文章编号:1671-4598(2021)01-0168-06 DOI:10.16526/j.cnki.11—4762/tp.2021.01.035 中图分类号:TP393.2 文献标识码:A

容器云中基于改进遗传算法的资源分配策略

张松霖
(太原理工大学 软件学院,山西 晋中 030600)

摘要:容器很容易针对 Web 应用程序提供包装、迁移和配置等服务,近年来已成为研究热点;提出了容器云中基于改进遗传算法的资源分配策略 Double—GA;Double—GA 是一种包括两个层次的资源分配策略:容器到虚拟机的资源分配和虚拟机到物理主机的资源分配;设计了容器云的两层资源分配的数学模型,以容器云中的整体物理主机能量消耗作为 Double—GA 策略的目标函数;Double—GA 以遗传算法为基础,设计了双染色体的表达方式并处理好了遗传算法的初始化、进化、交叉、变异等操作;真实的实验实例数据结果表明:Double—GA 双染色体算法明显优于普通遗传算法 GA 和递减最好适用算法。

关键词:资源分配;多维装箱算法;遗传算法;虚拟机分配;容器云

An Improved Genetic Algorithm for Resource Allocation in Container Clouds

Zhang Songlin
(School of software, Taiyuan University of Technology, Jinzhong 030600, China)

Abstract: Research of container is becoming a hot problem because it is easier for application providers to pack, migrate, and deploy web applications than using virtual machines. An improved Genetic algorithm for resource allocation in container clouds called Double—GA was proposed in this paper. Double—GA is a two—level resource allocation strategy, the containers are allocated to virtual machines and virtual machines are allocated to physical machines. The mathematics model of two—level resource allocation in container—based cloud was presented and the overall energy consumption of physical resource was designed as the objective function in GA. A new dual—chromosome representation was used for new genetic operators such as initialization, crossover, mutation and fitness function. The experimental results show that Double—GA gains much better than the single—GA and decreased best fit algorithm in all test instances.

Keywords: resource allocation; mutil—dimension packing algorithm; genetic algorithm; virtual machine allocation; container—based clouds

0 引言

随着无服务器及微服务技术等应用的发展,容器云(Container—based Cloud)已经成了软件领域主流平台[1],与早期的虚拟机比较起来,容器很容易针对 Web 应用程序提供包装、迁移和配置等服务。近年来 Google 和 Microsoft 等公司都开始在大数据中心部署大量的容器为基础的应用。虽然服务器合并技术是一个很好的节省能量消耗的办法,但是在容器云中服务器合并是很难达到的,因为容器云中的资源分配是两层的资源分配:容器到虚拟机的分配和虚拟机到物理主机的分配。在早期基于虚拟机的云中扩展的服务器合并技术也无法重新再使用[2]。

为了处理好容器云中的两层资源分配问题,必须解决好 2 个方面的难题:1)容器分配和虚拟机分配之间的交互问题,即最好的容器分配不一定能够获得最好的虚拟机分配结果,所以这两个过程必须同时进行。2)由于每一层的分配都是多维的装箱问题,必须处理好这类多目标优化的 NP—Hard 问题[3]。

本文的研究重点是设计与实现一个容器云中的两层资源分配策略 Double—GA,同时 Double—GA 必须使云数据中心的能量消耗尽量最小。为了达到这个目标,Double—GA 应用了多目标优化智能计算的遗传算法。遗传算法[4-5]已经被成功应用到组合优化领域,它可以很好地避免早熟和收敛速度慢的问题,文献[6-8]中遗传算法已经成功应用到了基于虚拟机的云资源分配领域。

Double—GA 中利用遗传算法解决容器云的资源分配主要途径是设计出染色体的表达方式,处理好遗传算法染色体的初始化、种群进化、交叉、变异等操作,原因是遗传算法中好的染色体表达方式可以扩展搜索空间使得资源的分配更接近最优解。Double—GA 中采用了一个新的双染色体的方式和新的进化操作,最后的仿真实验结果表明,Double—GA 针对其他的常见启发式智能算法具有更好的资源分配效果与更小的能量消耗[9]。

1 相关工作

本节首先描述了容器云中服务器合并技术的相关工作,接着描述了遗传算法在云资源分配方面的相关工作。

收稿日期:2020-05-25; **修回日期**:2020-06-18。
作者简介:张松霖(1990-),男,山西忻州人,硕士研究生,主要从事云计算,软件工程方向的研究。
引用格式:张松霖.容器云中基于改进遗传算法的资源分配策略[J].计算机测量与控制,2021,29(1):168-173.

当前容器云中的服务器合并问题被认为是一个动态问题[2]，即在客户端有请求达到的时候服务器同时分配一个容器。文献[10]和文献[11]采用的是云中的一组预定义的虚拟机类型，应用首次适用启发式算法 First Fit Algorithm 来分配虚拟机到物理主机。在迁移阶段，根据像最小全物理主机选择 Least Full Host Selection 算法[10]和首次适用物理主机选择 First Fit Host Selection 算法等方式来完成容器分配。

这些策略大部分都是贪心模式的，为了可以快速分配资源。上述的启发式算法具有 2 个方面的不足，首先正如文献[2]所指出的，动态的算法比静态的算法性能还要低，因为动态算法针对容器进行频繁的迁移会产生额外的通信开销。其次，贪心模式的启发式算法往往只能取得局部资源分配最优，不能得到全局资源分配最优。另外一方面，文献[2]建议静态的方式来完成最初的容器分配，因为初始的分配一般会有很长时间的容忍期，尽管静态算法可能有很长的处理时间，但是它一般比贪心算法可以获得更加好的性能。

随着近年来容器云的出现，大部分容器云都不能选择虚拟机的类型。文献[12]认为一个虚拟机是一个类型，每个物理主机容纳不超过 10 个虚拟机。

它提出了一个 integer linear programming（ILP）整数线性编程方式来分配容器。如果只考虑一个预定的虚拟机类型，不仅不能适应不同类型的资源需求，而且会导致资源浪费。而且分配更加多的虚拟机会导致更多的额外开销。从它们的算法的性能中，可以得知整数线性编程方式的处理时间开销随着问题尺寸几乎呈指数增长。

容器云所面对的双层资源分配如图 1 显示，图 1（a）是早期的基于虚拟机的资源分配，图 1（b）是现在常用的基于容器的资源分配。文献[9]中提出了多目标优化的遗传算法来解决两层容器云资源分配问题。它采用的是单染色体的表达方式，更好地完成了两层资源分配，尽管如此，该方法的染色体的设计、遗传算法的各类操作都比较限制，它不能采用交叉操作，完全依赖于局部搜索的交叉。遗传算法中的交叉操作是可以很好的增强搜索能力的，基于此目的，本文设计了改进高效的遗传算法 Double－GA，它具有好的交叉能力，同时染色体表达与普通遗传算法也不一样。

2　问题定义与系统建模

在容器云中假设有 N 个容器的集合 $\{1,....N\}$，在资源分配阶段要分配到云数据中心的 $V\{1,....V\}$ 个虚拟机上运行，然后把这些虚拟机分配到物理主机 $P\{1,2,....P\}$ 上，如图 2 所示，我们的目标是让所有的物理主机的能量消耗 AE 最小。下面的公式分别计算了物理主机的能量消耗 E_p 和整个容器云的能量消耗 AE。

$$AE = \sum_{p=1}^{P} E_p * | u_{cpu}(p) | \tag{1}$$

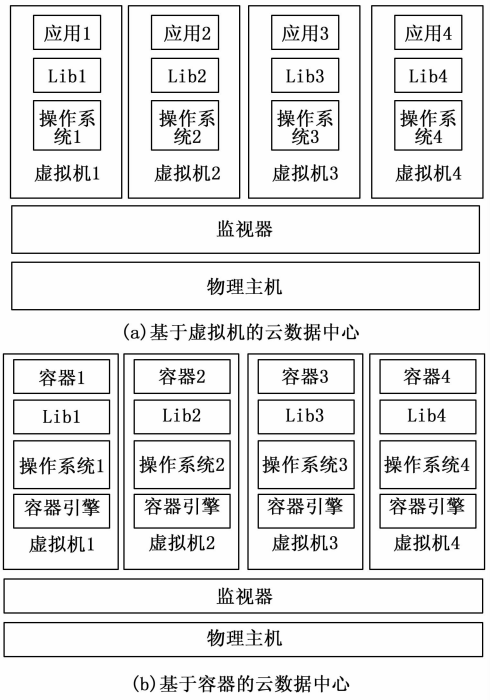

图 1　容器云和虚拟机云的比较

$$f = \min(AE) \tag{2}$$

f 表示了目标函数，E_p 在后面的公式中介绍，下面的约束条件式（3）～（4）保证了容器的整体资源需求不能超过它的目标虚拟机的处理能力。约束条件式（5）～（6）保证了虚拟机的整体资源需求不能超过它的目标物理主机的处理能力。约束条件式（7）保证了每个容器只能被配置一次。

$$\sum_{n=1}^{N} C_n * x_{nv} \leqslant VC_v \tag{3}$$

$$\sum_{n=1}^{N} M_n * x_{nv} \leqslant VM_v \tag{4}$$

$$\sum_{v=1}^{V} VC_v * y_{vp} \leqslant PC_p \tag{5}$$

$$\sum_{v=1}^{V} VM_v * y_{vp} \leqslant PM_p \tag{6}$$

$$\sum_{n=1}^{N} x_{nv} = 1 \tag{7}$$

式（1）中的 E_p 是活动物理主机的能量消耗。$| u_{cpu}(p) |$ 在物理主机的 CPU 利用率大于 0 的时候等于 1，其他的情况等于 0。

$$E_p = E_{idle} + (E_{\max} - E_{idle}) * (\frac{u_{cpu}(p) + u_{mem}(p)}{2}) \tag{8}$$

E_{idle} 和 $E_{\max}$ 分别表示了物理主机在空闲和满负载的时候的能量消耗。在 Double－GA 策略中，容器、虚拟机、物理主机都与装箱问题中的两个维度的物理资源所关联，处理器和内存。容器的处理器需求和内存需求定义为 C_n 和 M_n。虚拟机的处理器需求和内存需求定义为 VC_v 和 VM_v。物理主机的处理器计算能力和内存大小分别定义为 PC_p 和

PM_p。容器云中的资源的数量定义为区域范围［1，2，…，R］。每个虚拟机因为处理器和内存的额外开销定义为 OC_v 和 OM_v，额外开销在容器云中心也表示为资源。

本策略中假设大量的虚拟机需求类型 VC_v 和 VM_v 组合成 $Type_v$。对于容器而言，我们认为客户端的应用与容器是一对一的关系，这就意味着我们定义的容器的资源需求范围在 1 到虚拟机最大类型数量 $Type_v$ 之间，一个虚拟机可以容纳多个容器。

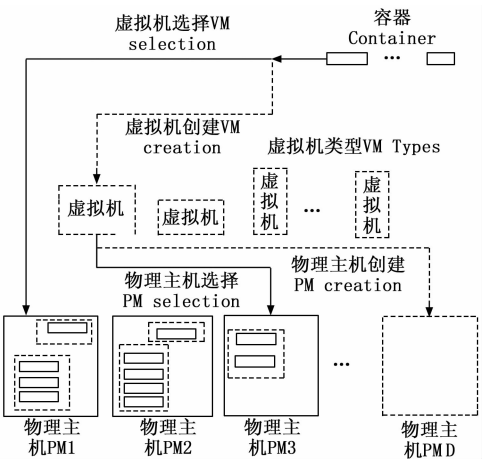

图 2 容器云中的两层资源分配策略

两层资源分配策略主要依赖于物理资源的利用效率情况来确定。在容器到虚拟机级别，主要体现在虚拟机的处理器和内存的利用率情况。它们分别通过式（9）和式（10）完成计算。

$$u_{cpu}(v)=\frac{\sum_{n=1}^{N}C_n*x_{nv}}{VC_v} \tag{9}$$

$$u_{mem}(v)=\frac{\sum_{n=1}^{N}M_n*x_{nv}}{VM_v} \tag{10}$$

变量 x_{nv} 的取值范围是 0 或 1，它分别表示容器 n 是否被分配到虚拟机 v 之上。在虚拟机到物理主机级别，主要体现在物理主机的处理器和内存的利用率情况。它们分别通过式（11）和式（12）完成计算。

$$u_{cpu}(p)=\frac{\sum_{v=1}^{V}(OC(v)+u_{cpu}(v)*VC_v)*y_{vp}}{PC_p} \tag{11}$$

$$u_{mem}(p)=\frac{\sum_{v=1}^{V}(OM(v)+u_{mem}(v)*VM_v)*y_{vp}}{PM_p} \tag{12}$$

物理主机的资源的利用率体现其上容纳的虚拟机的资源运行情况，变量 y_{vp} 的取值范围是 0 或 1，它分别表示虚拟机 v 是否被分配到物理主机 p 之上。

3 Double－GA 算法描述

这部分介绍了 Double－GA 策略的设计，包括染色体表达的设计，进化操作的设计和适应度函数的设计等。

3.1 总体描述

3.1.1 总体描述

Double－GA 策略是以遗传算法为基础，在算法思路、流程和算法步骤都参考了遗传算法和基于虚拟机的物理资源分配等技术，不同的地方是它带有针对特定的容器云的两层资源分配及其相关的染色体设计和操作设计。

3.1.2 染色体表达

个体的表示由于 2 个不同的染色体组成，一个染色体是用来做容器分配，另外一个作为虚拟机分配。图 3 显示了染色体的设计，包括算法的输入和输出。通过解码过程，一个完整的容器云资源分配问题可以得到很好的解决。所有的染色体都是具有整数值的向量。在染色体的容器分配阶段，每个值表示了最初的输入的容器的索引编号，染色体的长度就是整个容器的数量。

在染色体的虚拟机分配阶段，每个输入口表示了从所有虚拟机类型中所存在的一个虚拟机类型。虚拟机分配向量的长度等于容器分配向量的长度，这是因为我们最多可以使用 N 个虚拟机来容纳 N 个容器，即一对一的映射。

为了把一个输入的个体通过遗传算法的解码来完成最终的分配结果，Double－GA 在两个层次中都应用了启发式的装箱算法中的下次适用算法（Next Fit Algorithm），在容器到虚拟机的层次，容器都是按照顺序装载到虚拟机中。例如图 3 中，在容器 5 装到虚拟机 0 后，接下来的容器 1 就不能继续装载到容器 0 中了。因此我们将关闭虚拟机 0，打开虚拟机 1 来容纳容器 1。被关闭的虚拟机将不能再重新检测和被装载。当所有的容器都被分配完成，解码过程才结束。

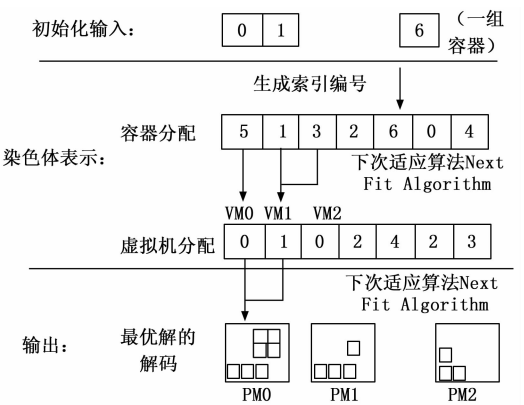

图 3 容器云中的双染色体完成资源分配

类似的，在虚拟机装载到物理主机的层次中，也采用同样的规则，一方面，这种染色体表示方式可以保证运用到下次适用算法来完成解码，再不必要设计另外的约束处理方法；另一方面，这种双染色体表示方法可以覆盖所有的解空间，也方便我们找到最优的容器云的资源分配。

3.1.3 初始化

对每个种群的个体而言，我们可以随机的对索引编号，

同时产生出容器的分配染色体序列。为了初始化虚拟机分配的染色体，这里统一的采用虚拟机的类型来产生。

3.2　交叉、变异与适应度函数

3.2.1　交叉操作

遗传算法中设计一个好的交叉操作对提高虚拟机资源的利用效率十分关键，这里利用了 order1 交叉操作来预防可能的下一代的染色体的预早熟现象[13]，针对在虚拟机资源分配中使用的染色体，本文采用了单点交叉操作[14]。

order1 交叉操作是从父代中随机的选择一个连续的序列，在另外一个父代中，把剩余的值将按照相同的顺序放到孩子中。

例如图 4，容器 3 和容器 4 都被选择，并从父代中拷贝到孩子 child 1 中。然后相同的容器（3 和 4）将被交叉输出到父代 parent2 中。从父代 parent2 中剩余的值将从第 2 个切入点开始，回滚到染色体的开始。例如容器 1，5 和 2。同样的规则被应用到第二个孩子中。单点交叉操作首先随机的切断一个染色体序列成 2 个部分，孩子将继承其中一个部分 parent1 和另外一个部分 parent2。

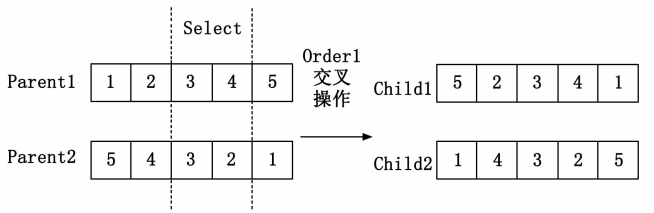

图 4　遗传算法中交叉操作的改进

3.2.2　变异操作

在遗传算法中，变异操作提供了一个局部最优解的搜索机制，它用来寻找当前个体的邻居情况。在染色体的两个层次分配中，Double－GA 设计了这个新的变异操作来完成局部最优解搜索。包括切换变异操作和虚拟机类型改变变异操作。

切换变异是在容器分配的染色体上随机的选择 2 个入口，并改变它们的值。这种变异可以改变 2 个容器的分配。虚拟机类型改变变异操作通过虚拟机分配染色体上循环，并以一定的概率统一的改变它们的值，这个变异操作可以改变虚拟机的类型。

3.2.3　适应度函数

适应度函数类似于组合优化的目标函数，就是通过解码操作后的云数据中心的所有物理资源的能量消耗。前面提到 *AE* 是云数据中心的整体能量消耗，所有这里遗传算法的适应度函数就是容器云的能量消耗 $AE = \sum_{p=1}^{P} E_p * | u_{cpu}(p) |$。

4　Double－GA 策略实验与性能分析

4.1　实验环境的设计

实验的目标是为了测试本文的双染色体的遗传算法中在处理两层容器云资源分配后云数据中心的能量消耗效果情况及时间成本，本节中利用了真实的样本数据集[15]，同时与常见的两种算法进行比较，单染色体算法 Single－GA 和最好递减装箱（decreased best fit algorithm，BFD）算法。

真实的数据集主要来自 AuverGrid 项目中的资源分配数据[15]。在容器云在硬件的配置上，采用了同等配置的物理主机，CPU 的主频为 3300 MHz，内存大小为 4 GB，一共 400 台，组成一个基于容器的云数据中心。为了测试不同虚拟机类型个数配置情况下容器资源分配的性能高低比较，我们设置了三个虚拟机类型集。如表 1～3 所显示，每个类型集中又有不同的虚拟机需求。

表 1　容器云中的虚拟机分类

虚拟机分类	虚拟机需求编号	处理器的 MIPS 能力需求/MHz	内存大小需求/MB
第一大类（个数为 5）	1	660	800
	2	1 320	1 600
	3	1 320	2 800
	4	1 980	2 400
	5	2 310	2 800

表 2　容器云中的虚拟机分类

虚拟机分类	虚拟机类型编号	处理器的 MIPS 能力需求/MHz	内存大小需求/MB
第二大类（个数为 7）	1	660	800
	2	1 320	1 600
	3	1 320	2 800
	4	1 980	2 400
	5	2 310	2 800
	6	660	2 400
	7	1 320	2 800

表 3　容器云中的虚拟机分类

虚拟机分类	虚拟机类型编号	处理器的 MIPS 能力需求/MHz	内存大小需求/MB
第三大类（个数为 10）	1	660	800
	2	1 320	1 600
	3	1 320	2 800
	4	1 980	2 400
	5	2 310	2 800
	6	660	2 400
	7	1 320	2 800
	8	660	2 800
	9	1 980	1 200
	10	2 310	1 600

第一大类虚拟机包括 5 种虚拟机需求（编号 1～5），第二大类虚拟机包括 7 种虚拟机需求（编号 1～7），第三大类虚拟机包括 10 种虚拟机需求（编号 1～10）。同时我们设计了四个实例数据（Instance），如表 4，为了是区别不同的容器个数。通过这样设置，对于每个虚拟机类型设置，三个大类和四个实例数据，这些实例数据上运行的 Double－GA 策略一共可以包括 12 个实验。

表 4 虚拟机的实例设置

实例数据名称	容器的数量
Instance1	100
Instance 2	200
Instance 3	500
Instance 4	1 000

Double－GA 策略的参数设置与基本的遗传算法一致，也有自身的改变，见表 5。在精英中配置的大小为 10，在竞赛中设置了大小为 7。因为这些参数都是标准设置和广泛使用的。针对单染色体遗传算法，设置了交叉率为 0.8，因为 Double－GA 策略的性能的提高很大程度上完全依赖于搜索的交叉。算法采用 Java 来实现，实验运行在 Intel i7 3.6 GHz 处理器的云客户端上，内存为 8 GB，操作系统为 Linux。

表 5 遗传算法的容器云资源分配参数设置

参数	取值
交叉率 x	80%
双染色体变异率	80%
单染色体变异率	80%
精英 elitism	Top 10 的个体
进化的代数量	1 000
种群 Population	100
Tournament Selection	Size＝7

4.2 实验比较对象

单染色体遗传算法在文献[9]中有描述过两层容器云的资源分配问题，有两个方面的原因导致本文对它进行了实际的改进：1）它是直接基于 NSGA－II 多目标优化的遗传算法。2）它们的主要假设是允许虚拟机超负载[2]，同时可以容纳某些物理资源超过利用率的虚拟机。在 Double－GA 策略中，虚拟机超负载是不允许的，因此，为了作为性能比较，在初始化的时候，既然不允许虚拟机超负载，在虚拟机没有足够的容纳能力下，实验中也不允许太多的容器被分配到一个虚拟机上。因此，针对 Single－GA 单染色体，虚拟机类型都是随机的产生并且容器都是运行首次适用算法 First Fit Algorithm 来分配，交叉操作中随机的切换两个容器。

在递减最好适用 Best Fit Descending algorithm 算法也作为本文的比较对象，在实验的时候，由于递减最好适用算法没有判断和选择虚拟机类型的功能，为了创建新的虚拟机，实验中我们一直选择的最大的虚拟机需求类型。例如表 1 中的类型 5，虚拟机处理需求为 2 310 MHz，内存需求大小为 2 800 MB。

4.3 实验结果与性能分析

4.3.1 能量消耗的比较

这个部分显示了三个容器云资源分配算法在能量消耗方面的比较，图 5 和图 6 显示三个算法一天 24 小时内在 Instance3 和 Instance4 数据集上的平均能量消耗比较，单位为千瓦特小时。这里只显示了 2 个实例因为 instance1 和 instance2 在数据上基本和 instance3 类似。图中表示递减最好适用算法 BFD 是容器云中整体能量消耗最大的，单染色体遗传算法 Single－GA 性能比 BFD 算法好，所有的测试实例数据中，本文的双染色体 Double－GA 遗传算法性能最优。

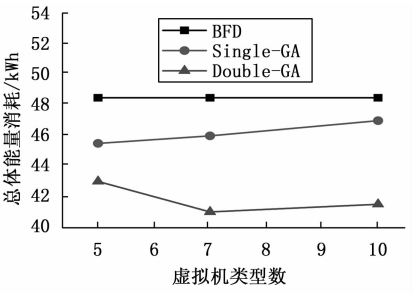

图 5 Intance3 数据下的能量消耗比较

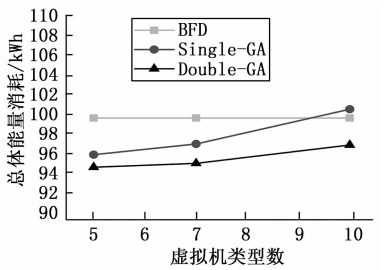

图 6 Intance4 数据下的能量消耗比较

该实验数据还表明容器云的能量消耗因为资源分配方式的不同而不同，递减最好适用算法 BFD 的性能变化不受虚拟机类型个数的影响，因为它一直在使用最大的虚拟机需求。另外，最好适用算法 BFD 是一个不可逆转的算法，这就意味着相同的容器输入，那么得到的能量消耗一直是相同的。

4.3.2 扩展性能的比较

表 6 显示了随着容器个数不断的增加，三个算法运行后的能量消耗的性能比较。双染色体遗传算法在各个实例数据情况下都比其他的 2 个算法能量消耗要节能，而且从图中可以看出随着容器数量增加，能量消耗都是稳步的缓慢增加，没有急剧的变化，这点证明 Double－GA 扩展性能较好。

4.3.3 虚拟机利用个数情况

表 7 显示了在实例 Instance3 和实例 Instance4 条件下的平均虚拟机个数利用情况，从这里可以看出，BFD 算法的虚拟机利用个数一直是最小的，因为它默认情况下选择了能力最大虚拟机。能力最大的虚拟机在物理主机中大约占有 70%的总资源。这就意味着 BFD 算法条件下只有 70%的物理资源利用效率，因为物理主机只能容纳一个需求最大的虚拟机。单染色体算法和双染色体算法比 BFD 算法可以利用更加多的虚拟机个数，因为在遗传算法检测使用的虚拟

表 6　三类容器云资源分配算法的时间消耗比较

容器数	资源分配算法	总体能量消耗/Kwh
100	BFD	12
	Single－GA	11
	Double－GA	10
200	BFD	18.5
	Single－GA	16.5
	Double－GA	15
500	BFD	48
	Single－GA	46
	Double－GA	41
1 000	BFD	99.5
	Single－GA	94.5
	Double－GA	92.5

机类型的时候，大部分情况下遗传算法可以找到能力需求互相补充的虚拟机。我们还发现遗传算法可以一直寻找到互相补充的虚拟机类型，同时能提高资源利用效率。在测试样例中，本文的双染色体遗传算法可以利用好更加多的虚拟机，间接的提高了物理资源利用效率，具有更低的能量消耗。

表 7　容器云中平均虚拟机利用数量比较

虚拟机类型数	资源分配算法	Instance3 下虚拟机利用数量/个	Instance4 下虚拟机利用数量(单位:个)
5	BFD	46	95
	Single－GA	65	105
	Double－GA	66	107
7	BFD	46	95
	Single－GA	67	106
	Double－GA	69	110
10	BFD	46	95
	Single－GA	66	106
	Double－GA	68	108

另外一个结论是提供过多的虚拟机类型可能会产生副面影响，因为多余的虚拟机类型将扩展最优解的搜索空间，形成比较坏的适应度函数，导致降低效率。表 7 中实例 Instance3 表明本文的双染色体遗传算法如果检测 7 个虚拟机类型，它的性能差于样例 Instance4 的 10 个虚拟机类型。例如在实例 Instance4 中，由于容器的数量比较大，搜索也接着变大，两类遗传算法的性能都有降低。

5　结束语

本文建立了容器云中的两层物理资源分配问题的数学模型，提出了一个基于双染色体的改进遗传算法来解决容器云的资源分配问题，真实的实验样本数据结果表明双染色体算法明显优于单染色体和递减最好适用算法。本文的容器云资源分配策略可以供其他云服务提供商构造节能绿色云数据中心做为参考。

参考文献：

[1] Shi T，Ma H，Chen G. Energy－Aware Container Consolidation Based on PSO in Cloud Data Centers [A]. IEEE Congress on Evolutionary Computation [C]. IEEE，2018.

[2] Wolke A，Bichler M，Setzer T，et al. dynamic control：Resource allocation in corporate clouds [J]. IEEE Transactions on Cloud Computing，2016，4 (3)：322－335.

[3] Kaaouache M A，Bouamama S. Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud [J]. Procedia Computer Science，2015，60 (1)：1061－1069.

[4] Tang M，Pan S. A hybrid genetic algorithm for the energy－efficient virtual machine placement problem in data centers [J]. Neural Processing Letters，2014：1－1.

[5] Joseph C T，Chandrasekaran K，Cyriac R. A Novel Family Genetic Approach for Virtual Machine Allocation [J]. Procedia Computer Science，2015，46：558－565.

[6] 刘开南. 云数据中心基于遗传算法的虚拟机迁移模型 [J]. 计算机应用研究，2020，37 (4)：1115－1118.

[7] Meera Vasudevan，Yu－Chu Tian，Maolin Tang，et al. Energy－efficient Application Assignment in Profile－based Data Center Management Through a Repairing Genetic Algorithm [J]. Applied Soft Computing，2018，67 (1)：1－10.

[8] Ding Z，Tian Y C，Tang M. Efficient fitness function computation of genetic algorithm in virtual machine placement for greener data centers [A]. IEEE International Conference on Industrial Informatics (INDIN) [C]. IEEE，2018，181－186.

[9] Tan B，Ma H，Mei Y. A NSGA－II－based approach for service resource allocation in cloud [A]. IEEE Congress on Evolutionary Computation (CEC) [C]，2017，2574－2581.

[10] Piraghaj S F，Dastjerdi A V，Calheiros R N，et al. A Framework and Algorithm for Energy Efficient Container Consolidation in Cloud Data Centers [A]. IEEE International Conference on Data Science and Data Intensive Systems [C]. 2015，368－375.

[11] Kaur K，Dhand T，Kumar N，Zeadally S. Container－as－a－service at the edge：Trade－off between energy efficiency and service availability at fog nano data centers [J]. IEEE Wireless Communications，2017，24 (3)：48－56.

[12] Guan X，Wan X，Choi B，et al. Application oriented dynamic resource allocation for data centers using docker containers [J]. IEEE Communications Letters，2017，21 (3)：504－507.

[13] Poon P，Carter J. Genetic algorithm crossover operators for ordering applications [J]. Computers and Operations Research，Genetic Algorithms，1995，22 (1)：135－147.

[14] Agrawal R B，Deb K，Agrawal R. Simulated binary crossover for continuous search space [J]. Complex systems，1995，9 (2)：115－148.

[15] Shen S，Beek V v，Iosup A. Statistical characterization of business－critical workloads hosted in cloud datacenters [A]. IEEE/ACM International Symposium on Cluster，Cloud and Grid Computing [C]. 2015，465－474.