的一个组，代替父个体 2 中对应的组；

②针对交叉后的父个体 2，将其中重复的虚拟机对应的物理节点从使用物理节点集合中删除，加入未使用物理节点集合，其上的虚拟机也随之删除；

③在保留父个体 2 的虚拟机链式结构基础上生成未分配的虚拟机序列；对未分配的虚拟机序列应用优先匹配启发式算法重新分配，得到新的子个体；

④同理，随机选取父个体 2 中的一个组代替父个体 1 中对应的组，按照步骤②－④生成子个体 2。

改进式单点交叉的实现过程如图 4 所示。

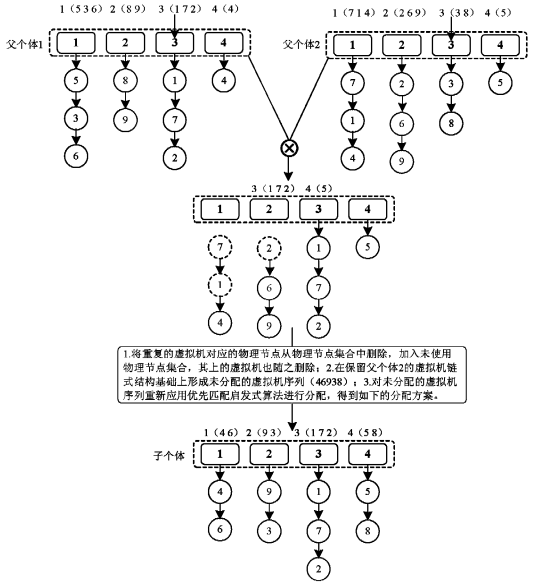

图 4　改进式单点交叉

2. 变异算子

变异操作采用一种智能突变因子。从染色体中随机选取 K 个物理节点(其中 $K<M/2$)，将 K 个物理节点从使用物理节点集合中删除后，将其加入未使用物理节点的集合中，物理节点上部署的虚拟机也随之被删除；然后在保留变异个体的虚拟机链式结构基础上形成未分配的虚拟机序列，最后对未分配的虚拟机序列按照优先分配启发式算法进行重新分配。其中参数 K 的取值结合变异率来给定，参数 K 的取值影响多目标演化算法的性能，通过实验来确定参数 K 的取值大小。

变异操作实现过程如图 5 所示。

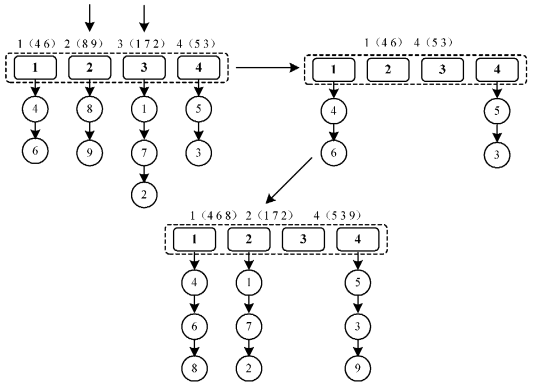

图 5 K 点变异

3.4 基于超体积的种群更新机制

对当前种群进行交叉变异后生成新一代子种群，将当前种群与子种群叠加后进行快速非支配排序，根据个体间支配关系、拥挤距离以及超体积[9]贡献来产生下一代种群进行演化。具体的种群更新机制如下：

(1)对叠加后的大种群按照支配关系进行分级排序，假设等级数为 v，则非支配集合有 $L_1, L_2, \cdots, L_v$。

(2)假设种群规模为 N，若 $Count(L_1)$ 为 N，则将 L_1 集合中的个体作为下一代种群。

(3)若 $Count(L_1)$ 大于 N，则计算 L_1 集合中除两个目标极值外的所有个体的超体积贡献值，选择贡献值最大的 $N-2$ 个个体与两个目标极值的个体作为下一代种群。

(4)若 $Count(L_1)$ 小于 N，则将 L_1 中所有个体加入下一代种群，再将 $L_2, L_3, \cdots$ 中的个体加入，直到下一代种群规模达到 N。若 L_i 加入后下一代种群规模大于 N，则计算 L_i 中个体的拥挤距离，选择拥挤距离大的个体加入到下一代种群，使下一代种群规模为 N。

假设一个排序前沿 $R_V=\{s_1, s_2, \cdots, s_{|R_v|}\}$，其中 v 为前沿等级，$|R_v|$ 为该前沿的个体总数，Δ_φ 为个体的超体积贡献，贡献值可由下式计算得出。

$$\Delta_\varphi(s_i, R_v)=(f_1(s_{i+1})-f_1(s_i))*(f_2(s_{i-1})-f_2(s_i))$$

其中，f_1 为目标一的评估函数，f_2 为目标二的评估函数。

计算超体积贡献值的示意图如图 6 所示。

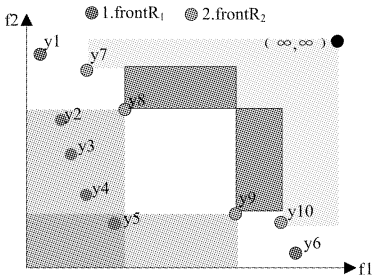

图 6 超体积贡献示意图

图 6 以目标一的评估函数 f_1 为横轴，以目标二的评估函数 f_2 为纵轴。图中坐标点集合{$y7 \cdot y8 \cdot y9 \cdot y10$}为第一前沿集，先以 f_1 为参考值按升序排序，若同前沿集中多个体 f_1 值相等，再以 f_2 为参考值按降序方式排序，则排序后的集合为{$y7 \cdot y8 \cdot y9 \cdot y10$}；同理，第二前沿排序后的集合为{$y1 \cdot y2 \cdot y3 \cdot y4 \cdot y5 \cdot y6$}。下面以计算图中坐标点 $y8$ 的超体积贡献值为例，$y8$ 为第一前沿集合中的第二个坐标点，则超体积贡献值 $\Delta_\varphi(s_2, R_1)=(f_1(s_3)-f_1(s_2))*(f_2(s_1)-f_2(s_2))$。

3.5 基于 SMS-EMOA 的云计算虚拟机分配算法

在多目标优化算法中，NSGA-II(Non-dominated Sorting Genetic Algorithm II)[10]和 SPEA2(Improving the Strength Pareto Evolutionary Algorithm)被广泛使用，但两种算法在衡量改善优化方面的进展上存在缺陷。SMS-EMOA(Multi-objective selection based on dominated hyper-volume)[11]算法在多目标优化问题上对于不同的锋面形状表现出了更好的性能和更优化的结果。因此，将云计算资源分配的基本演化算子嵌入 SMS-EMOA 中，设计基于 SMS-EMOA 的云计算资源分配算法。

基于 SMS-EMOA 的云计算资源分配算法框架如图 7 所示。

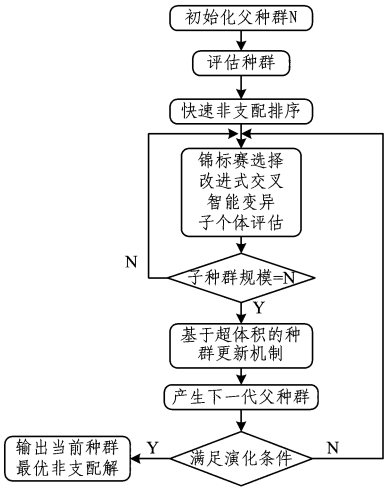

图 7 基于 SMS-EMOA 的云计算资源分配算法框架

算法描述如下：

①按基于组的链式编码方式，初始化父个体，生成规模为 N 的父种群，并对种群中个体进行适应度评估。

②对评估后的父种群进行快速非支配排序，划分出种群中个体的支配等级。

③对种群进行演化操作，该算法采用锦标赛选择法从种群中分别选取两个父个体，并对两父个体进行改进式单点交叉生成一个杂交个体，然后对杂交个体进行智能变异，得到一个子个体，最后对子个体进行适应度评估，并将子个体加入子种群池。重复此步骤，直到子种群规模为 N。

④对父种群和子种群采用基于超体积的种群更新机制筛选最优的 N 个个体作为下一代父种群。

⑤对新一代父种群重复③、④步骤，直到满足演化条件(迭代次数$<MaxGeneNum$)，则算法终止。

4 实验结果分析

为验证新算法的性能，分别使用传统的优先匹配启发式算法(Priority matching heuristic，PMH)、以物理节点数量为目标的单目标简单遗传算法 CGA、新提出的多目标演化算法(SMS-EMOA)进行模拟实验。模拟实验采用澳大利亚墨尔本大学的网格实验室和 Gridbus 项目提出的云计算仿真软件 CloudSim[12]、编程工具 Myeclipse6.5、编译工具 Ant1.8.4 进行仿真。

为实现该算法，需要在 CloudSim 的基类基础上编写对应算法的继承类及方法。从 Host 类中获取各物理机的可用性

能向量，从 DatacenterCharacteristics 类中获取虚拟机的需求性能向量。VmAllocationPolicy 抽象类用于实现虚拟机到物理节点上的部署过程，类中 allocationHostForVm(Vm vm)方法的作用是将指定虚拟机分配到目标物理节点。继承类 VmAllocationPolicySimple 为 CloudSim 自带的虚拟机分配策略类，通过编写自定义继承类 VmAllocationSMSEMOAPolicy，实现基于 SMS-EMOA 的云计算虚拟机分配算法，编写类方法 SMSEMOA(p,v)演化得出虚拟机部署序列。依据此序列，编写 allocationHostForVm(Vm vm)方法，将各虚拟机分配到指定的物理节点上。在完成 CloudSim 平台中类的扩展后，使用 Ant 工具重编译重写的类，最后编写 CloudSim 仿真程序进行试验仿真。

CloudSim 仿真程序模拟数据中心的物理服务器为 30 台以及分别部署 20、30、40、50 个虚拟机的情况(见表 1)，并输出 f_{PN} 和 f_{LB}。

表 1　虚拟机分配算法模拟实验参数

物理节点数	虚拟机数	虚拟机分配算法
30	20	PMH、CGA、SMS-EMOA
30	30	PMH、CGA、SMS-EMOA
30	40	PMH、CGA、SMS-EMOA
30	50	PMH、CGA、SMS-EMOA
30	60	PMH、CGA、SMS-EMOA

演化算法的种群规模和演化代数分别设为 30 和 1000，交叉率、变异率分别设为 0.6 和 0.02(见表 2)。

表 2　虚拟机分配算法参数

种群规模	演化代数	交叉率	变异率
30	1000	0.6	0.02

依据以上参数进行演化实验，结果见表 3。

表 3　算法性能对比表

虚拟机数	算法	开启物理节点数 f_{PN}	集群负载方差 f_{LB}
20	PMH	17	0.0040
	CGA	6	0.0025
	SMA-EMOA	7	0.0021
30	PMH	28	0.0041
	CGA	10	0.0022
	SMA-EMOA	10	0.0022
40	PMH	30	0.0042
	CGA	13	0.0026
	SMA-EMOA	14	0.0023
50	PMH	30	0.0033
	CGA	16	0.0024
	SMA-EMOA	17	0.0021
60	PMH	30	0.0031
	CGA	19	0.0022
	SMA-EMOA	19	0.0020
70	PMH	30	0.0039
	CGA	21	0.0023
	SMA-EMOA	21	0.0019

为进一步分析表 3 中算法的性能，图 8、图 9 给出了直观的数据曲线图。由图可以看出，在启用物理节点数性能方面，基于 SMS-EMOA 的虚拟机部署算法所占用的物理节点数与基于 CGA 的虚拟机部署算法所占用的物理节点数量基本相当，且少于优先匹配启发式(PMH)部署算法，物理节点占用数量越小，越节省能耗。而在集群的整体负载性能方面，基于 SMS-EMOA 的虚拟机部署算法的集群整体负载方差均小于以上两种虚拟机部署算法的集群整体负载方差，负载方差越小，服务器集群负载均衡效果越好。基于以上分析，可以得出基于 SMS-EMOA 的虚拟机部署算法可以大大减少物理服务器开启数量，且能使服务器集群达到更好的负载均衡效果。

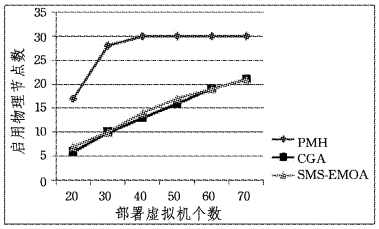

图 8　启用物理节点数示意图

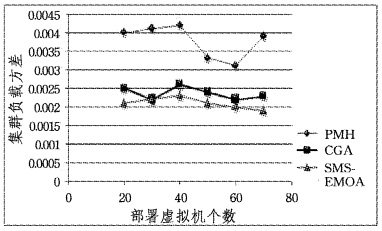

图 9　集群负载方差示意图

结束语　本文主要研究基于多目标演化算法的云计算虚拟机分配策略。通过分析云计算虚拟机资源模型，针对模型中虚拟机与物理机的映射问题，以及虚拟机多资源因子、多优化目标的特点，我们将一个多目标虚拟机分配问题转化成一个多维装箱问题，引入多目标演化算法的思想进行求解。算法设计了基于组的虚拟机分配链式编码和染色体评估函数，并根据编码设计了最低索引-最大长度交叉、改进单点交叉两种交叉算子和一种智能突变的变异算子，基于 SMS-EMOA 算法较其他演化算法对不同的锋面形状表现出更好的性能。将云计算虚拟机分配的演化算子嵌入 SMS-EMOA 中，设计了基于 SMS-EMOA 的云计算虚拟机分配算法，算法更新策略基于超体积的种群更新机制能保证种群质量与多样性。为验证 SMS-EMOA 的性能，分别使用 PMH、CGA、SMS-EMOA 进行模拟实验，实验结果表明基于 SMS-EMOA 的虚拟机部署算法在性能上更优。本文下一步的研究工作重点将放在如何设计性能更好的演化算子上，以此设计能快速高效进行云计算虚拟机分配的改进算法。

参 考 文 献

[1] Chen Kang, Zheng Wei-Min. Cloud Computing: System Instances and Current Research[J]. Journal of Software, 2009, 20(5): 1337-1348

[2] DMTF. Open virtualization format specification, DSP0243[S]. Portland, OR: DMTF, 2009

[3] 杨星，马自堂，孙磊. 云环境下基于性能向量的虚拟机部署算法[J]. 计算机应用，2012，32(1)：16-19

[4] 华夏渝，郑骏，胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报：自然科学版，2010(1)：127-134

[5] Li Qiang, Hao Qin-fen, et al. Adaptive Management and Multi-Objective Optimization for Virtual Machine Placement in Cloud Computing [J]. Chinese Journal of Computers, 2011, 34(12)

[6] 徐星. 基于动力学演化算法的云任务与虚拟机分配策略研究[J]. 科学技术与工程，2013，13(1)：85-89

[7] Falkenauer E, Delchambre A. A genetic algorithm for Bin Pa-

cking and line balancing[C]//Proceedings of the IEEE International Conference on Robotics and Automation. 1992:1186-1192
[8] Ülker Ö, et al. A Grouping Genetic Algorithm Using Linear Linkage Encoding for Bin Packing[J]. Lecture Notes in Computer Science,2008,5199:1140-1149
[9] Emmerich M,et al. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion[J]. Lecture Notes in Computer Science,2005,3410:62-76
[10] 方锦明. 云计算中基于 NSGA-II 的虚拟资源调度算法[J]. 计算机工程与设计,2012,33(4)
[11] Beume N,et al. SMS-EMOA:Multi-objective selection based on dominated hypervolume[J]. European Journal of Operational Research,2007,181(3):1653-1669
[12] Calheiros R N,Ranjan R,De Rose C A F,et al. CloudSim:A novel framework for modeling and simulation of cloud computing infrastructures and services[R]. GRIDS-TR-2009-1. Parkville, VIC:The University of Melbourne Australia,Grid Computing and Distributed Systems Laboratory,2009

(上接第 21 页)

同,而且后 5 个任务图的迭代次数要远高于前 5 个。这是因为前 5 个任务图的任务量范围是 1～10,而后 5 个任务量范围是 1～20。随着任务量的增加,Orsila 算法的迭代次数增加。ASA 算法的各任务的迭代次数只与核心数和任务数有关,所以各个任务图的迭代次数相差不大。对比图 5(d)(e)和图 5(f),ASA 算法的迭代次数在一定程度上反映了加速比和相对偏差,特别是当核心数为 16 时,任务 0 和任务 4 的迭代次数较少,加速比与并行度的差值较大,其相对偏差较高;其余任务的迭代次数较多,其相对偏差较少,其中任务 1、任务 2、任务 3、任务 7 和任务 8 均达到了最优解。这比较符合迭代次数与解的优化度之间的矛盾关系,Orsila 算法没有这一优势。

比较图 5(c)和图 5(f),ASA 算法在各个核心数各个任务图的相对偏差都低于 Orsila 算法,而且随着核心数的增加,相对偏差降低比较快,其中在 16 核心时,任务 1、任务 2、任务 3、任务 7 和任务 8 均达到了最优解,而 Orsila 算法均未能达到最优解,这表明 ASA 算法在核心数达到一定程度时,是可以寻到最优解的,这一性质对于某些要求最优解的应用领域非常重要。

结束语 为了平衡基于 SA 算法在多核/众核任务分配上的迭代次数与解精度的关系,本文在深入研究 SA 算法原理和多核/众核任务分配特点的基础上,建立初始温度、终止温度、每个温度级别的迭代次数、连续拒绝次数的最大值、接受概率函数、降温函数等 SA 算法参数与处理器内核数和任务数之间的优化关系,提出了一种 ASA 算法。本算法与已有的 Orsila 算法相比,能在处理器内核数增加的情况下,有效控制算法迭代次数增长速度,极大降低近似解的相对偏差,具有更好的环境自适应能力。

SA 算法虽然提出比较早,也较早应用在多核/众核任务分配中,但是因为算法本身的复杂性以及多核/众核平台的不断发展,我们还需要对模拟退火算法做进一步深入研究,使其能够适应更为复杂的环境,具有更高的优化效率和更快的收敛速度。

参 考 文 献

[1] Partitioning S V. scheduling parallel programs for execution on multiprocessors[M]. MIT Press,1989
[2] Huang L, Yuan F, Xu Q. On task allocation and scheduling for lifetime extension of platform based mpsoc designs[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(12):2088-2099
[3] Kim J-K,Shivle S,Siegel H J,et al. Dynamically mapping tasks with priorities and multiple dead-lines in a heterogeneous environment[J]. ParallelDistrib. Comp. ,2007,67(2):154-169
[4] Koch P. Strategies for realistic and efficient static scheduling of data Independent algorithms onto multiple digital signal processors[R]. Technical report. The DSP Research Group, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, December 1995
[5] Ferrandi F,Pilato C,Sciuto D,et al. Mapping andscheduling of parallel C applications with ant colony optimization onto heterogeneous reconfigurable MPSoCs[C]//Design Automation Conference(ASP-DAC), 2010 15th Asia and South Pacific. 2010:799-804
[6] Coroyer C, Liu Z. Effectiveness of heuristics andsimulated annealing for the scheduling of concu-rrent tasks an empirical comparison[M]//Rapport derecherche de l'INRIA Sophia Antipolis. 1991:452-463
[7] Heikki O. Optimizing algorithms for task graph mapping on multiprocessor system on chip SoCs[D]. Tampereen teknillinen yliopisto,Julkaisu Tampere University of Technology, Publication,2011
[8] 温平川,徐晓东. 并行遗传/模拟退火混合算法及其应用[J]. 计算机科学,2003,30(3):86-89
[9] 彭晓明,郭浩然,庞建民. 多核处理器——技术、趋势和挑战[J]. 计算机科学,2012,39(Z11):320-326
[10] Wentzlaff D,Griffin P,Hoffmann H,et al. On chip interconnection architecture of the tile processor[J]. IEEE MICRO,2007,27(5):15-31
[11] Zhang Y P,Jeong T,Chen F,et al. A study of the on chip interconnection network for the ibm cyclops64 multicore architecture[C]//Proceedings of the 20th international conference on Parallel and distributed processing, ser. IPDPS'06. Washington, DC, USA:IEEE Computer Society,2006:64-74
[12] Fan Dong-rui, Yuan Nan, Zhang Jun-chao, et al. Godson-t: An efficient many-core architecture for parallel program executions[J]. Journal of ComputerScience and Technology,2009,24(6):1061-1073
[13] Standard Task Graph Set [OL]. http://www.kasahara.elec.waseda.ac.jp/schedule/,2005-12-20
[14] Hashemi M. Automated Software synthesis for streaming applications on embedded manycore processors[D]. University of California,Davis,2011
[15] Kw Y K,Ahmad I. Static scheduling algorithms forallocating directed task graphs to multiprocessors[J]. ACM Comput. Surv. ,1999,31(4):406-471
[16] 王颖锋,刘志镜. 面向同构多核处理器的节能任务调度方法[J]. 计算机科学,2011,38(9):294-297