云计算中虚拟机放置的自适应管理与多目标优化

李 强[1),2),3)] 郝沁汾[3)] 肖利民[2),3)] 李舟军[3)]

[1)](湖南师范大学数学与计算机科学学院高性能计算与随机信息处理省部共建教育部重点实验室 长沙 410081)

[2)](北京航空航天大学软件开发环境国家重点实验室 北京 100191)

[3)](北京航空航天大学计算机学院 北京 100191)

摘 要 云计算的一个关键需求是其基础设施中大规模虚拟机的放置问题.虚拟机和物理结点之间的映射决定了如何将云计算中虚拟化资源分配给多个 Web 应用,对云计算系统的性能、能耗和 QoS 保证有重要影响.文中提出了云计算中虚拟机放置的自适应管理框架,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,用于制定框架中的虚拟机放置策略.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,根据不同染色体长度的变化设计交叉和变异遗传算子.算法对解空间内的多个区域同时搜索,具有群体和自我进化的优势,优化一次就能获得对不同目标的权值运算多次才能得到的最优解.实验结果表明,与传统的启发式和单目标优化算法相比,提出的框架及算法使得多个应用的服务级目标的违背率最低,且能有效减少虚拟机迁移次数和物理结点的使用数量.

关键词 云计算;虚拟化;虚拟机放置;自适应资源管理;遗传算法

中图法分类号 TP312 **DOI 号**: 10.3724/SP.J.1016.2011.02253

Adaptive Management and Multi-Objective Optimization for Virtual Machine Placement in Cloud Computing

LI Qiang[1),2),3)] HAO Qin-Fen[3)] XIAO Li-Min[2),3)] LI Zhou-Jun[3)]

[1)](*Key Laboratory of HPCSIP, Ministry of Education of China, College of Mathematics and Computer Science, Hunan Normal University, Changsha* 410081)

[2)](*State Key Laboratory of Software Development Environment, Beihang University, Beijing* 100191)

[3)](*School of Computer Science and Engineering, Beihang University, Beijing* 100191)

Abstract Virtual machine placement in the cloud infrastructure is an important problem that remains to be effectively addressed. The mapping problem between virtual machines and physical nodes is to decide how to allocate virtualized resources on the cloud to many Web applications, thus it greatly impacts on the performance, cost and QoS guaranteed service. An adaptive management framework for virtual machine placement in the cloud is proposed. A multi-objective optimization genetic algorithm is presented to determine placement strategy in the framework, which is subject to application service level objects (SLOs) constraint. It encodes the chromosome using the group method, and crossover and mutation operations deal with the chromosome which length is varying. It decodes the chromosome using three-dimensional split method. The experimental results show that, the proposed solution could effectively reduce the number of used nodes and virtual machine migration, and minimize violation of many application SLOs, compared with traditional heuristic methods and single objective solution.

收稿日期:2011-05-09;最终修改稿收到日期:2011-10-16.本课题得到国家自然科学基金(60973007,60973008)、国家教育部博士点专项基金(20101102110018)、软件开发环境国家重点实验室探索性自主研究课题(SKLSDE-2009ZX-01)、中央高校基本科研业务费专项资金资助(YWF-10-02-058)、湖南省教育厅资助科研项目(10C0937)资助.**李 强**,男,1979 年生,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为虚拟化与云计算.E-mail: qianglee2004@gmail.com.**郝沁汾**,男,1969 年生,博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为计算机体系结构、高性能计算.**肖利民**,男,1970 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为计算机体系结构、高性能计算、虚拟化与云计算.**李舟军**,男,1963 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为安全协议形式化分析、进程代数理论、数据挖掘.

Keywords cloud computing; virtualization; virtual machine placement; adaptive resource management; genetic algorithm

1 引 言

近年来,随着信息系统的日益复杂化和网络应用服务的急速膨胀,迫切要求对 IT 基础架构进行集成和整合,便于集中监控和管理,不断降低总体拥有成本.在此背景下,计算机工业界和学术界提出了云计算的概念[1-2].目前,大多数系统的操作、配置、控制和管理基本停留在人工阶段.企业 IT 预算的 90%以上用于系统的管理和维护[3].随着云计算环境下网络应用的大规模部署和复杂性不断增长,IT 管理人员的工作变得越来越困难.IT 管理人员能力的提高和人数配置的增加都非常有限,使得这一问题更加严重.减少云计算系统管理的人工干涉,让系统可以根据 IT 管理人员制定的策略管理系统本身,实现系统的自适应管理,从而维护云计算的高性能和高可用性,这主要面临以下几个方面的问题:理解应用的负载、理解不同负载下应用的性能及虚拟化资源之间的关系以及虚拟机放置优化.

当应用负载的动态变化导致物理结点的资源无法满足所有应用的服务级目标时,可以通过动态迁移[4]该结点上的某个或几个虚拟机到其他结点来解决.在云基础设施服务(Infrastructure as a Service, IaaS)中,云基础设施提供者为 Internet 应用提供者分配计算资源,两者之间签定并遵从服务级合约(Service Level Agreement,SLA).服务级合约包含一个或多个服务级目标.服务级目标指 Internet 应用提供者向网络用户保证的应用性能指标.云基础设施提供者应该分配适合的资源以满足应用的服务级目标,提高资源利用率并减少不必要的开销."一个 Web 应用的吞吐量不低于每秒 100 个请求"是服务级目标的一个例子.服务级目标包括 3 个部分:应用的性能指标类型(例如,吞吐量)、界限(例如,100 个请求/s)和关系操作符(例如,不低于).在云计算中需要考虑如何合理地放置虚拟机到相应结点,在满足不同应用的服务级目标的同时,实现资源使用的最优化.可将虚拟机放置描述为向量装箱问题.装入的物品是正在运行着的虚拟机,虚拟机所用资源是物品的大小,是可变的.箱子是物理结点,箱子容量是结点资源的使用阈值.资源包括 CPU、内存、磁盘和网络带宽等.资源的种类数即向量装箱问题的维度.假设物理结点的数目为 M,虚拟机的数目为 N,虚拟机部署到物理结点的解空间为 M^N,是一个类似于装箱问题的 NP-hard 问题[5].需要找到一个近似优化解.

研究者大多采用启发式方法来进行全局优化搜索,解决虚拟机重新部署下的约束满足问题[6-8].启发式方法基本是单点搜索,很容易陷入局部最优解.此外,云计算中大量应用的负载、虚拟化资源和性能指标的数据需要测量与统计.而且,虚拟机迁移会花费一定的时间,并和结点上的其它活动产生相互影响,导致迁移时延的增加和应用性能的下降.实现迁移还需要额外的网络存储系统和高速的通信网络.如果虚拟机在结点间的迁移次数比较多,将导致无法满足多个应用的服务级目标和云计算系统的不稳定.

针对上述问题,本文通过构建典型 Web 应用的长期负载性能模型,分析云计算范围下的资源需求分布,提出云计算基础实施中虚拟机放置的自适应管理框架,将虚拟机放置定义为一个带多个约束的多目标优化问题,提出基于 NSGA-II 遗传算法的带应用服务级目标约束的虚拟机放置多目标优化算法.算法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,根据不同染色体长度的变化设计交叉和变异遗传算子.算法对解空间内的多个区域同时搜索,具有群体和自我进化的优势,优化一次就能获得对不同目标的权值运算多次才能得到的最优解.实验结果表明,与传统的启发式和单目标优化算法相比,本文提出的虚拟机放置的自适应管理框架和算法使得多个应用的服务级目标的违背率最低,且能有效减少虚拟机迁移次数和物理结点的使用数量.

本文第 2 节介绍相关工作;第 3 节描述虚拟机放置的自适应管理框架;第 4 节给出长期负载性能模型;第 5 节提出虚拟机放置多目标优化遗传算法;第 6 节对基于多目标优化遗传算法的自适应管理框架进行实验评测;最后总结全文并对下一步工作进行展望.

2 相关工作

在云计算基础设施中,如何合理地确定虚拟机到结点的映射是一个装箱问题,即寻找最优的将虚拟机分配到结点的方案,从而使每个结点中虚拟机的使用资源之和不超过结点所能提供的上限,而使

用的结点数量最少. 装箱问题属于 NP-hard 问题，目前还没有多项式最优解算法，一般采用一些启发式算法. 大多数启发式算法基于贪心方法，并使用一些简单规则，例如，次优配合、优先配合或最佳配合等. 目前，一般通过改进传统启发式算法来搜索虚拟机优化放置的最优解[7-10]. 但是启发式算法基本是单点搜索，很容易陷入局部最优解，局部优化并不总是能获得全局优化解，并且可能完全无法产生任何解.

装箱问题及其变体作为 NP-hard 问题得到了大量的研究[11]. 但是，解决向量装箱问题和带约束的向量装箱问题并应用到实际生产当中的工作还比较少[12]. 文献[13]从理论上探讨了在集群中部署大规模分布式应用数量最大化的问题，但只考虑了 CPU 一个维度. 文献[14]给出了根据资源需求变化动态地将应用实例部署到服务器的在线算法. 这些工作没有引入虚拟化技术，是以应用而不是以虚拟机为放置单位. 基于最低过载负载平衡算法，文献[7]提出了一种改进的优先配合降序方法来解决结点装箱问题. 该方法只涉及负载峰值情况下的结点整合，并且没有考虑物品和箱子不相容的约束.

以上研究都只考虑与向量装箱问题相关的容量约束，而没有对结点整合过程中存在的不相容约束建模. 当不能将两个应用同时分配到同一个目标结点时，则发生项-项不相容约束. 当不能将一个已运行的应用迁移到某个指定的目标结点时，则发生项-箱子不相容约束. 文献[8]提出一个解决带项-项和项-箱子不相容约束的结点整合问题的两阶段启发式算法. 在第一阶段，为应用找到满足项-项或项-箱子不相容约束的集群，假设这样的每个集群容量是无限制的. 在第二阶段，根据目标结点的实际容量，将上述集群中的应用分配到该结点. 由于启发式方法容易陷入局部最优而很难获得全局优化解，当大多数应用只出现在一个项-箱子不相容约束中时，这种方法性能比较差. 文献[9]给出了一种组基因算法来提高全局搜索能力，从而克服上述不足.

文献[7-9]的前提是负载是确定性的，用于构建模型的输入负载都是基于应用的负载峰值，导致大多数时期提供的资源过多. 文献[15]将虚拟化数据中心的结点整合描述为随机装箱优化问题来处理负载的随机性，只考虑了 CPU 资源，没有将内存作为一种约束来处理.

能耗管理是虚拟化数据中心和云计算的一个研究热点. 应用动态的虚拟机迁移能力来提高硬件等资源的利用率，降低系统的能源消耗. 文献[16-17]将虚拟机动态迁移描述为优化问题，优化目标是能源消耗最小化.

目前，云计算中虚拟机放置研究工作大多只涉及某一方面的优化，或者是考虑服务级目标的保证，或者是使用结点的数量最少，或者是降低虚拟机迁移代价，或者是减少能源消耗等. 但是这些目标的优化是相互冲突的. 为了减少结点的使用数，将虚拟机更紧凑地放置到数量更少的结点上，从而将闲置的结点关闭来节省能耗和管理开销，需要通过次数更多的虚拟机迁移来实现. 如果是以减少迁移为目标，则可能会增加结点的使用个数. 有效的策略应该考虑所有这些目标并进行权衡和折衷. 目前很少工作将服务级目标保证、结点使用数量和虚拟机迁移次数综合进行考虑来实现虚拟机放置的多目标优化.

文献[18]将问题分解成装箱的组合优化问题和多目标优化问题两部分. 首先采用遗传算法处理虚拟机放置到结点的组合优化问题，然后结合模糊逻辑优化多个目标，包括总体资源浪费、能源消耗和热量消散代价最小化. 但是没有考虑虚拟机迁移的开销，并且只考虑了静态的虚拟机放置，而没有涉及到基于虚拟机迁移的动态放置.

目前大部分虚拟机放置优化方法都是将多目标优化问题转化为若干个单目标优化问题分阶段来解决，很少是同时对多个目标进行优化的，大多数时候只能获得局部的而不是全局优化解.

3　虚拟机放置的自适应管理框架

典型 Web 应用服务的负载在大多数时间段呈现很明显的周期性变化，同时也存在着不可预见的短时期的负载高峰变化[19]. 另外，云计算基础设施通常由数以万计的结点组成. 因此，虚拟化云计算的大规模和复杂性使得建立自适应管理的模型和架构非常困难，需要从不同的时间范围和空间规模分层次进行处理. 我们在文献[20]中对于短期内不可预见的、突发性的负载采取被动方式，提出基于反馈控制的自适应虚拟化资源管理方法，通过反馈控制及时调整单个结点上的虚拟化资源，以快速响应负载变化，满足应用的服务级目标. 大规模的虚拟机迁移会导致极大的系统开销，并需要花费较长的时间来完成全部的迁移，在整个云计算基础设施规模下频繁地进行虚拟机重放置，实践上很难有好的效果. 因此，按空间大小和时间长短将虚拟机放置分为两种类型：迁移域和全局的. 我们在文献[20]中通过将相

互距离比较近的一些结点组成一个迁移域,把云计算基础设施划分为多个迁移域,提出一个改进的迁移域最佳配合降序算法,对迁移域内的虚拟机进行重放置.

本文对于长期的、可预测的、周期性的负载采取主动方式,使用统计和机器学习方法分析大量的负载变化统计数据和系统日志,建立长期负载模式下的性能模型,为实现大规模虚拟机放置的多目标优化提供决策支持,并设计实现虚拟机放置的自适应管理框架.

虚拟机放置的自适应管理框架如图 1 所示.框架包括 4 个模块.其中,全局监控器模块周期性地接收每个虚拟机上的资源使用监控器发来的虚拟机资源使用情况、每个虚拟机上的性能监控器发来的应用性能指标、来自结点控制器的虚拟机迁移请求及每个结点的资源使用和可用情况.然后将这些信息发送给策略生成器.在每个全局控制循环周期(小时或天时间级)中,通过模拟的方法,策略生成器分析每个应用的当前和历史数据来预测将来的负载并建立长期负载性能模型.通过长期负载性能模型预测结点的资源需求并判断结点是否过载或低载等.根据以上预测和判断结果,通过一个多目标优化算法获得全局范围虚拟机放置的最优解,并结合来自结点控制器的迁移请求制定相应的放置策略.基于放置策略,迁移规划器模块制定虚拟机部署到结点的配置方案和相应的迁移规划.迁移驱动器模块根据迁移规划执行虚拟机迁移并尽可能降低对应用性能的影响.

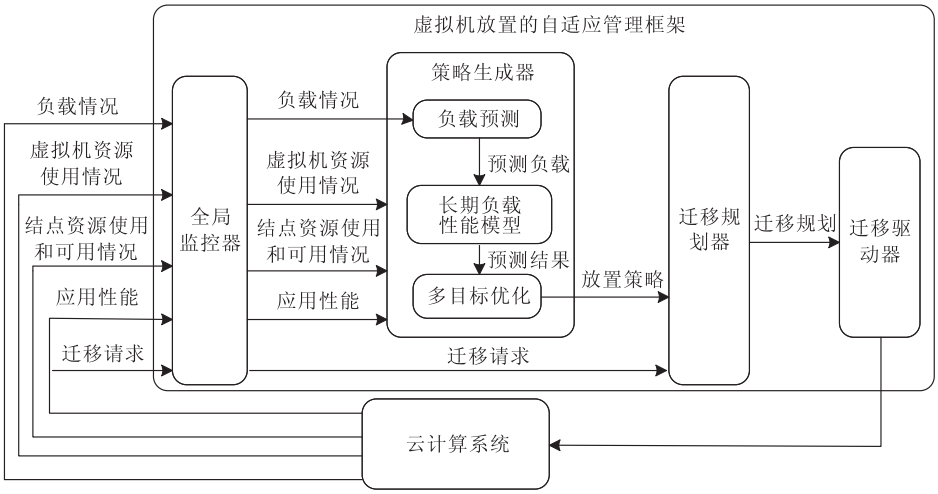

图 1 虚拟机放置的自适应管理框架

放置策略的目的是通过改变全局虚拟机的布局和配置来满足多个应用的服务级目标并优化使用结点的数量和虚拟机迁移次数.虚拟机放置的自适应管理需要做出几个决定:什么时候、迁移哪些虚拟机及将这些虚拟机迁移到哪个结点.什么时候迁移相当于确定结点什么时候过载或低载,从而才能将虚拟机整合以减少结点的使用个数.我们在文献[20]中给出的虚拟机控制器、结点控制器和迁移域控制器可以被动地调整结点资源和迁移虚拟机来处理短期内突发的负载变化.虚拟机控制器中的应用性能与所需资源关系模型刻画了短时期内在虚拟机中满足一定服务级目标时负载和资源之间的关系.另一方面,为预测长时期虚拟机(结点)是否能处理某个特殊的负载而满足服务级目标,基于实验获得的数据集,应用统计和机器学习技术建立准确的系统长期负载性能模型.这两种模型可用于量化虚拟机和结点在不同时间和空间范围的过载或低载.

本文的虚拟机放置的自适应管理框架通过主动地调整全体虚拟机的布局来处理长期内可预测的负载变化.因此,在每个全局控制循环周期初始决定是否需要进行重放置.通过负载预测和长期负载性能模型可以判定哪些结点过载或低载,从而得出需要重放置的规模.如果规模较大,超出了迁移域的处理能力,则启动虚拟机放置策略的生成.第 5 节提出的多目标优化算法确定迁移哪些虚拟机到哪些结点上.框架的 4 个基本模块的实现可能无法适合所有的 Web 系统,可以替换其中的每个模块而不影响其它模块,模块间的耦合性比较弱.

4 长期负载性能模型

本节对 Web 应用长时期的负载进行分析和预

测，并建立相应的性能模型，以此预测一个结点是否能处理某个特定的负载，从而不违背应用的服务级目标. 全局虚拟机放置策略结合负载预测和模型来检测出哪些结点过载或低载，并决定将过载或低载结点上的虚拟机迁移到哪些结点上. 为建立长期负载性能模型，我们设计了几组实验. 在实验中，按照一定的时间间隔对应用的请求数进行取样，样值表示这段时间的负载大小. 通过这种处理过程得到一个负载频率曲线，表示应用请求率的长时期变化. 为了使全局虚拟机放置策略不对负载中细小的高峰作出响应，对负载频率曲线进行一定的处理. 这里只考虑负载的周期性部分，所以将负载中无法预见的突发的部分去除掉. 长期负载性能模型的参考输入采用预测的负载，这样能够通过在结点间迁移虚拟机来直接控制每个结点的负载.

在实验中，负载输入为 HTTP 请求，类型主要是 GET 和 POST 方法. 采用 RUBiS[21] 作为 Web 服务器处理 HTTP 请求. 测试程序线性的增长负载，范围从 100 个请求/s 到 800 个请求/s. 服务级目标定为请求响应时间 200 ms. 使用几种不同的混合 GET 和 POST 的负载：95% GET 和 5% POST，85% GET 和 15% POST，75% GET 和 25% POST. 这些混合负载类似于在 EPA-HTTP① 基准测试上使用的负载. 对每一种混合负载，收集数据集 $D=\{(w,f)\}$. 其中，w 表示 30 s 时间间隔期间结点上的负载率，即 30 s 内的 HTTP 请求的平均数；f 表示在相同时间间隔期间响应时间大于 200 ms 的请求部分.

图 2 和图 3 表明了两种混合负载下结点的性能变化. 为使负载和不满足指定请求响应时间部分的依赖关系具体化，以 20 请求/s 的宽度将 w 组成块，针对每个块计算 f 的期望和标准方差. 如图 2 和图 3 所示，横坐标表示负载，即每秒的请求数. 纵坐标表示请求的响应时间. 水平虚线段表示指定的服务级目标 200 ms 的请求响应时间. 对于每一个负载，响应时间的期望用点表示，标准方差用小节线表示. 95% GET 和 5% POST 配置下，当请求在 500 请求/s 到 600 请求/s 时，性能急剧下降. 而在 85% GET 和 15% POST 配置下，当请求在 460 请求/s 到 550 请求/s 时，性能急剧下降.

将数据集 $D=\{(w,f)\}$ 作为训练集，采用 logistic 回归生成系统性能的线性分类模型. 模型使用两个输入特征，HTTP 的 GET 和 POST 请求率. 通过将每个负载块的响应时间离散化为高值或低值，获得二分变量. 高值或低值基于指定的服务级目标的响应时间（图 2 和图 3 中的水平虚线段）. 图 4 给出了最终的二分数据集和匹配的线性 logistic 回归模型. 在图 4 中，横坐标表示 GET 方法的请求率. 纵坐标表示 POST 方法的请求率. 小的实心圆表示结点能处理负载，满足服务级目标. 大的实心方块表示结点不能处理负载，违背服务级目标. 3 条射线段表示 3 种不同的混合负载，从右下到左上依次为 95% GET 和 5% POST，85% GET 和 15% POST，75% GET 和 25% POST. 根据匹配模型，粗斜线段表示可以和不可以接受性能之间的分界线. 在不可以接受性能的区域内，应用所在的结点过载，无法保证应

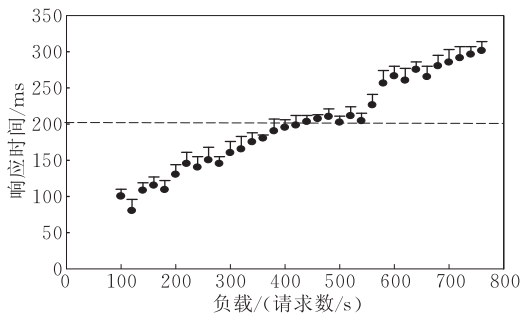

图 2　95% GET 和 5% POST 类型负载下的性能指标

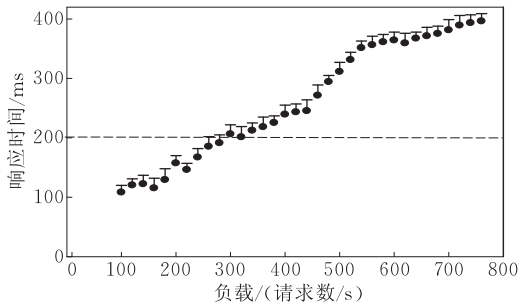

图 3　85% GET 和 15% POST 类型负载下的性能指标

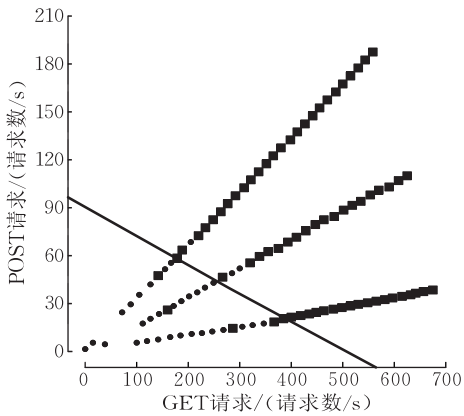

图 4　长期负载性能模型

① EPA-HTTP-a day of HTTP logs from EPA WWW server. http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html

用的服务级目标.以此可以预判需要对哪些虚拟机进行迁移.长期负载性能模型可以预测一个结点的负载变化以及处理特定负载的能力.为全局控制循环中虚拟机放置策略的制定提供支持.

5 虚拟机放置多目标优化遗传算法

5.1 多目标虚拟机放置问题的描述

多目标虚拟机放置问题是个组合优化问题,同时也是个多目标优化问题.可将虚拟机放置问题看作为多维装箱问题.每个物理结点的可用资源(箱子)是一个 d 维向量,每一维表示某种资源(如CPU、内存、磁盘和网络带宽等).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)放入多个物理结点(箱子),并使得物理结点的个数最少及虚拟机迁移的次数(物品移动的次数)最少.多目标虚拟机放置问题的描述如下:

目标:

$$\min\sum_j y_j \text{ 并且 } \min\sum_i m_i$$

$\min\sum_j y_j$ 表示物理结点的使用数量最少.$\min\sum_i m_i$ 表示虚拟机迁移次数最少.

约束:

$$y_j \in \{0,1\} \tag{1}$$

如果 $y_j=1$,则使用了物理结点 j.

$$m_i \in \{0,1\} \tag{2}$$

如果 $m_i=1$,则迁移了虚拟机 i.

$$x_j^{pi} \in \{0,1\} \tag{3}$$

假设多层 Web 应用的一层(例如,表现层、逻辑层、数据层)封装在一个虚拟机中.如果 $x_j^{pi}=1$,则将应用 p 中的某一层所在的虚拟机 i 放置在物理结点 j 上.

$$\sum_j x_j^{pi}=1,\ \forall i,p \tag{4}$$

应用 p 的某一层所在的虚拟机 i 只能放置在某个物理结点 j 上.

$$\sum_p\sum_i r_{pi}^{\mathrm{CPU}}\cdot x_j^{pi}\le c_j^{\mathrm{CPU}},\ \sum_p\sum_i r_{pi}^{\mathrm{MEM}}\cdot x_j^{pi}\le c_j^{\mathrm{MEM}},\ \sum_p\sum_i r_{pi}^{\mathrm{I/O}}\cdot x_j^{pi}\le c_j^{\mathrm{I/O}},\ \forall j \tag{5}$$

多个虚拟机放置在某个物理结点 j 时,虚拟机资源不能超过物理结点 j 的资源容量.r_{pi}^{CPU}、r_{pi}^{MEM} 和 $r_{pi}^{\mathrm{I/O}}$ 分别表示应用 p 中的虚拟机 i 使用的 CPU、内存和 I/O 资源.c_j^{CPU}、c_j^{MEM} 和 $c_j^{\mathrm{I/O}}$ 分别表示物理结点 j 的 CPU、内存和 I/O 资源容量.

$$PM_p\ge SLO_p,\ \forall p \tag{6}$$

满足每一个应用的服务级目标.PM_p 表示应用 p 的性能指标,SLO_p 表示应用 p 的服务级目标.

在每个全局控制循环周期中,使用第 4 节的长期负载性能模型预测一个结点的负载变化以及结点处理特定负载的能力,判断结点是否能满足约束(6).虚拟机放置的优化目标是在保证多个应用的服务级目标的情况下,使物理结点的使用数量和虚拟机迁移次数最小.

5.2 虚拟机放置多目标优化遗传算法设计与实现

理论上组合优化问题的最优解可以通过简单枚举得到,但实际上一般无法实现.尤其对于规模巨大的实际问题来说,可行解的数量可能特别大.如何有效处理组合爆炸是解决问题的关键之一.一种重要思路就是采用遗传算法.下面详细描述所提出的虚拟机多目标优化遗传算法的设计步骤和具体的处理过程.

编码

编码的选择是影响遗传算法搜索效果和效率的重要因素.编码实现从问题的解到染色体的映射.在这里,染色体为虚拟机放置到物理结点的解的编码.装箱问题有 3 种遗传编码方法:(1)基于箱子的表示;(2)基于物品的表示;(3)基于组的表示.由于装箱问题的目标函数依赖于物品组,而前两种编码方法是面向单个物品的,都具有很多缺点.文献[22]提出了一种基于组的编码方法,图 5 给出了该方法的

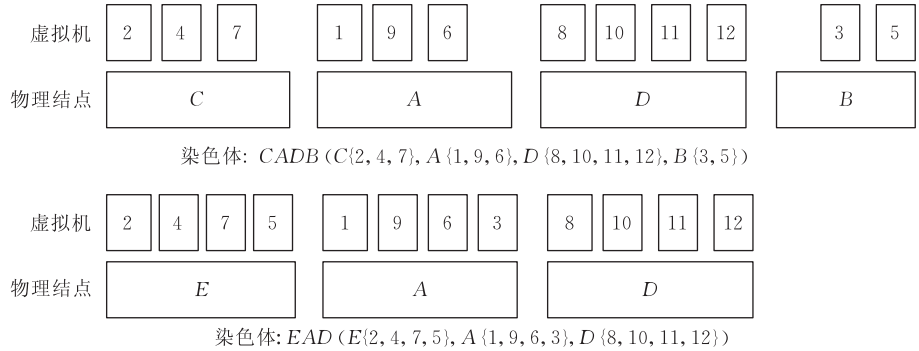

图 5 基于组编码方法的虚拟机放置

一个例子. 12 个虚拟机被分为 4 个组，相对应的染色体有 4 个基因，每个基因包含一组虚拟机，基因既表示物品又表示箱子. 这种编码的关键在于遗传算子对于染色体的组部分进行操作，其物品部分仅用于判定由哪些虚拟机形成该组. 由于物理结点可以放置的虚拟机数目不同，相同的多个虚拟机可能放置到个数不同的物理结点上，因此遗传算子要处理不同长度的染色体. 如图 5 所示，染色体 $CADB$ 和染色体 EAD 长度不同.

译码

基于组的编码得到的染色体字符串是解的基因型表示，需要将其简单、快速地转化为表现型(即虚拟机在结点中的空间布局)，这是能否有效应用遗传算法的关键. 译码采用文献[23]提出的空间分解方法. 在装箱的过程中，采用三空间的分割方法. 这里假设资源类型只有 3 种. 如图 6 所示，当一个虚拟机放入一个物理结点时，将其放置在当前放置空间的左下角，并将放入的虚拟机编号从染色体字符串中删除. 这样，除了放入虚拟机占用的空间外，该物理结点被分割成 3 个空间，即左空间、上空间和右空间. 依次将 3 个空间作为当前空间，重复上述分解过程，直至物理结点没有可使用的空间或没有待放置虚拟机满足要求时停止. 这样，染色体字符串就对应为空间布局图，并克服了空间干涉约束，对字符串进行选择、交叉和变异等算子操作就是改变虚拟机在物理结点上的放置位置，从而产生不同的空间布局图(即不同的解). 采用三叉树数据结构表示布局空间. 当前布局空间为根结点，分割后的剩余 3 个空间为根结点的左、中、右子结点. 文献[23]表明该空间分解方法能有效处理多目标及多约束的问题.

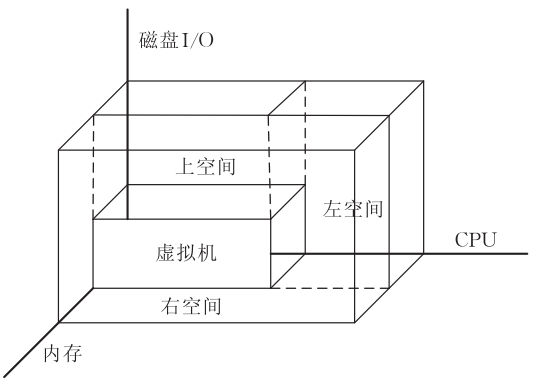

图 6　空间分解策略示意图

初始染色体种群的生成

首先随机地生成 S 个虚拟机请求放置的序列，接着对每一个虚拟机请求放置序列，应用基于优先配合启发式的染色体中将虚拟机放置到物理结点算法，将多个虚拟机放置到多个物理结点上，得到 S 个不同的虚拟机放置方案(S 个染色体)，从而构成初始种群.

在优先配合启发式算法中，第 1 个虚拟机放入第 1 个物理结点，然后是根据下标上升的顺序放入第 2，…，n 个虚拟机，如果当前物理结点的资源足够，则每个虚拟机放入当前物理结点，否则放入下一个新的物理结点. 其时间复杂度为 $O(n\log n)$. 下面详细描述染色体中将虚拟机放置到物理结点算法.

1. 按照虚拟机请求放置序列的顺序逐个选出虚拟机. 找出与所选虚拟机属于同一个应用的其它层的虚拟机. 然后按下述步骤查找能将这些虚拟机分配到其上的物理结点. 若虚拟机请求放置序列为空，则停止.

2. 在已使用的物理结点集合中选取最小下标的物理结点. 若所有已使用的物理结点已选完，则从未使用的物理结点集合中选取最小下标的物理结点. 若所有物理结点都选完，则返回步 1. 初始化时，已使用的物理结点集合为空，未使用的物理结点集合包括全部的可用物理结点.

3. 检查是否满足多目标虚拟机放置问题的约束(1)、(2)、(3)和(4)，若满足，继续；否则，返回步 2.

4. 检查在选出的物理结点上部署的应用的服务级目标是否满足. 检查是否满足多目标虚拟机放置问题的约束(5)和(6)，若满足，则将虚拟机放置到选出的物理结点，并从虚拟机请求放置序列中删除该虚拟机，返回步 1；否则，返回步 2.

适应度

基于经典的第二代进化多目标优化算法——NSGA-II 算法[24]中的方法，首先采用快速非支配排序确定种群中每一个个体的非支配等级，然后通过局部拥挤距离算法对其进行密度估计，确定个体的局部拥挤距离.

选择

采用 2-约束竞赛选择法. 在该选择方法中，从当前种群中随机地挑选两个个体，然后将适应值最大的个体(最好的个体)选做父个体. 重复执行这个过程，直到个体数目达到预定的种群规模. 选出的父个体作为下一代种群. 两个个体的比较方法采用 NSGA-II 引入的拥挤比较算子.

交叉

虚拟机放置到物理结点的问题中，基于组的表示包括两个部分：物理结点和虚拟机. 用组来表示染色体的基因. 组包含的虚拟机个数不是固定的，所以交叉可能需要处理长度可变的染色体. 交叉的步骤如下.

1. 对参与交叉的两个父个体随机地选定交叉部分和交叉点；

2. 将第 1 个父个体交叉部分的内容插入到第 2 个父个

体的交叉点位置之前.交叉是对染色体的虚拟机组进行操作,即从第 1 个父个体插入一个虚拟机组到第 2 个父个体中;

3. 在上一步得到的第 2 个父个体中,某些虚拟机可能会出现两次,删除所有重复出现的虚拟机所在的物理结点;

4. 在上一步的删除过程中,可能会删除一些虚拟机,从而在染色体中不会出现这些虚拟机,采用上述的染色体中将虚拟机放置到物理结点算法重新将虚拟机插入到物理结点,得到第 1 个子个体,如图 7 所示;

5. 互换两个父个体,重新执行步 2 到步 4 生成第 2 个子个体.

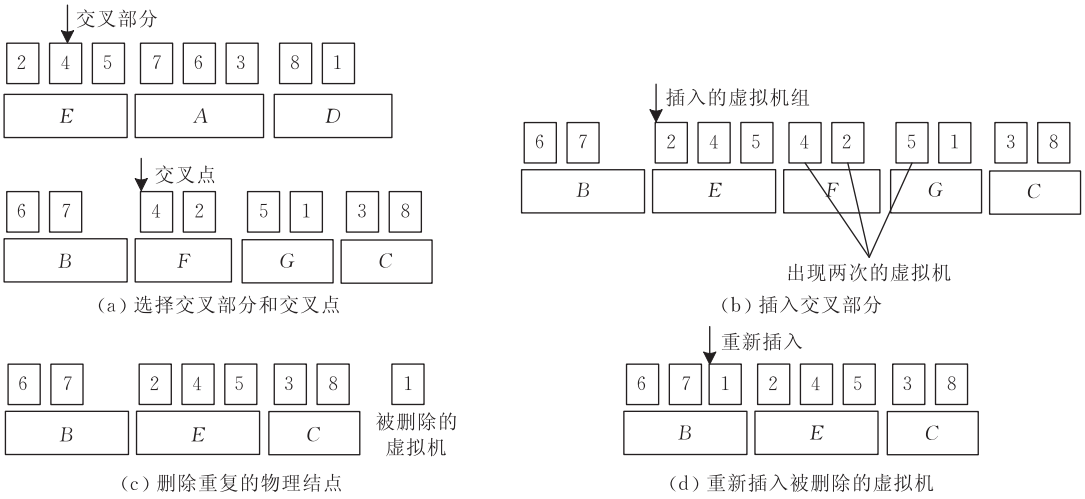

图 7 交叉过程

变异

装箱问题的变异必须是对组而不是物品(虚拟机)进行操作.对父个体随机选择一个虚拟机组,然后将其删除.采用上述的染色体中将虚拟机放置到物理结点算法重新将被删除的虚拟机们插入到物理结点,得到子个体.

评估染色体

采用物理结点的使用个数来评估染色体,从前一个染色体变迁到下一个染色体的虚拟机迁移次数也用来评估染色体.

虚拟机迁移过程中的约束

虚拟机放置方案可能有多个解,需要找到优化解.采用上述优化算法搜索优化解,其中的一个目标是迁移代价最小,即从前一个放置方案迁移到下一个放置方案时,要求涉及到的虚拟机迁移的次数最少.遗传算法的种群并不是表示一个放置方案,而是随机生成的下一个放置方案的多个解.以此为基础,通过种群的不断进化,最终找到满足多个目标的优化解.此外,还要考虑从前一个放置方案到下一个放置方案过程中所碰到的两种迁移约束问题.采用有向图来表示虚拟机的迁移.有向图的一条有向边代表一次虚拟机的迁移,边尾表示迁移的开始,边首表示迁移结束.有向边指定需要迁移的虚拟机及其所需的资源.有向图的结点代表物理结点及其剩余的资源,并且描述物理结点上部署的虚拟机和虚拟机使用的资源.

如图 8 所示,如果将虚拟机 VM1 从物理结点 A 迁移到物理结点 B,将虚拟机 VM3 从物理结点 B 迁移到物理结点 C,使得物理结点 A 上没有运行的虚拟机,可以关闭物理结点 A,从而将原来 3 个物理结点上的多个虚拟机整合到两个物理结点上,节省了一个物理结点 A.但是,当 VM3 还在物理结点 B 上时,由于物理结点 B 上的资源无法满足 VM1 的需求,因此两次迁移不能够并行执行,只能在 VM3 从物理结点 B 迁移到物理结点 C 后,VM1 才能从物理结点 A 迁移到物理结点 B.将这种约束称之为顺序约束.

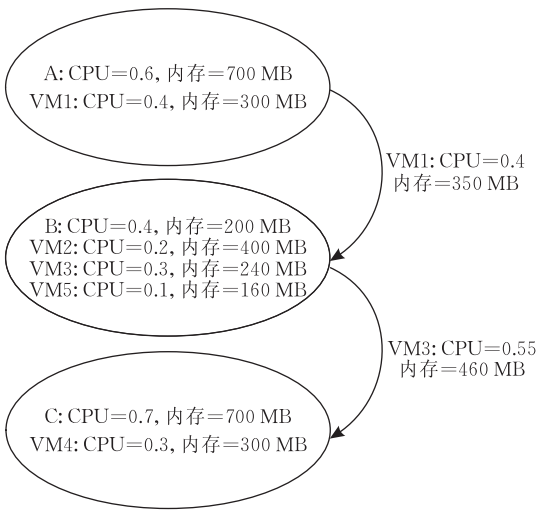

图 8 无法并行执行的迁移

如图 9(a)所示,由于 CPU 的约束,VM1 只有

在 VM2 从物理结点 B 迁移到物理结点 A 之后，才能从从物理结点 A 迁移到物理结点 B. 同时，由于 CPU 的约束，VM2 只有在 VM1 从物理结点 A 迁移到物理结点 B 之后，才能从从物理结点 B 迁移到物理结点 A. 这样就形成了迁移死锁. 将这种约束称之为死锁约束. 引入一个额外的物理结点，临时放置一个或多个虚拟机，打破迁移死锁. 如图 9(b)所示，通过将 VM1 迁移到引入的物理结点 C，然后将 VM2 迁移到物理结点 A，再将 VM1 从物理结点 C 迁移到物理结点 B. 这些迁移步骤只能顺序执行，从而形成顺序约束.

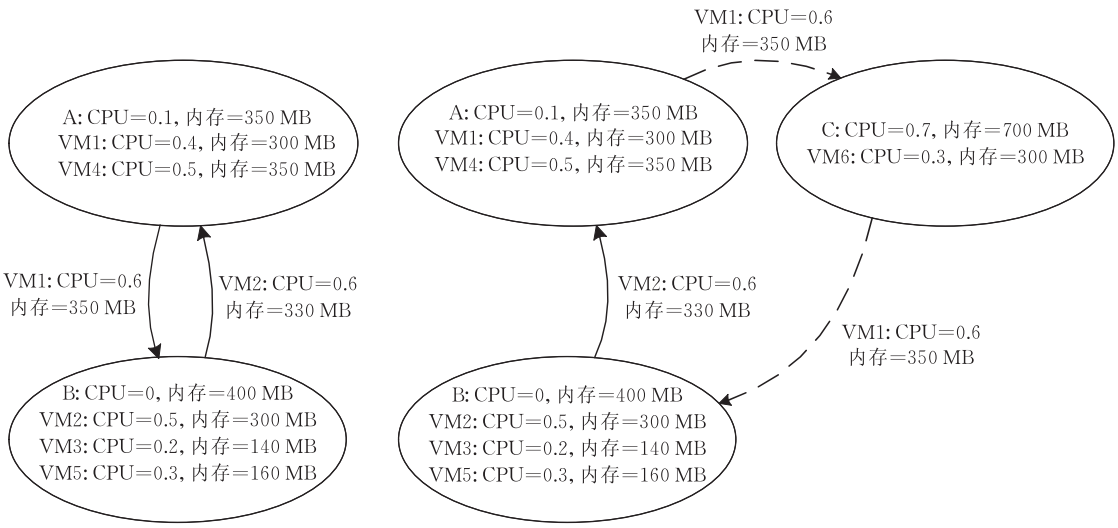

图 9　迁移死锁及处理

顺序迁移和死锁迁移是在从前一个放置方案到下一个放置方案过程中出现的，因此需要进行处理，以提高整体的迁移效率.

6　实验分析

本节通过实验和分析实验结果表明提出的框架和算法能合理地确定虚拟机放置策略，并基于策略迁移虚拟机，具有一定的可行性和准确性. 使用一组模拟实验集对传统的最佳配合启发式装箱算法，单目标遗传算法和虚拟机放置多目标优化遗传算法进行评测，比较它们的性能优劣. 模拟采用从测试实验获得的建模参数，使得结果能准确捕获真实系统的行为.

实验平台包括 26 个物理结点，每个结点装备两个处理器(dual Intel Xeon 3.0 GHz)，4 GB 内存，2 MB L2 Cache，两块 7200 转 250 GB 的 SCSI 磁盘，一块 Intel Pro1000G 网卡. 通过一台千兆以太网交换机连接所有结点. 4 个结点作为客户端向虚拟机发出服务请求，产生负载. 两个结点作为文件服务器，提供基于 NFS 的虚拟机磁盘镜像. 为减少对实验结点的影响，使用一台专用的结点执行算法，结点的配置和实验结点一样.

使用 RUBiS[21] 作为 Web 应用服务的基准测试. RUBiS 的每一层运行在一个虚拟机中，运行一个 RUBiS 实例需要 3 个虚拟机. 所有虚拟机的虚拟 CPU 个数为 1. 运行 EPA-HTTP 基准测试为 RUBiS 应用生成负载，并收集实验结果数据. EPA-HTTP 的跟踪日志包含了位于 Research Triangle Park，NC 的 EPA(美国环保署) WWW server 的超过一整天的所有 HTTP 请求. 在 EPA-HTTP 中，负载输入为 HTTP 请求，类型主要是 GET 和 POST 方法. 重放这些跟踪日志多次来构建一个 50 d 实验的模型. 为了节省时间，通过将 24 h 跟踪日志凝缩为 24 min，并同时维持跟踪日志形状和其它属性，从而将 50 d 的实验凝缩为一个 20 h 的实验. 实验初始时，使用 17 个 RUBiS 实例，产生总共 51 个虚拟机，放置在 19 个结点上.

分别使用传统的最佳配合启发式算法(Best-Fit Heuristic，BFH)、两个单目标遗传算法和提出的多目标优化遗传算法进行实验.

BFH-CPU、BFH-MEM 和 BFH-DISK. 最佳配合启发式算法：第 1 个虚拟机放入第 1 个物理结点，然后将当前虚拟机放入具有最小剩余资源量的可行物理结点中，其时间复杂度为 $O(n\log n)$. BFH-CPU、BFH-MEM 和 BFH-DISK 分别表示在最佳配

合启发式算法中的资源为 CPU、内存和磁盘.

SGA-N 和 SGA-V. 两个单目标遗传算法使用组基因算法[9]来搜索解空间,分别根据所用物理结点的个数(SGA-N)和虚拟机迁移次数(SGA-V)来设置适应度的值.

MOGA. 提出的虚拟机放置多目标优化遗传算法.

使用 Matlab 中 Global Optimization Toolbox① 实现遗传算法. 采用 Python 2.6 编写控制脚本. 单目标遗传算法和多目标遗传算法的参数设置中,交叉概率和变异概率分别设为 0.7 和 0.05,种群个数和遗传代数分别为 22 和 14. 大部分虚拟机迁移的时间范围和空间规模是在迁移域内. 有几次是主动地调整虚拟机的全局放置来处理长期内可预测的负载变化. 选取其中的一次虚拟机重放置进行分析.

图 10 显示了应用虚拟机放置多目标优化遗传算法(MOGA)时优化解的进化过程. 初始种群规模为 22,MOGA 代数分别为 8、14、20 和 25. 从图中可以看出,随着 MOGA 迭代代数的增加,目标函数值(物理结点使用个数和虚拟机迁移次数)的个数逐渐减少,向着最优解的方向移动. 在 14 代前目标函数值变化较大,而之后则变化不大,即使到 25 代还有所改善. 另外,非支配解的个数在逐渐增加,在 14 代时群体共有 22 个个体,其中非支配解的个数有 17 个,解已经基本趋于稳定. 在 19 个物理结点上放置 51 个虚拟机,搜索空间包括 19^{51}(1.646014469E+65)个可能解,而 MOGA 仅通过 $22\times14=308$ 次循环执行遗传算法就能得到 Pareto 前沿面. 这里,MOGA 的时间复杂度为 $O(mN^2)$.

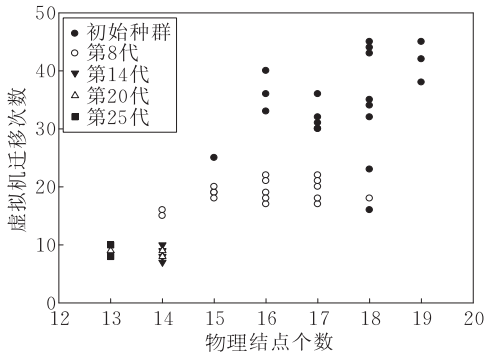

图 10 基于 MOGA 的多目标优化的进化过程

图 11～13 描述了相同实验配置下 5 个不同放置策略的评测结果. 所有策略都使用相同的虚拟机控制器、结点控制器和迁移域控制器,不同之处在于使用的放置策略. BFH 和 SGA-N 由于试图将虚拟机整合到更少的物理结点上,导致虚拟机迁移次数更多和服务级目标违背率高. 与 BFH 相比,SGA-N 能更有效地对解空间进行全局搜索,因此可以使用更少的物理结点来满足多个应用的服务级目标,但是服务级目标违背率相对高些且虚拟机迁移次数相对要少些. SGA-V 不考虑所用物理结点的多少,为满足多个应用的服务级目标,尽可能地将虚拟机平均分布到所有可用的物理结点上,使得每个物理结点的资源使用率和服务级目标违背率都比较低,而使用的物理结点比较多. MOGA 满足服务级目标约束条件下,搜索每一个目标的优化解,在多个相互冲突的目标间实现最优权衡和折衷,服务级目标违背率最低,且产生相对比较少的虚拟机迁移次数和物理结点使用数量.

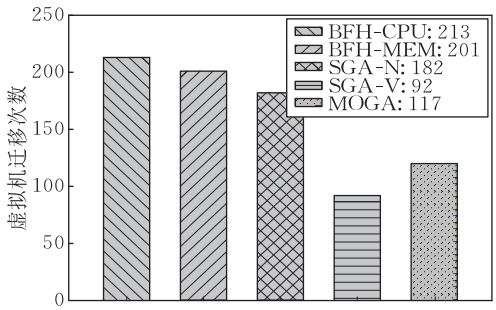

图 11 不同放置策略下的虚拟机迁移次数

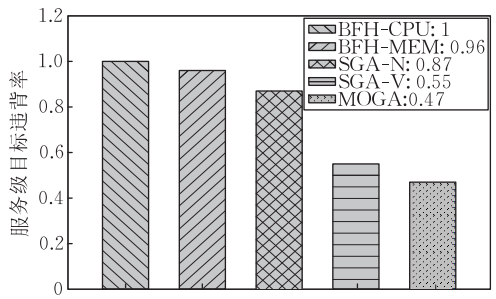

图 12 不同放置策略下的服务级目标违背率

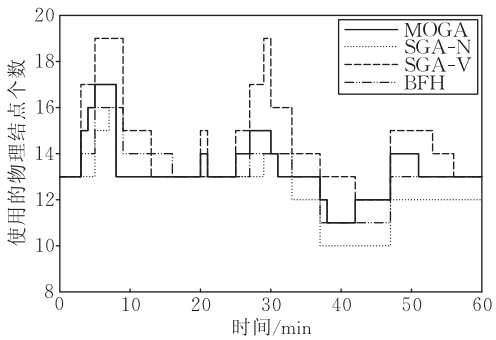

图 13 不同放置策略下的物理结点使用数

① Matlab. Global Optimization Toolbox. www.mathworks.cn/products/global-optimization/index.html?BB=1

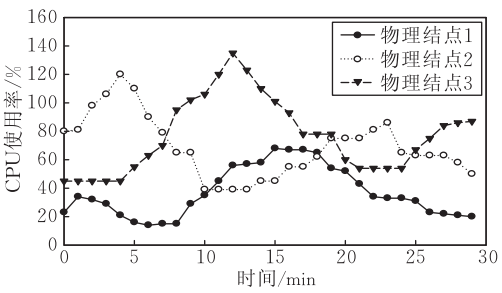

图 14　BFH-CPU 放置策略下 3 个物理结点的 CPU 使用率

为了更细致地分析比较不同放置策略，选取 30 min(凝缩的)时间范围内，3 个物理结点在 BFH-CPU 和 MOGA 下的 CPU 使用率. 如图 14 所示，BFH-CPU 放置策略没有搜索到足够的 CPU 资源来满足变化负载的资源需求，使得物理结点的 CPU 使用率超过 100%(如物理结点 2 和 3)，违背了应用的服务级目标，从而引起该物理结点上的虚拟机迁移. MOGA 放置策略能通过全局搜索找到资源分配的最优解，为负载提供合适的可用资源. 如图 15 所示，所有 3 个结点的 CPU 都没有过载，减少了服务级目标的违背率，避免了虚拟机的迁移.

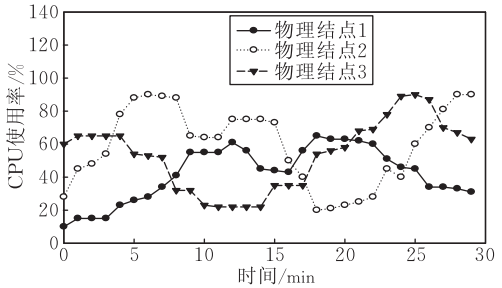

图 15　MOGA 放置策略下 3 个物理结点的 CPU 使用率

7　结束语

本文提出虚拟机放置的自适应管理框架. 研究了用遗传算法对全局虚拟机放置问题进行求解. 在云计算基础设施中，虚拟机放置通常需要在满足一定约束的条件下考虑多个目标的实现. 随着基础设施规模的进一步增大，减少虚拟机迁移次数变得越来越重要. 本章将云计算中的虚拟机放置描述为一个多目标优化问题. 阐述了解决多目标优化问题的方法. 综合考虑虚拟机放置映射中结点使用数量和虚拟机迁移次数两个目标，基于经典的 NSGA-II 算法提出了虚拟机放置多目标优化遗传算法. 该算法利用遗传算法的群体和自我进化的优势，相对于传统的启发式和单目标优化算法，在保证多个应用服务级目标的情况下，能有效地减少结点使用数量和虚拟机迁移次数. 目前，能耗问题[25]是云计算中的重点研究方向之一. 下一步工作将探索如何将资源控制和能耗控制结合起来，实现这些目标的最优化.

参　考　文　献

[1] Armbrust M, Fox A, Griffith R et al. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50-58

[2] Chen Kang, Zheng Wei-Ming. Cloud computing: System instances and current research. Journal of Software, 2009, 20(5): 1337-1348(in Chinese)
(陈康，郑纬民. 云计算：系统实例与研究现状. 软件学报，2009，20(5)：1337-1348)

[3] Patterson D, Brown A, Broadwell P et al. Recovery oriented computing (ROC): Motivation, definition, techniques, and case studies. Berkeley: UC Berkeley, Technical Report: UCB/CSD-02-1175, 2002

[4] Clark C, Fraser K, Hand S et al. Live migration of virtual machines//Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05). Boston, 2005: 273-286

[5] Zhu X, Young D, Watson B. J, Wang Z et al. 1000 Islands: An integrated approach to resource management forvirtualized data centers. Cluster Computing, 2008, 12(1): 45-57

[6] Li Bo, Li Jian-Xin, Huai Jin-Peng et al. EnaCloud: An energy-saving application live placement approach for cloud computing environments//Proceedings of the International Conference on Cloud Computing. Bangalore, 2009: 17-24

[7] Ajiro Y, Tanaka A. Improving packing algorithms for server consolidation//Proceedings of the 33rd International Computer Measurement Group Conference. San Diego, 2007: 399-406

[8] Gupta R, Bose S. K, Sundarrajan S et al. A two stage heuristic algorithm for solving server consolidation problem with item-item and bin-item incompatibility constraints//Proceedings of the 2008 IEEE International Conference on Services Computing (SCC'08). Hawaii, 2008: 39-46

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

[10] Wood T, Shenoy P J, Venkataramani A. Black-box and gray-box strategies for virtual machine migration//Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI'07). Cambridge, MA, 2007: 229-242

[11] Coffman J, Garey M R, Johnson D S. Approximation algorithms for bin packing: A survey. Approximation algorithms for NP-Hard problems. Boston: PWS Publishing, 1997: 46-93

[12] Zhang De-Fu, Wei Li-Jun, Chen Qing-Shan, Chen Huo-Wang. A combinational heuristic algorithm for the three-dimensional packing problem. Journal of Software, 2007, 18(9): 2083-2089(in Chinese)
(张德富，魏丽军，陈青山，陈火旺. 三维装箱问题的组合启发式算法. 软件学报，2007，18(9)：2083-2089)

[13] Urgaonkar B, Rosenberg A, Shenoy P. Application Placement on a cluster of servers. International Journal on Foundations of Computer Science (IJFCS), 2007, 8(5): 1023-1041

[14] Tang C, Steinder M, Spreitzer M et al. A scalable application placement controller for enterprise data centers//Proceedings of the 16th International Conference on World Wide Web (WWW'07). New York, 2007: 331-340

[15] Chen M, Zhangy H, Ya-Yunn S et al. Effective VM sizing in virtualized data centers//Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM'2011). Dublin, 2011: 594-601

[16] Wang Xiao-Rui, Wang Ye-Fu. Coordinating power control and performance management for virtualized server clusters. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(2): 245-259

[17] Jung G, Hiltunen M, Joshi K et al. Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures//Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS'2010). Genoa, 2010: 62-73

[18] Xu J, Fortes J. Multi-objective virtual machine placement in virtualized data center environments//Proceedings of 2010 IEEE/ACM International Conference on Green Computing and Communications (GreenCom'2010). Hangzhou, 2010: 179-188

[19] Zhang H, Jiang Guo-Fei, Yoshihira K. Intelligent workload factoring for a hybrid cloud computing model//Proceedings of the 2009 IEEE Congress on Services. Los Angeles, 2009: 701-708

[20] Li Qiang, Hao Qin-Fen, Xiao Li-Min et al. An integrated approach to automatic management of virtualized resources in cloud environments. The Computer Journal, 2011, 54(6): 905-919

[21] Cecchet E, Chanda A, Elnikety S et al. Performance comparison of middleware architectures for generating dynamic Web content//Proceedings of the 2nd International Middleware Conference. Rio de Janeiro, 2003: 242-261

[22] Falkenauer E, Delchambre A. A genetic algorithm for bin packing and line balancing//Proceedings of the IEEE International Conference on Robotics and Automation. Nice, France, 1992: 1186-1192

[23] He Da-Yong, Zha Jian-Zhong, Jiang Yi-Dong. Research on solution to complex container-loading problem based on genetic algorithm. Journal of Software, 2001, 12(9): 1380-1385(in Chinese)
(何大勇, 查建中, 姜义东. 遗传算法求解复杂集装箱装载问题方法研究. 软件学报, 2001, 12(9): 1380-1385)

[24] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197

[25] Lin Chuang, Tian Yuan, Yao Min. Green network and green evaluation: Mechanism, modeling and evaluation. Chinese Journal of Computers, 2011, 34(4): 593-612(in Chinese)
(林闯, 田源, 姚敏. 绿色网络和绿色评价: 节能机制、模型和评价. 计算机学报, 2011, 34(4): 593-612)

LI Qiang, born in 1979, Ph. D., lecturer. His research interests include virtualization technology and cloud computing.

HAO Qin-Fen, born in 1969, Ph. D., associate professor. His research interests include computer architecture, high performance computing, virtualization and cloud computing.

XIAO Li-Min, born in 1970, Ph. D, professor, Ph. D. supervisor. His research interests include computer architecture, computer system software, high performance computing, virtualization and cloud computing.

LI Zhou-Jun, born in 1963, Ph. D, professor, Ph. D. supervisor. His research interests include concurrency theory and process algebra, formal analysis and verification of security protocols, data mining and bioinformatics.

Background

Cloud computing, as a newly emergent computing environment, promises dynamic flexible infrastructures required to host Internet applications and application service level objectives (SLOs) guaranteed services in a pay-as-you-go manner to the public. However, virtual machine placement in cloud infrastructure is an important problem that remains to be effectively addressed. The mapping problem between virtual machines and physical nodes is to decide how to allocate virtualized resources on the cloud to many Web applications, thus it greatly impacts on both the performance and cost.

At present a lot of research focuses on single objective optimization of virtual machine placement. This paper presents a multi-objective optimization method for virtual machine placement which is subject to application SLOs constraint and based on NSGA-II genetic algorithm. The proposed solution could effectively reduce the number of used nodes and virtual machine migration, and minimize violation of many application SLOs, compared with traditional heuristic methods and single objective solution.

This work is partly supported by the National Natural Science Foundation of China (Nos. 60973007, 60973008), the Doctoral Fund of Ministry of Education of China (No. 20101102110018), the Fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2009ZX-01), the Fundamental Research Funds for the Central Universities (YWF-10-02-058) and the Scientific Research Fund of Hunan Provincial Education Department (No. 10C0937).