文章编号：1674－9057(2018)03－0555－06 doi:10.3969/j.issn.1674－9057.2018.03.028

一种基于 OpenStack 的云计算虚拟机放置策略

罗 平[a,b]，王 勇[b,c]，俸 皓[b,c]，何 倩[b,c]

（桂林电子科技大学 a. 信息与通信学院；b. 广西云计算与复杂系统高校重点实验室；
c. 计算机与信息安全学院，广西 桂林 541004）

摘 要：针对当前 OpenStack 云平台的虚拟机调度策略未综合考虑服务器能耗、使用数量、计算性能带来的高能耗、服务器数量较多、较高的服务等级协议(SLA)违背率的影响，构建了综合这几方面因素的能效优化模型，根据该模型设计了一种基于自适应双策略差分进化算法的虚拟机放置策略。算法包括变异阶段根据种群分组差异选择不同变异策略，增加自适应交叉与变异参数用于提高算法收敛速度，使用综合能效优化模型作为适应度函数评价进化个体。实验仿真表明，本文策略使得 OpenStack 中虚拟机放置具有比其内置算法和 DE、PSO 算法更低的能耗、更小 SLA 违背率和更少的服务器使用数量。

关键词：云计算；OpenStack；能耗；虚拟机放置；差分进化算法

中图分类号：TP393 **文献标志码：**A

随着云计算的不断普及，云计算数据中心的数量规模也不断扩大，但能耗也在与日俱增。据统计，全球的 CO_2 排放量有将近2%是由云数据中心产生[1]。同时也普遍存在资源利用率低的现象。高能耗、低利用率已成为云数据中心普遍面临的问题。目前云计算数据中心降低能耗、提高资源利用率主要通过关闭闲置服务器实现。由于云计算数据中心通常由大量异构服务器集群构成，在服务器集群中大规模部署虚拟机时，简单的关闭闲置服务器并不能够达到降低能耗的目的。同时在保证低能耗情况下，降低服务器使用数量将会影响服务器计算性能，增加服务等级协议 SLA 违背率。云计算数据中心的虚拟机放置策略、确定多个虚拟机与多个服务器之间的映射关系，已成为云数据中心节能研究的热点。

目前，OpenStack 平台采用 FilterScheduler 资源调度策略，该调度策略针对 CPU 资源使用情况确定虚拟机与服务器之间的映射关系，且并未考虑能耗、服务器使用数量和 SLA 违背率问题。本文结合 OpenStack 开源云平台，研究 OpenStack 中虚拟机调度问题。基于 OpenStack 的调度策略，提出一种基于自适应双策略差分进化算法虚拟机放置算法用于解决 OpenStack 调度策略存在的问题。

1 OpenStack 调度策略分析

目前，OpenStack 项目中关于虚拟机调度管理问题，已经专门孵化出 Nova 项目用于管理虚拟机调度，其体系结构如图1所示。

在 Nova 项目体系结构中，各核心组件间的通信主要是由符合 AMQP 协议的 RabbitMQ 实现。整

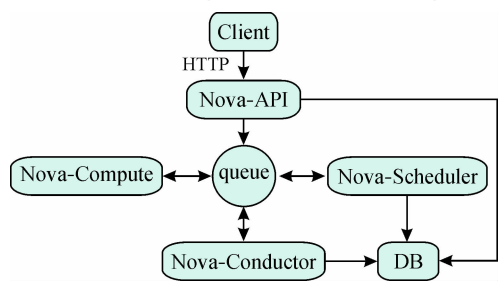


图1 Nova 体系结构图
Fig. 1 Nova system structure

收稿日期：2016－12－23
基金项目：国家自然科学基金项目(61661015；61662018)；广西云计算与复杂系统高校重点实验室项目(2015209；14103；15208)；广西云计算与大数据协同创新中心研究课题(YD16303)；广西研究生教育创新计划项目(JGY2014062)
作者简介：罗 平（1993—），男，硕士研究生，研究方向：云计算，lcarrotp@163.com。
通讯作者：王勇，博士，教授，ywang@guet.edu.cn。
引文格式：罗平，王勇，俸皓，等. 一种基于 OpenStack 的云计算虚拟机放置策略［J］. 桂林理工大学学报，2018，38（3）：555－560.

个项目包括 4 个核心组件，其中与调度关系最为紧密的当属 Nova-Compute 与 Nova-Scheduler 两个核心组件。Nova-Compute 组件直接通过调用系统底层虚拟机监控器 libvirt 提供的 API 用于创建、删除虚拟机操作，主要部署在整个体系的计算节点。Nova-Scheduler 根据各种规则，如各计算节点的内存使用率、CPU 使用率等制定相应的调度策略。目前该组件仅实现了 ChanceScheduler 随机选择策略、FilterScheduler 过滤选择策略。FilterScheduler 策略根据各计算节点 CPU 使用率等权重值(Weighting)进行过滤(Filetering)、排序选择虚拟机放置的目标节点。Nova-Scheduler 目前实现的调度策略非常简单，且考虑问题单一，因此基于 OpenStack 的调度策略也成为该项目的研究热点。

邓志龙等研究了在 OpenStack 中资源负载动态变化情况下的虚拟机迁移问题，减少了数据中心的能耗[2]。许波等充分考虑数据中心服务器异构问题，提出 EAPC 算法解决虚拟机部署的能耗问题，通过将物理节点划分为多个子集群，采用二级递阶分解寻找虚拟机与服务器最优映射关系，该方法有效降低了 IDC 能耗[3]。豆育升等在解决数据中心多资源能耗虚拟机放置的问题时，引入生物启发式算法——粒子群算法的高能耗虚拟机放置算法，粒子群算法与资源调度相结合可以有效避免采用贪心算法易陷入局部最优的不足[4]。差分进化算法目前已被运用到云数据中心任务调度中，使任务调度时间花费最少[5-6]。差分进化算法属于生物启发式算法，该算法能够发挥生物种群进化中的群体优势，寻找每次种群迭代中全局最优个体基因，并且具有收敛速度快，搜索范围广等优点。因此本文考虑将差分进化算法与虚拟机放置策略结合起来，研究基于差分进化算法以降低数据中心能耗为目标的虚拟机放置策略。通过分析 OpenStack 中已存在的调度策略具有贪心策略仅考虑当前最优的固有缺点，并且缺乏在大规模部署虚拟机时考虑全局能耗、服务器使用数量、服务器性能问题，构建了包括服务器能耗、使用综合能效模型，根据该模型设计出一种自适应双策略差分进化算法的虚拟机放置算法，求解 OpenStack 中虚拟机与服务器之间的映射关系。实验证明，通过本文策略在数据中心大规模部署虚拟机时，在数据中心增加的功率、服务器使用数量和 SLA 违背率均比 OpenStack 内置算法和基本差分进化算法以及 PSO 算法更优。

2 问题描述

2.1 策略约束条件

虚拟机放置问题可看成是装箱问题，寻找最佳的放置策略使数据中心增加的功率最小。物品即虚拟机 $N \in \{1,2,\cdots,n\}$，箱子即服务器 $M \in \{1,2,\cdots,m\}$，此类型问题必须满足实际应用情况的约束条件。虚拟机放置问题约束条件如下：

$$\sum_{i}^{n} H_{ij} = 1,\ J \in N = \{1,2,\cdots,m\},\ H_{ij} = 0 \text{ 或 } 1; \quad (1)$$

$$\sum_{i}^{n} r_i^{\mathrm{CPU}} \cdot H_{ij} < C_j^{\mathrm{CPU}}, \sum_{i}^{n} r_i^{\mathrm{RAM}} \cdot H_{ij} < C_j^{\mathrm{RAM}},$$

$$\sum_{i}^{n} r_i^{\mathrm{bw}} \cdot H_{ij} < C_j^{\mathrm{bw}}, \quad \sum_{i}^{n} r_i^{\mathrm{disk}} \cdot H_{ij} < C_j^{\mathrm{disk}}。 \quad (2)$$

i、j 表示虚拟机 i 被放置在物理机 j 上，0 时表示事件不成立，反之成立；N 表示服务器编号，虚拟机 i 可以部署在任意服务器中，数据中心服务器的总量为 m。式(2) 表示虚拟机的资源需求不能超过服务器提供的资源，其中，r_i^{CPU}、r_i^{RAM}、r_i^{bw}、r_i^{disk} 分别代表虚拟机 i 所使用的 CPU、内存、带宽、磁盘等资源；C_j^{CPU}、C_j^{RAM}、C_j^{bw}、C_j^{disk} 表示服务器 j 所能提供的 CPU、内存、带宽、磁盘等资源的最大值。

2.2 综合能耗优化模型的构建

本文算法的目标是在数据中心大规模创建虚拟机时，使得数据中心总功率增加最小，进而达到降低数据中心能耗的目的，同时保证服务质量与减少活动服务器数量。综合能耗优化模型的相关指标：

① 服务器功率。数据中心能耗模型能够有效预测服务器耗能情况，并为制定节能策略提供依据。动态电压频率调整技术(DVFS)得到 CPU 使用率与服务器能耗呈线性关系[7]。服务器功率可划分为静态功率和动态功率两部分，表示为

$$f_{\mathrm{P}} = \sum_{n=1}^{N} (k \cdot P + (1-k) \cdot P \cdot u)。 \quad (3)$$

其中：P 为服务器满载时的最大功率；k 表示服务器静态功率相对于满载功率的占比，通常取值为 0.7；u 表示服务器中 CPU 使用率，该值通过服务器负载情况确定；N 表示整个 IDC 中服务器总数，通过计算 IDC 所有服务器的功率累加和可得数据中心整体功率。

② 服务器数量。减少运行服务器数量能够降低运维成本，关闭闲置服务器降低能耗，同时也能减少宕机事故的发生，提供稳定的服务，可用下式统计服务器中任务服务器的数量。

$$f_{\mathrm{h}} = \sum_{j=1}^{m} H_j 。 \tag{4}$$

③ CPU 使用率。服务器的性能与其 CPU 使用率有关，过高的 CPU 使用率将降低服务器性能，也将触发虚拟机迁移，迁移过程同样会造成性能衰退。用下式统计活动服务器中 CPU 使用率

$$f_{\mathrm{s}} = \sum_{i=1}^{n} H_i / \sum_{j=1}^{m} H_j , \tag{5}$$

m、n 分别表示 IDC 中活动主机数与资源过载主机数。超过 0.75 的服务器数量作为衡量数据中心 SLA 违背率的性能指标，f_{s} 越小表示虚拟机服务器性能越优，越大则服务器性能受影响，

④ 个体适应度评价。综合式(3)、(4)、(5)，引入式(6)构建本文放置策略的综合优化目标模型，优先考虑能耗、其次考虑服务器数量与 SLA 违背率：

$$\min\{\mathrm{first}(f_{\mathrm{P}}), \mathrm{second}(f_{\mathrm{h}}), \mathrm{third}(f_{\mathrm{s}})\} 。 \tag{6}$$

该模型将充当本文改进算法的个体适应度评价函数，以达到降低整体功率、服务器使用数量和低 SLA 违背率目的。

3　双策略差分进化算法实现与分析

差分进化算法[8]与虚拟机放置策略结合，能够充分发挥算法的优势。最佳虚拟机与物理机的映射关系能够有效降低异构数据中心能耗。具体算法的设计与分析如下。

3.1　种群中个体基因编码与初始化

个体基因均为虚拟机与服务器之间的映射关系，个体适应度值为该基因所代表策略的综合能效值。个体基因向量 $\boldsymbol{X}^t = (x_{1,1}^t, x_{2,2}^t, \cdots, x_{D,M}^t)$ 表示虚拟机与服务器的映射关系；$x_{D,M}^t$ 表示在第 t 代时，编号为 D 虚拟机部署到编号为 M 的服务器中，如 $\boldsymbol{X}^t = (x_{1,1}, x_{2,2}, x_{3,1}, x_{4,3})$，表示编号为 1，2，3，4 的虚拟机部署到的编号为 1，2，1，3 的服务器中。在种群初始化阶段，$\boldsymbol{X}^t$ 中每一维度值 D 在服务器集合 $M \in \{1, 2, \cdots, m\}$ 中随机选取，构成每一个体基因。每一维度的新基因必须满足约束条件式（3）、（4）。采取先随机生成基因，然后逐一检查每一维度基因是否符合约束。记录随机生成的每个基因值，若当前维度对应的值不满足，则该值将排除在随机生成值之外再随机生成，直到满足条件产生下一维度基因。算法采用此方式运行的速度更快。

3.2　自适应参数的设计

3.2.1　*自适应变异参数 F 的设计*　基本变异策略对 F 值的选取多采用固定经验值方式确定。固定值 F 不能反映优势个体与劣势个体对于差异基因扰动的不同需求情况。因此在确定 F 值时多采用自适应策略[9]，根据每个个体基因的情况确定 F 值：

$$F = F_{\mathrm{l}} + (F_{\mathrm{u}} - F_{\mathrm{l}}) \cdot \left| \frac{fitness(i) - avgfitness}{bestfitness - avgfitness} \right| \cdot \frac{T - t}{T} 。 \tag{7}$$

这种改进型自适应变异参数策略在算法初期阶段，为确保整个种群的差异向量尽可能的大，建议 $F \in [0.1, 0.9]$，$F_{\mathrm{u}} = 0.9$，$F_{\mathrm{l}} = 0.1$，根据当前个体 i 适应度值 $fitness(i)$ 与种群平均适应度值 $avgfitness$ 的距离差值，改变 F 值的大小。差值越大表示当前个体适应度较差，应该使得 F 尽量取较大值，增大算法扰动。差值越小表示当前个体适应度值越好，应使得 F 取值尽量小，保持优秀个体的基因。T 表示整个迭代次数，t 表示当前迭代次数。总趋势满足在整个算法前期探索性广、后期收敛快。

3.2.2　*自适应交叉参数 Cr 的设计*　基本交叉策略采用固定值 Cr 并不能反映整个种群进化前后期以及种群中个体对于算法收敛性与探索性的需求。改进自适应交叉参数策略的 Cr 值常采用自适应方式[9]确定：

$$Cr = \begin{cases} Cr_{\mathrm{l}} + (Cr_{\mathrm{u}} - Cr_{\mathrm{l}}) \cdot \dfrac{t}{T}, & fitness > avgfitness; \\ Cr_{\mathrm{l}}, & \text{other}。 \end{cases} \tag{8}$$

T、t 表示总迭代次数与当前迭代次数；Cr_{u} 表示自适应交叉参数的最大值；Cr_{l} 表示最小值。Cr 取值总体趋势为在进化初期，取较小值，保证种群多样性；在进化后期，Cr 取较大值，加强算法局部搜索能力。

3.3　双策略变异策略的设计

变异操作确定算法的进化方向、收敛性、探索空间。目前关于对差分进化算法的改进主要集中在变异策略方面。选取两种基本变异策略：

$$v_g^i = x_g^{\mathrm{best}} + F \cdot (x_g^{r1} - x_g^{r2}) ; \tag{9}$$

$$v_g^i = x_g^{r1} + F \cdot (x_g^{r2} - x_g^{r3})。 \quad (10)$$

式中：v_g^i表示第 g 代时，个体的基因向量；指数 i，$r1$，$r2$，$r3$，$best \in (1, 2, \cdots, NP)$ 表示在种群中随机选取不重复个体基因参与变异得到新基因；$x_g^{rx} - x_g^{ry}$表示两个个体的差异向量；F 表示变异参数，采用自适应方式确定，变异参数控制着整个变异过程。

为充分发挥种群多样性优势、提高算法收敛速度、精度。整个种群采用分组策略，分为优势与劣势种群：优势种群采用式(9)的变异策略，即加快其收敛速度的变异策略；劣势种群采用式(10)变异策略。由于整体适应性差，需要加强变异操作增加随机扰动的机率。分组策略如下：

1）种群初始化时，得到整个种群平均适应度值，个体适应度值大于平均适应度值则为优势种群，反之为劣势种群。

2）初始化之后的每一代。在得到上一代优势、劣势种群的平均适应度值 $fit1$、$fit2$ 后，求得当前每一个体适应度值与 $fit1$、$fit2$ 的差值 Δfit，将该个体迁移到差值小的种群中，更新优、劣势种群。

3）整个分组操作直到满足截止条件，否则每代迁移个体，更新优、劣势种群。

3.4 具体算法

首先对整个种群初始化，随机生成个体，依据不同个体的适应度值将整个种群分为优、劣两个种群，并采用不同的变异策略。变异过程、交叉过程分别采用自适应变异参数、交叉参数调节算法的收敛速度与探索广度。算法具体步骤如下：

1）初始化规模为 NP 的种群。其中个体基因即为虚拟机与服务器之间的映射关系，并采用第 2 节所述约束条件检测并修正。

2）种群分组与变异操作，根据种群 NP 的平均适应度值，将种群划分成优、劣两个子种群(两种群采用的变异策略方式见 3.3 节)。在此过程中，变异采用自适应变异参数(式 7)，变异产生的新个体基因进行约束条件检测并修正。

3）交叉操作，确保个体基因中至少存在一个维度值来源于变异操作产生的基因。在此过程中，采用自适应交叉参数(式 8)，交叉产生的新个体基因进行约束条件检测并修正。

4）选择操作，评价函数通过综合能耗优化模型(式 6)确定，采用一级能耗最小选择，二级对一级结果进行过滤，选择活动服务器数量最小的映射关系，三级对二级映射关系进行过滤，选择 SLA 违背率最低的映射关系进入下一代的基因。

5）判断是否停止迭代，不满足则回到步骤 2)，往下反复执行。

6）选取种群最优个体基因，即为最佳虚拟机服务器映射关系。

4 实验与结果分析

4.1 实验配置

本次实验选择具有开源特性的 CloudSim 云计算仿真平台[10]，在 CloudSim 平台上实现 OpenStack 内置 FF 算法、PSO 算法[4]、DE 算法[8]与本文算法进行比较，并进行性能评估。实验环境为 Eclipse(luna)、jdk 1.7、CloudSim 3.0.1。实验模拟 IDC 中服务器异构特点，随机生成 100 台配置异构的服务器，其配置如表 1 所示。IDC 提供 4 种类型虚拟机，其配置见表 2，同时模拟 IDC 实际运行情况，保证服务器都处于运行状态下，分别部署 20、40、60、80、100 台虚拟机，不同数量虚拟机对应各类型数量如表 2 所示。

表 1 服务器配置

Table 1 Server configuration

服务器类型	CPU/MIPS	RAM/MB	Max Power/W	数量
1	1 000	2 048	270	32
2	2 000	2 048	400	22
3	4 000	4 096	700	25
4	6 000	4 096	1 000	21

表 2 虚拟机配置

Table 2 Virtual machine configuration

虚拟机类型	CPU/MIPS	RAM/MB	20	40	60	80	100
1	256	256	4	11	16	22	31
2	512	256	5	11	14	20	24
3	1 024	512	6	10	16	20	24
4	1 024	1 024	5	8	14	18	21

本策略针对 IDC 中服务器与虚拟机都处于实时运行状态时的虚拟机与服务器初始映射关系，并未考虑任务异构对策略的影响。PSO 算法中，惯性权重 w 取 0.9，权重因子 c_1、c_2 均取 2。DE 算法的变异参数 F 取 0.6，交叉参数 Cr 取 0.8。PSO、DE、本文算法都属于生物启发式算法，因此，种

群规模取 100、种群迭代次数取 500，为避免偶然性，重复 50 次实验，取所求数值平均值。本文所有对比实验均在上述实验环境下进行，分析不同算法对于数据中心能耗增加情况与服务质量的影响。

4.2　结果分析

PSO、DE、本文算法在 IDC 中部署 20 台虚拟机时，3 种算法收敛性的比较见图 2。当种群迭代到 100 代左右时，PSO 与本文算法均收敛到最小值，而 DE 算法直到 250 代左右才收敛到最小值，由此可说明本文算法、PSO 收敛速度均优于 DE。从算法探索精度上，本文算法得到的最小值均低于 PSO、DE 算法，因此本文算法收敛速度快、更利于全局最优，同时也能解决算法陷入局部最优情况。

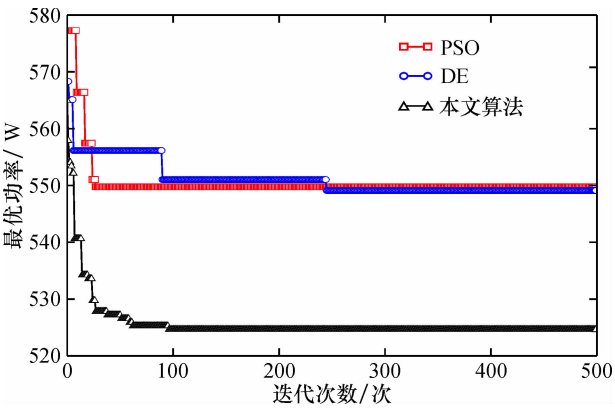


图 2　算法收敛性分析

Fig. 2　Algorithm convergence analysis

PSO、DE、本文算法在批量部署异构虚拟机时，给数据中心额外增加功率方面低于 FF 算法。从算法性质上分析，PSO、DE 算法与本文算法均属于生物启发式算法，而 FF 算法属于传统贪心算法，因此面对批量部署虚拟机时，FF 算法具有单点最优特性，而生物启发式算法具有组合优化特性，面对批量部署异构虚拟机到异构服务器中具有全局优化特性，所以得到图 3 中 PSO、DE、本文算法的 IDC 增加功率均低于 FF 算法结果。

由图 4 可知，本文算法与 FF 算法的活动服务器基本一致，均低于 PSO、DE 算法。FF 算法属于贪心算法范畴，其基本原理是确保虚拟机批量部署到 IDC 中时，使得其活动服务器数量尽量少。而 PSO、DE 算法仅针对能耗最小，并未考虑活动服务器数量问题。本文算法使用式(6)进行功率最小、活动服务器数量最少、SLA 违背率最低的三级过滤方式，因此可在满足较低功率的映射关系下，选择活动服务器数量最少的虚拟机与服务器映射关系。结合图 3、图 4 可知，针对 IDC 异构环境服务器数量并非简单与能耗呈现正相关。

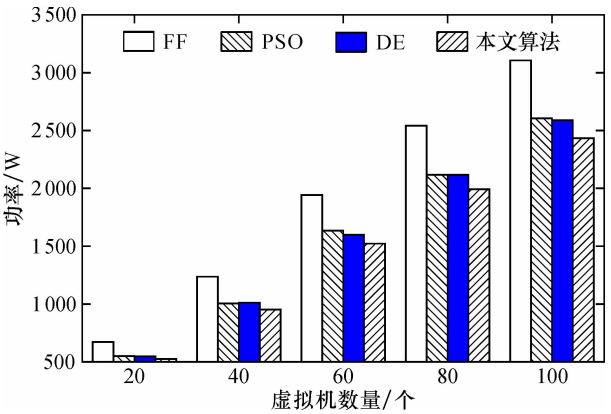


图 3　数据中心增加功率对比

Fig. 3　Comparison of the increased power of data center

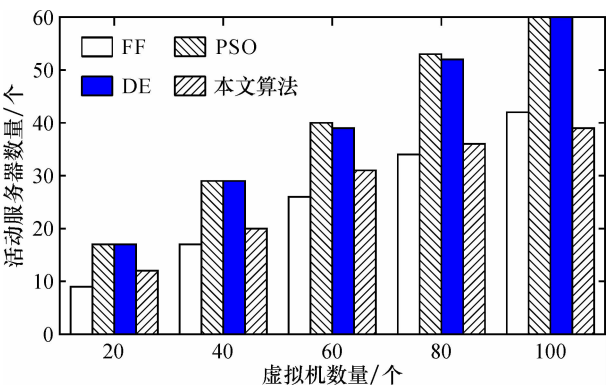


图 4　服务器使用数量对比

Fig. 4　Comparison of numbers of used servers

由图 5 可知，PSO、DE、本文算法的 SLA 违背率均远低于 FF 算法。FF 算法要求将虚拟机部署在已使用服务器中，同时要求服务器剩余资源最小，这种方式针对 IDC 中异构环境使得资源配置低的服务器得到优先集中部署，从而导致服务器的 CPU 使用率超过 0.75 的数量较多。而 PSO、DE、本文算法从 IDC 整体一次性批量部署虚拟机确定映射关系，能够有效减少虚拟机集中部署情形，因此 CPU 使用率过载情形较少发生。由于 IDC 中异构性，虽然服务器数量一致，但其中的服务器类型可能存在巨大差别，造成 FF、本文算法在活动服务器数量在基本一致情况下，SLA 存在巨大差异。

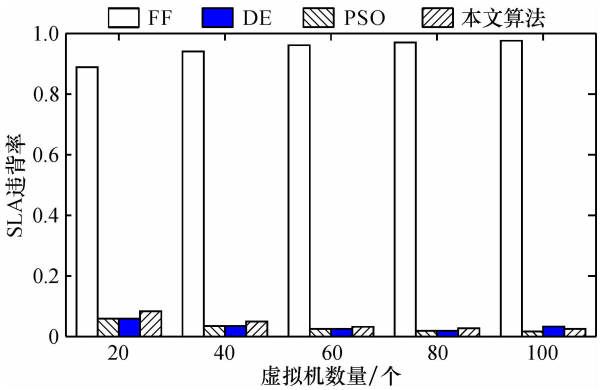


图 5 SLA 违背率情况对比

Fig. 5 Comparison of SLA violation rate

综合可知，本文算法在解决虚拟机初始放置问题满足改进要求，各项指标均有所提升。

5 结束语

本文所提出基于生物启发式算法用于资源配置的虚拟机放置策略主要解决异构数据中心能耗问题，并取得了良好效果，但目前并未部署在 OpenStack 平台，期待后期能够实现生产部署。生物启发算法耗时是个重要缺陷，因此可通过将算法并行化，试图减少算法的时间花费。此外，该类属于静态部署问题，还未考虑动态调度问题，期待后续工作能够解决上述问题。

参考文献：

[1] Garg S K, Buyya R. Green cloud computing and environmental sustainability[M]//Murugesan S, GangadharanG R. Harnessing Green IT: Principles and Practices. UK: Wiley Press, 2012: 315 – 340.

[2] 邓志龙，段哲民，李刘涛. OpenStack 环境下的资源动态调度研究[J]. 西北工业大学学报，2016，34(4):650 – 655.

[3] 许波，陈珂，朱兴统，等. 云计算中基于能耗优化的虚拟机多目标放置算法 [J]. 小型微型计算机系统，2014，35(6)：1304 – 1308.

[4] 豆育升，崔晟圆，唐红，等. 云数据中心高能效的虚拟机放置算法 [J]. 小型微型计算机系统，2014，35 (11)：2543 – 2547.

[5] Xue J, Li L T, Zhao S S, et al. A study of task scheduling based on differential evolution agorithm in cloud computing [C] //2014 Sixth International Conference on Computational Intellingence and Communication Networks, 2014.

[6] 董丽丽，黄贲，介军. 云计算中基于差分进化算法的任务调度研究[J]. 计算机工程与应用，2014，50(5):90 – 95.

[7] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of datacenters for Cloud computing [J]. Future Generation Computer Systems, 2012, 28 (12): 755 – 768.

[8] Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization, 1997, 11 (4): 341 – 359.

[9] 吴亮红. 差分进化算法及其应用研究 [D]. 长沙：湖南大学，2007.

[10] Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers [J]. Concurrency and Computation: Practice and Experience, 2012, 24 (13): 1397 – 1420.

Virtual machine placement strategy for cloud computing based on OpenStack

LUO Ping[a,b], WANG Yong[b,c], FENG Hao[b,c], HE Qian[b,c]

(a. School of Information and Communication; b. Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems; c. School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: Current scheduling strategies for virtual machines on OpenStack cloud platforms do not focus on high energy consumptio. Large number of servers, and high rate of service level agreement (SLA) violation were brought about by server energy consumption, and computing performance. This paper constructs an energy efficiency optimization model to integrate these three factors. Based on the model, a virtual machine placement strategy based on adaptive dual-strategy differential evolution algorithm is designed. This new strategy includes modifying differential evolution algorithm (DE) by citing different mutations way in different groups and self-adaption factors of mutation operation and cross operation. It can improve the convergence speed, and evaluate evolutionary individuals by integrated energy efficiency model as the algorithm fitness function. The simulation results show that the strategy has lower energy consumption and SLA violation rate, less number of servers used than built-in algorithms, DE algorithms, and PSO algorithms.

Key words: cloud computing; OpenStack; energy consumption; virtual machine placement; differential evolution algorithm