软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(4):867−885 [doi: 10.13328/j.cnki.jos.004756]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种基于服务选取的 SBS 云资源优化分配方法*

赵秀涛, 张 斌, 张长胜

(东北大学 信息科学与工程学院,辽宁 沈阳 110819)

通讯作者: 张斌, E-mail: zhangbin@mail.neu.edu.cn

摘 要: 获取满足全局优化目标的资源分配策略,是影响云环境中基于服务的软件系统(service-based software system,简称 SBS)运行时优化效果的关键.然而,由于 SBS 内部复杂的业务逻辑关系和云环境中的资源约束,现有分配方法无法得到最优资源分配量.以满足 SLA 约束和最小化资源成本为目标,根据不同资源状态对应不同组件服务性能的特点,将组件服务可能的资源分配量、相应性能及成本转换为备选逻辑服务集,进而提出了一种云环境中基于服务选取的 SBS 资源优化分配模型,并设计了一种求解模型的混合遗传算法.算法采用整数编码以提高求解效率,并在选择算子中引入了精英保留策略,从而保证收敛到全局最优解.为提高遗传算法的局部搜索能力、加快收敛速度,以局部搜索策略改进了标准变异算子.实验验证了所提出的资源优化分配模型和求解算法的有效性,并表明:与分支定界法及精英保留策略遗传算法相比,混合遗传算法能够在较大规模的问题上快速获得具有较低资源成本的资源分配策略.

关键词: 云计算;基于服务的软件系统;资源分配;服务选取;遗传算法

中图法分类号: TP316

中文引用格式: 赵秀涛,张斌,张长胜.一种基于服务选取的 SBS 云资源优化分配方法.软件学报,2015,26(4):867−885. http://www.jos.org.cn/1000-9825/4756.htm

英文引用格式: Zhao XT, Zhang B, Zhang CS. Service selection based resource allocation for SBS in cloud environments. Ruan Jian Xue Bao/Journal of Software, 2015,26(4):867−885 (in Chinese). http://www.jos.org.cn/1000-9825/4756.htm

Service Selection Based Resource Allocation for SBS in Cloud Environments

ZHAO Xiu-Tao, ZHANG Bin, ZHANG Chang-Sheng

(School of Information Science & Engineering, Northeastern University, Shenyang 110819, China)

Abstract: Runtime adaptation of service-based software systems (SBS) in cloud environments is a key to acquire resource allocation strategy that meets global optimization goals. However, because of the complex business logic in SBS as well as cloud resource constraints, the optimal resource allocation cannot be obtained using existing methods. This paper offers an approach to meet SLA and minimize resource costs by exploiting the fact that different resource states result in different performance of component services. With the new method, the potential resource allocation for component services, together with responding performance and resource costs, are first transformed into candidate logical service sets. Then a service selection based resource allocation model for SBS in the cloud is constructed. A hybrid genetic algorithm is also designed for solving the model. Integer encoding is applied in the algorithm to improve the efficiency and an elitism maintenance strategy is introduced into selection operator to ensure its convergence to the global optimal solution. In order to improve the local search ability of genetic algorithm and speed up the convergence speed, the standard mutation operator is replaced by local search. Experiments validate the effectiveness of the proposed resource allocation model and its algorithm, and show that the presented algorithm can obtain the resource allocation strategy quickly with lower cost on large-scale problems than the branch and bound method and elitism genetic algorithm.

* 基金项目: 国家自然科学基金(61100090, 61100027); 国家科技支撑计划(2012BAH1305); 中央高校东北大学基本科研专项基金(N110204006, N120804001, N110604002, N120604003)

收稿时间: 2014-07-02; 修改时间: 2014-10-14; 定稿时间: 2014-11-14

Key words: cloud computing; service-based software system; resource allocation; service selection; genetic algorithm

云计算的资源弹性分配特性允许企业和政府等组织按照实际需求购买资源,使其逐渐成为分布式软件系统的主要部署平台[1,2].随着软件规模和复杂性的迅速增长,大部分分布式软件系统开始采用面向服务的体系架构(service-oriented architecture,简称 SOA),进而具备可灵活配置、动态重构、维护难度低等特点.这类基于 SOA 泛型开发的软件系统也称作基于服务的软件系统(service-based software system,简称 SBS)[3].SBS 可表示成一个由一系列组件服务根据不同组合规则(如顺序结构、循环结构、并行结构等)构成的业务流程,其中,各组件服务用于完成特定功能,而组合规则则定义了组件服务之间的交互关系.为了处理用户请求,需要将各组件服务初始化为相应的服务实例,并部署到云环境中分配有一定数量资源的虚拟机上.

由于云资源的按需付费模式,应用提供商在部署 SBS 时,通常希望以最少的资源满足与应用消费者之间达成的服务水平协议(service level agreement,简称 SLA)[4],其中描述了关于服务质量(quality of service,简称 QoS)属性的约束,如响应时间、吞吐量和可靠性等.实际上,云服务提供商的资源并不是无限的,如果应用的资源需求量过大,则当前云资源的状态可能无法满足其需要.因此,面向全局成本优化目标的 SBS 资源分配策略(即,各个组件服务的资源分配量)既要避免 SLA 违例,又要满足当前云环境可用资源状态的约束,分配策略的好坏直接影响 SBS 运行时的环境适应能力[5].

目前,针对 SBS 的云资源分配量计算问题尚未提出有效的分配方法,主要依赖应用提供商的手工分配.对于结构简单的应用,如单层应用或者线性多层应用,可以比较容易地根据运行经验手工确定应用(每层)的最少资源分配量.在手工分配资源方法中,主要通过反复尝试为 SBS 各组件服务分配不同资源量,即,产生不同测试用例,然后从中选取满足 SLA 约束且资源成本最小的资源分配策略.然而在资源众多的云环境中,对于任意组件服务往往存在大量不同资源分配量,进而导致测试用例的组合爆炸问题,此时,手工分配方法是不现实的.

SBS 云资源的手工分配,本质上是通过不断尝试一定数量的测试用例来确定最优资源分配策略的过程,因此可根据基于搜索的软件工程(search-based software engineering,简称 SBSE)[6]思想将其转换为一个最优化问题,并采用元启发式搜索算法求解.分析可知,为 SBS 各组件服务确定最佳资源分配量类似于服务选取问题中为抽象服务选择最优具体服务的过程[7],其中,SBS 对应组合服务流程,组件服务对应抽象服务,其可能的资源分配量对应备选具体服务.由于服务选取是一种面向全局目标求解带约束的组合优化问题的有效手段,因此本文提出将 SBS 云资源的优化分配问题转换成为一个服务选取问题来进行求解.然而,如何将资源划分为组件服务的备选具体服务,是一个必须解决的难点.同时,与基本的服务选取问题不同,求解 SBS 最优资源分配策略时要考虑哪些备选具体服务不能被同时选取,以免违反可用资源状态的约束.

针对上述问题,为了确定使 SBS 整体资源成本最小的资源分配策略,假设资源可以细粒度分配[8],并且能够获得云环境的当前可用资源状态.本文根据不同资源状态对应不同组件服务性能的特点,首先通过资源划分方法获取当前可用资源状态下组件服务可能的资源分配量,并利用性能模型和资源定价模型分别计算相应的组件服务性能与资源成本,进而生成组件服务的备选逻辑服务集;然后,建立了一种基于服务选取的 SBS 云资源优化分配模型,并提出了求解该模型的混合遗传算法.算法采用整数编码方式对个体进行十进制编码,同时引入了精英保留策略,从而保证算法的全局收敛性.另外,针对遗传算法局部搜索能力差的不足,提出了基于局部搜索的变异算子.实验结果表明:所提出的基于服务选取的 SBS 云资源优化分配方法能够有效地确定每个组件服务的最优资源分配量,且与常用的求解整数规划的分支定界法和基于精英保留策略的遗传算法相比,本文的混合遗传算法能够在较大规模问题上获得资源成本较低的资源分配策略,且具有更快的收敛速度.另外,实验分析了不同资源划分策略对解的质量和算法求解效率的影响.

本文的主要贡献在于:

(1) 提出了两种能够显著缩小可行解搜索空间从而提高优化问题求解效率的资源划分策略,包括等宽划分策略和 Ent-MDLP 划分策略;

(2) 在此基础上,根据 SBSE 的思想构建了一种云环境中基于服务选取的 SBS 资源优化分配模型;

(3) 设计了一种能够有效求解该优化模型的基于精英保留策略和局部搜索变异的混合遗传算法.

本文第1节介绍相关工作.第2节描述云环境中基于服务选取的SBS资源优化分配过程.第3节给出组件服务备选逻辑服务集的确定方法.第4节提出SBS云资源优化分配模型及其求解算法.第5节对本文提出的优化分配方法和求解算法进行实验分析.最后总结全文并展望下一步工作.

1 相关工作

1.1 基于服务的应用的云资源分配

现有工作主要从工作流和业务过程角度进行云资源分配的研究.

- 文献[8]研究了一种云环境中基于控制论的自主动态资源分配问题,目标是在满足任务执行时间和资源预算约束的基础上最大化自适应应用的 QoS,其中,自适应应用是由多个服务构成的,但是没有涉及到应用的组成结构;
- 文献[9]采用云服务实现科学工作流,并且提出了一种基于粒子群优化的资源调度方法,其考虑了数据传输和存储代价,并以最小化整体成本为目标;
- 文献[10]研究了由多个工作流构成的SBS的资源动态分配方法,其优化目标是适应资源状态和QoS等的变化,从而最大化系统整体吞吐量;
- 文献[11]提出了一种用于视频监控组合服务应用的资源分配方法,该方法将虚拟机资源分配问题映射为一个多维背包问题,并且采用线性规划和最佳适应下降法进行求解,其假设构成应用的每个媒体服务所需虚拟机的资源量是已知的,问题的实质是确定虚拟机在物理机上的优化放置,目标是使所占用的物理机数量最少;
- 文献[12]针对基于服务的应用,将应用性能目标转换成组件级别的目标,进而确定各组件的资源需求,并使用多层应用程序验证了方法有效性;
- 文献[13]针对资源超额分配带来的高成本问题,提出了 ViePEP 弹性过程平台,其将业务过程管理系统与云资源管理系统的功能相结合,能够调度整个过程或者单个任务以满足服务等级目标,同时尽可能地降低资源成本和资源空闲率.

与这些工作不同的是:

- 本文研究的是在当前云资源状态下求解满足 SLA 约束与资源成本优化目标的 SBS 资源分配策略,而以上工作主要针对 QoS 优化[8−10],如系统吞吐量,或者在最小化成本时未考虑 SLA 约束;
- 其次,本文采用了基于服务选取的方法解决云环境中SBS的资源优化分配问题;
- 最后,在确定 SBS 的各组件服务最优资源分配量时,考虑了资源有限情况下组件服务对资源的竞争,以及资源分配策略对云环境中各物理机可用资源状态约束的满足问题.

1.2 基于搜索的软件工程方法

本文本质上是基于SBSE的思想,采用混合遗传算法在SBS各组件服务的资源分配量空间中寻找能够最小化资源成本,且满足 SLA 约束以及云环境可用资源状态约束的组件服务最优资源分配量组合.

SBSE 方法近年来被广泛用于软件设计、测试、需求工程、软件项目管理等领域[6].这些工作面向不同目标,通过定义合适的适应度函数将软件工程问题转换成基于搜索的最优化问题,并采用不同元启发式搜索算法进行求解.遗传算法能够有效求解 NP 问题且实现相对简单,因此被大量运用于基于 SBSE 方法的研究中.文献[14,15]采用遗传算法研究了 QoS 感知的 Web 服务组合与服务选取问题,表明该算法能够在令人满意的时间内找到近似最优解;文献[16]在考虑性能、可靠性以及代价等因素的基础上,提出一种面向基于组件系统的软件优化及部署的遗传算法;文献[17]从云用户的角度,基于排队论和历史平均到达率研究最优化 QoS 属性的服务部署问题,并提出了一种遗传算法 E^3-R 进行求解,该方法能够减少冗余 QoS 目标;文献[18]研究了如何在考虑云环境特性、部署架构等条件下将由服务构成的软件组件迁移到云平台,并采用遗传算法提高了在巨大解空间中搜

索最优解的效率.

在这些工作中,遗传算法均采用标准的变异算子,而本文在求解资源优化分配模型的混合遗传算法中引入局部搜索代替标准变异算子,以增强算法的局部搜索能力,进而克服早熟问题并加快收敛到最优解的速度.

2 基于服务选取的 SBS 云资源优化分配过程

本文研究的 SBS 包括一个组件服务集合 $S=\{S_i|1\leqslant i\leqslant m,i\in N\}$,这些组件服务根据某些组合规则(包括顺序结构、循环结构、并行结构和选择结构)构成系统的业务流程.图 1 给出了一个 SBS 示例,该 SBS 由 7 个组件服务 $S_1,S_2,\ldots,S_7$ 构成,并且包含 4 种组合规则,即:顺序结构(如 S_2,S_3)、循环结构(如 S_7,且循环次数等于 5)、并行结构(如 S_2,S_4),以及选择结构(如 S_5,S_6,且各分支选择概率分别为 0.3 和 0.7).

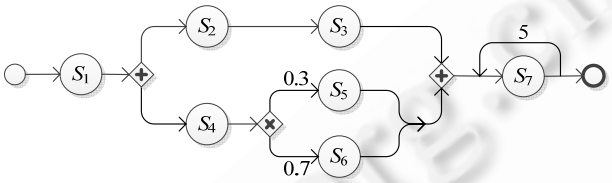


Fig.1 An example of SBS

图 1 SBS 示例

SBS 向多个并发用户提供服务,为保证性能,SBS 具有相应的负载阈值,即,单位时间内的最大并发请求数量.在部署该系统时,要求为各组件服务确定最少的资源分配量,使得它们的性能聚合值在满足 SLA 约束的同时,最小化资源成本.为了避免同一虚拟机上不同组件服务之间可能存在的性能互扰问题,与现有研究相同[11,13],本文假设组件服务与虚拟机的部署关系是 1:1 的,即:一个组件服务只能部署在一台虚拟机上,同时,每个虚拟机上只能部署一个组件服务.可知:对于 SBS,各组件服务的不同资源分配量构成多种组合(即,SBS 的资源分配策略),而不同组合会带来不同的系统性能和资源成本.为了获得资源成本最小的资源分配策略,本文根据不同资源状态对应不同组件服务性能的特点,确定当前可用资源状态下各组件服务所有可能的资源分配量,然后,基于服务选取的方法从中选择最佳资源分配量.

下面举例说明基于服务选取的 SBS 资源优化分配思想.为便于描述,假设当前云环境中存在 3 台物理机:PM1,PM2 和 PM3,各物理机上已经部署了一些其他应用的虚拟机(如 VM1,VM2,…,VM6),剩余的可用资源状态如图 2 所示.

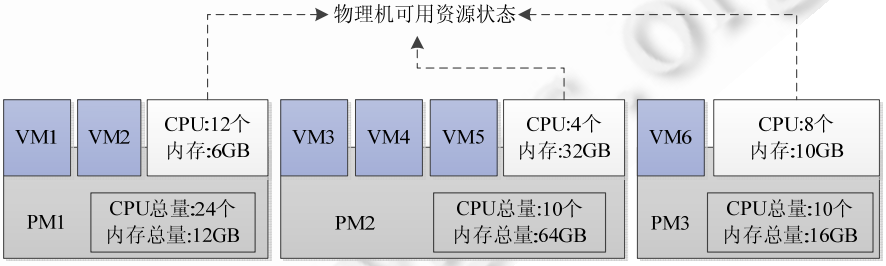


Fig.2 Available resources

图 2 可用资源状态

在将图 1 中的 SBS 部署到该环境时,为了确定各组件服务的最佳资源分配量,使得它们的响应时间聚合值在一定负载阈值下满足端到端响应时间不超过 10s,基于服务选取的 SBS 云资源优化分配过程如下:

(1) 获取组件服务可能的资源分配量:根据某种策略划分当前可用资源(即所有物理机的可分配资源量),从而得到 SBS 中任意组件服务所有可能的资源分配量,即,不同类型资源的分配数量(如 CPU 个数、

内存大小等).这里,通过资源划分,组件服务 S_2 共有 5 种不同的资源分配量,如图 3 所示.

(2) 根据 SBS 的负载阈值和流程结构计算各组件服务的负载阈值,并针对组件服务的每种资源分配量,计算其负载阈值下的响应时间和资源成本.为了尽可能准确地估计资源分配量与组件服务性能之间的关系,通常需要进行资源与性能关系的建模,并根据模型计算任意资源分配量对应的性能(如响应时间、吞吐量等).同时,对于任意资源分配量,根据不同类型资源的定价规则计算其资源成本.此时,可将任意资源分配量及其对应的性能、资源成本视为一个服务,有别于服务计算领域中的服务定义,该服务不仅具有 QoS 属性(即性能和成本),还包括资源属性(即资源分配量),由于其并不具有功能属性,因此,为便于区别和描述,本文称其为逻辑服务.对于任意组件服务,可获得其所有可能的资源分配量、性能及资源成本构成的备选逻辑服务集,如图 4 所示.

(3) 求解组件服务的最优资源分配量.由图 4 可知:SBS 描述(包括组件服务集合、流程结构和 SLA 约束)以及各组件服务的备选逻辑服务集构成了一个以最小化资源成本为目标的服务选取问题.因此,可以根据合适的服务选取算法,基于 SLA 约束为各组件服务选择最优逻辑服务,进而得到其最优资源分配量.

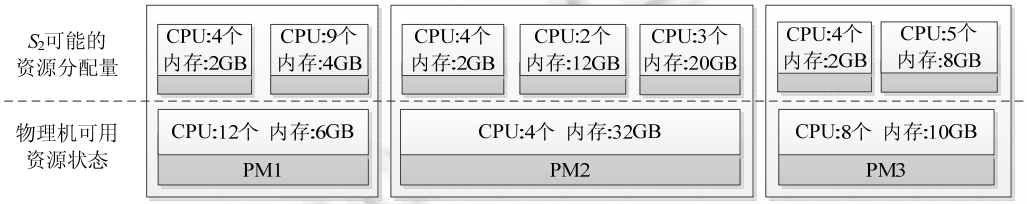


Fig.3 Resource allocation of S_2

图 3 组件服务 S_2 的资源分配量

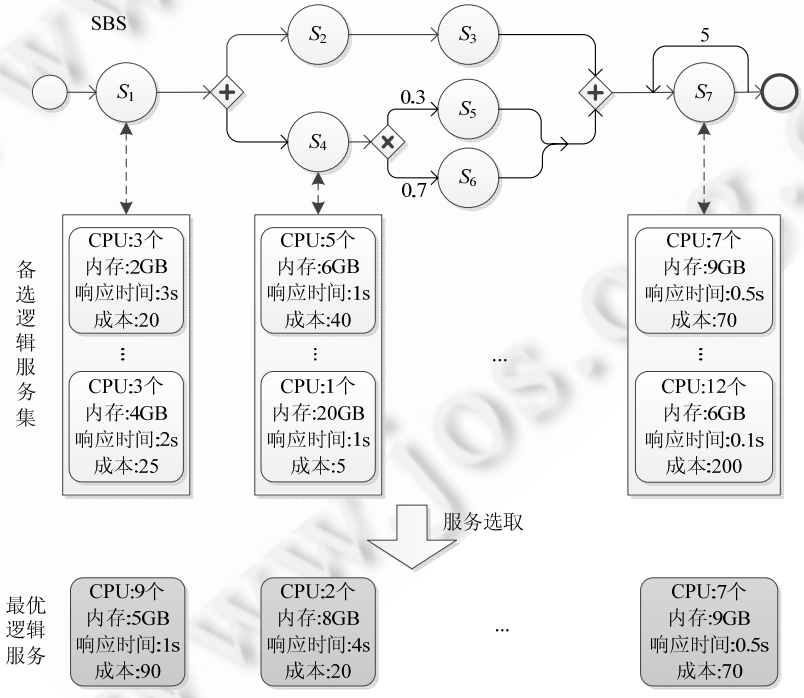


Fig.4 Transformed service selection problem

图 4 转换后的服务选取问题

综上,基于服务选取的 SBS 资源优化分配的具体过程如图 5 所示.

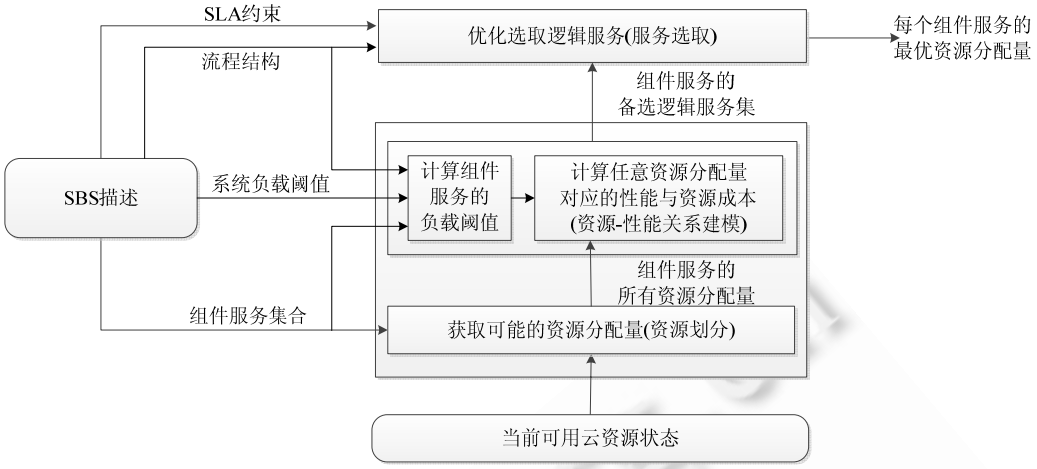


Fig.5 Procedure of service selection based cloud resource allocation optimization for SBS

图 5 基于服务选取的 SBS 云资源优化分配过程

3 确定组件服务的备选逻辑服务集

为了便于描述,本节首先定义逻辑服务.

定义 1(逻辑服务,logical service,简称 LS). LS 是一个描述与某种资源分配量对应的组件服务性能和资源成本的元组,表示为 $LS=\langle R,Q,C\rangle$,其中,$R=(r^1,r^2,\ldots,r^u)$表示资源状态向量,$r^\alpha(1\leqslant\alpha\leqslant u)$为资源$\alpha$的分配数量;$Q=(q^1,q^2,\ldots,q^v)$表示组件服务性能向量,$q^\beta(1\leqslant\beta\leqslant v)$为 QoS 属性$\beta$的值;$C$ 表示 R 中资源数量的总成本.

例如,假设可分配资源类型包括 CPU 个数与内存大小,组件服务性能由响应时间和可靠性描述,如果在某固定负载下,为组件服务 S_1 分配 3 个 CPU 和 10GB 内存时,其响应时间与可靠性分别为 1.5s 和 0.98,且这些资源成本为 500,则此时的逻辑服务 $LS=\langle(3,10),(1.5,0.98),500\rangle$.

实际中,可以采用多种不同的 QoS 属性描述组件服务性能,如响应时间、吞吐量和可靠性等.不同的 QoS 属性受不同因素的影响也不尽相同,不失一般性,本文选择组件服务的响应时间 q 作为其性能指标,此时,根据定义 1,$Q=(q)$.同时,本文仅考虑 4 种常用的可能影响组件服务响应时间的资源,包括 CPU 个数、内存大小、网络带宽和磁盘 I/O,且分别以 r^1,r^2,r^3,r^4 表示它们的分配量.这里,类似 Amazon EC2 计算单元(compute units),对于具有不同架构和处理能力的 CPU,采用统一的度量标准(如 MIPS)将其转化为具有相同计算能力的 CPU[19].

由定义 1 可知:为了获得组件服务的备选逻辑服务集,首先必须根据某种资源划分策略确定其在当前可用资源状态下所有可能的资源分配量;然后,构建任意资源状态与组件服务性能之间的映射关系模型,本文称该模型为组件服务的性能模型,并利用该模型计算任意资源分配量下的组件服务性能.另外,需要根据不同资源的成本计算方式(称作资源定价模型)计算任意资源分配量的资源成本.

3.1 可用资源划分策略

对于资源量很大的云环境,如果资源分配粒度过小,不同类型的资源组合会产生大量资源状态向量,进而导致逻辑服务优化选取的可行解空间过大,影响求解效率;而且,过小的资源分配粒度往往也会对云资源的分配造成不便.另外,如果资源分配粒度过大,则会导致组件服务的资源过剩,进而增加不必要的资源成本.因此,需要将可用资源划分为合适的分配粒度.本文根据资源划分是否依赖组件服务的性能特征,提出了两种资源划分策略.

3.1.1 等宽(equal-width)资源划分

在该策略中,针对任意资源类型,采用相同宽度的划分,并且以划分点为可用资源分配量.如图 6 所示,对于 CPU 资源,假设以宽度 2 进行划分,则对于给定拥有 9 个可用 CPU 的物理机,得到可分配资源量为 2,4,6,8.

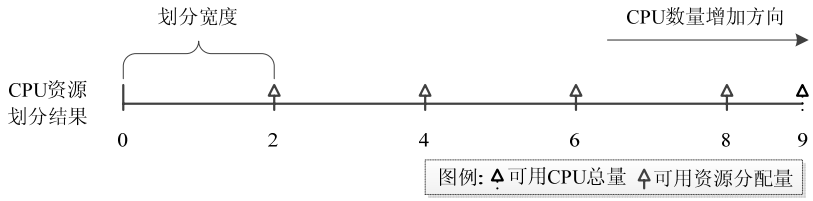


Fig.6 An example of equal-width resource partition

图 6 等宽资源划分示例

3.1.2 熵-最小描述长度原则(Ent-MDLP)[20]资源划分

首先,采用 *K*-均值聚类算法将组件服务在不同资源状态下的响应时间进行聚类,以簇边界作为不同区间的分割点,从而使得每个资源状态与一个响应时间的类相对应.然后,基于类信息,采用 Ent-MDLP 算法将各类型资源属性划分为不同区间.

如图 7 所示,根据 Ent-MDLP 算法将 CPU 资源离散化为 4 个区间:[0,3.5),[3.5,5.5),[5.5,8.5),[8.5,+∞).由于资源属性离散化得到的结果是区间,故无法直接据此划分资源.为此,本文采用求区间平均值的方法,以平均值作为可用的分配资源量.由于最后一个区间是上界为+∞的开区间,无法求平均值,为此,本文以待划分的资源总量作为区间上界,并求平均值,以实现对资源的充分利用,因此得到 1.75,4.5,7 和 8.75.由于 CPU 的分配单位是整数,因此,对平均值四舍五入取整,得到可用资源分配量:2,5,7,9.

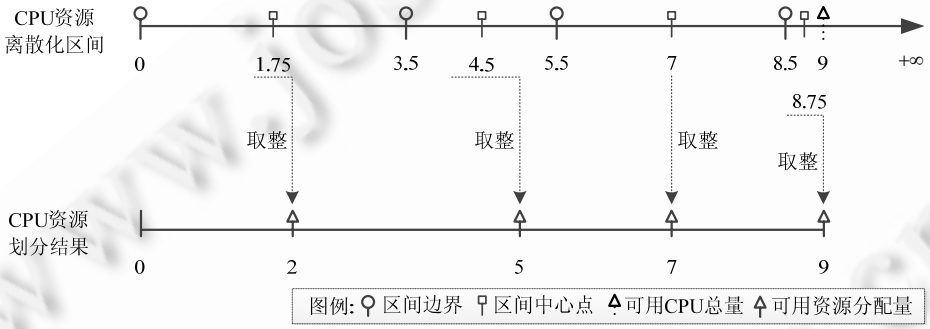


Fig.7 An example of Ent-MDLP based resource partition

图 7 Ent-MDLP 资源划分示例

3.2 构建组件服务的性能模型

根据现有研究[21]可知,负载、资源类型及其数量是影响组件服务响应时间的主要因素,其中,负载包括并发请求数量以及每个请求的事务规模.为简单起见,本文假设每个对组件服务的请求均具有相同的事务规模,因此可将负载定义为单位时间内所有用户对组件服务的并发请求总数.基于此,给出本文组件服务性能模型的定义.

定义 2(组件服务性能模型,component service performance model,简称 CSPM). CSPM 是一个描述在一定负载 L 下,资源状态向量 R 与组件服务响应时间 q 之间映射关系的模型,表示为 $\Phi:\langle L,R\rangle\rightarrow q$,其中,$\Phi$表示 L 和 R 到 q 的映射函数.

为了构建组件服务 $S_i(1\leqslant i\leqslant m)$的性能模型,通常需要对其进行性能基准测试以获得一定数量的执行日志 $D_i=\{(l,r^1,r^2,r^3,r^4,q)\}$,其中包含 S_i 在不同负载 l 和不同资源状态(r^1,r^2,r^3,r^4)下的响应时间 q.获得组件服务的执行日志后,将其作为建模的训练集,进而建立性能模型.根据建模方法的不同,映射函数Φ可以有不同形式,如回归函数、决策树或人工神经网络等[22].由于应用性能与资源分配量之间通常不存在线性关系[23],线性回归方法难以适用,因此本文采用支持向量回归(support vector regression,简称 SVR)[24]来构建组件服务的性能模型Φ.

3.3 资源定价模型

在实际应用中,可以根据云服务提供商的业务需要采用不同的定价模型计算资源成本,如线性定价模型、指数定价模型等.为便于描述,本文对所有类型的资源均采用线性定价模型,即:假设资源成本与资源数量、资源使用时长呈正比,且不考虑资源之间的交叉定价问题,则资源$\alpha(1\leqslant\alpha\leqslant4)$的成本:

$$c^{\alpha}=c^{\alpha}_{unit}\times c^{\alpha}_{amount}\times t_{duration} \tag{1}$$

其中,c^{α}_{unit}表示α的单位资源在单位时间内的价格,c^{α}_{amount}表示α的分配总量,$t_{duration}$表示c^{α}_{amount}的使用时长.本文中,CPU、内存、网络带宽和磁盘 I/O 的资源单位分别为:个、GB、Mbps 和 MB/s.

3.4 备选逻辑服务集生成算法

对于任意组件服务,为了获取其备选逻辑服务集合,首先采用文献[25]中的标记树方法,根据 SBS 的系统负载阈值和流程结构计算各组件服务的负载阈值,然后确定组件服务的资源分配粒度.在此基础上遍历所有物理机,根据某种资源划分策略得到所有可能的资源状态向量,并根据组件服务的性能模型和资源定价模型计算每个资源状态向量对应的响应时间和资源成本,从而获得备选逻辑服务集.

给定一个由 m 个组件服务 $S_1,S_2,\ldots,S_m$ 构成的 SBS,以及包含 n 个物理机 $pm_1,pm_2,\ldots,pm_n$ 的云环境,各物理机上的可用资源量为 R^{α}_k $(1\leqslant\alpha\leqslant4,1\leqslant k\leqslant n)$,则组件服务的备选逻辑服务集生成算法(candidate logical service set generation algorithm,简称 CLSG)描述如下:

算法 1. 备选逻辑服务集生成算法(CLSG).

输入:组件服务集合 $S=\{S_i\}$,可用资源量 R^{α}_k,系统负载阈值 L_{sbs};

输出:任意组件服务 S_i 的备选逻辑服务集 Z_i.

过程:

1. $Z_i\leftarrow NULL$; /*初始化备选逻辑服务集为空*/
2. $L_i\leftarrow ComputeWorkload(L_{sbs})$; /*根据 L_{sbs} 计算 S_i 的负载阈值*/
3. **foreach** S_i in S **do**
4. **if** $PartitionStratergy(S_i)$='Ent-MDLP' **then** /*基于 Ent-MDLP 划分资源属性*/
5. $Classifier\leftarrow GetCluster$('$K$-Means',$D_i,q$); /*对响应时间进行 K-均值聚类*/
6. **foreach** α **do**
7. $g^{\alpha}_i\leftarrow GetResourceInterval$('Ent-MDLP',$Classifier,D_i,\alpha$);
8. **end for**
9. **end if**
10. **foreach** pm_k **do**
11. /*确定资源划分点 $PartitionStratergy$:'Equal-Width'或'Ent-MDLP'*/
12. **foreach** α **do**
13. $\rho^{\alpha}\leftarrow GeneratePartitionPoint$('$PartitionStratergy$', $R^{\alpha}_k,g^{\alpha}_i$);
14. **end for**
15. /*计算不同资源的笛卡尔积,得到所有可能的资源状态向量*/
16. $\{(r^1,r^2,\ldots,r^4)\}\leftarrow ComputeCartesianProduct(\rho^1,\rho^2,\ldots,\rho^4)$;
17. /*根据 S_i 的性能模型计算不同资源状态向量下的响应时间*/
18. $\{q\}\leftarrow ComputeResponseTime(L_i,\{(r^1,r^2,\ldots,r^4)\})$;
19. $\{c\}\leftarrow ComputeResourceCost(\{(r^1,r^2,\ldots,r^4)\})$; /*计算资源成本*/
20. $Z_i\leftarrow Z_i\cup\{\langle(r^1,r^2,\ldots,r^4),q,c\rangle\}$; /*获得 S_i 的备选逻辑服务集*/
21. **end for**
22. **end for**

23. **return** $\{Z_1,Z_2,\ldots,Z_m\}$;

在资源划分策略1中,由于组件服务 S_i 在物理机 pm_k 上的所有可能的资源状态向量数目为 $\prod_{\alpha=1}^{4}\left\lfloor R_k^\alpha / g^\alpha \right\rfloor$,因此,算法1的时间复杂度为 $O\left(mn\prod_{\alpha=1}^{4}\left\lfloor \max_{1\leqslant k\leqslant n}(R_k^\alpha)/g^\alpha \right\rfloor\right)$,其中,$\max_{1\leqslant k\leqslant n}(R_k^\alpha)$ 为资源α的最大可用数量.可知,该算法的时间复杂度较高.实际中,可以通过调整资源划分宽度减少可能的资源状态向量,从而提高算法效率.然而,这样却有可能增加资源成本.

在资源划分策略2中,K-均值聚类算法的时间复杂度和Ent-MDLP有监督离散化算法的时间复杂度[26]分别为 $O(K\times|D_i|\times T)$和 $O(4\times|D_i|\times\log|D_i|)$,其中,$K$ 为区间划分个数,T 为聚类算法的迭代次数,$|D_i|$为组件服务执行日志的大小.

4 基于服务选取的SBS云资源优化分配模型及求解

SBS云资源优化分配问题的约束之一是满足用户与应用提供商之间的SLA,而SLA通常包括不同的QoS属性.本节首先给出SBS的QoS模型以及QoS属性的计算方法,然后形式化资源优化分配问题,并建立相应的优化模型以及提出基于混合遗传算法的求解算法.

4.1 SBS的QoS模型

为便于描述基于服务选取的SBS云资源优化分配模型,不失一般性,本文仅考虑一个QoS属性,即SBS的端到端响应时间,而其他QoS属性(如吞吐量、可靠性、可用性、信誉度等)均可以很容易地作为该优化模型的约束条件.这里需要注意的是:在增加其他QoS属性约束时,需要相应地改变逻辑服务中的组件服务性能向量.

- 端到端响应时间$\mathcal{Q}$:表示用户从调用SBS到收到其返回结果所消耗的时间.

本文定义SBS的SLA约束为:当单位时间内用户对SBS的并发请求数不超过系统负载阈值 L_{sbs} 时,保证$\mathcal{Q}$低于最大端到端响应时间$\mathcal{Q}_{\max}$.

为了判断某种资源分配策略下的SBS性能是否满足SLA,采用标记树对SBS的组合流程进行等价转换并计算端到端响应时间[25].因此,SBS 可表示为标记树 $T=(V,E,\mathcal{B})$,其中,$V,E,\mathcal{B}$分别为树节点、边和边标记的集合.任意非根节点 $v\in V$ 的响应时间计算规则见表1.表1中,$d(v)$表示 v 的直接后继节点,$\lambda(v,u)$表示 v 内 u 的执行次数,$\mathcal{Q}(v)$表示 v 的响应时间.

Table 1 Computing rules for response time

表1 响应时间的计算规则

节点 $v\in V$	计算表达式
顺序结构	$\mathcal{Q}(v)=\sum_{u\in d(v)}\mathcal{Q}(u)$
循环结构	$\mathcal{Q}(v)=\lambda(v,d(v))\mathcal{Q}(d(v))$
分支结构	$\mathcal{Q}(v)=\sum_{u\in d(v)}\lambda(v,u)\mathcal{Q}(u)$
并行结构	$\mathcal{Q}(v)=\max_{u\in d(v)}\mathcal{Q}(u)$
组件服务	$\mathcal{Q}(v)=\mathcal{Q}(S_i)$

根据标记树的递归计算规则,可得SBS端到端的响应时间$\mathcal{Q}=\mathcal{Q}(root)$,其中,$root$ 表示 T 的根节点.

另外,由标记树的结构可知:对于每次请求,任意组件服务 S_i 的期望执行次数 $V_i=\prod_{j\pm S_i}\lambda(f(j),j)$. 其中,$f(j)$表示 j 的父节点.因此,若SBS系统负载阈值为 L_{sbs},则 S_i 的负载阈值 $L_i=L_{sbs}V_i$.

4.2 基于服务选取的SBS云资源优化分配模型

给定一个由 m 个组件服务 $S_1,S_2,\ldots,S_m$ 构成的SBS以及包含 n 个物理机 $pm_1,pm_2,\ldots,pm_n$ 的云环境,其中,各物理机上的可分配资源量为 R_k^α $(1\leqslant\alpha\leqslant4,1\leqslant k\leqslant n)$.根据CLSG算法已知,各组件服务的备选逻辑服务集 $Z(S_i)=$

$\{C_{i,j}|1\leqslant j\leqslant n_i\}$,其中,$C_{i,j}=\langle r_{i,j},q_{i,j},c_{i,j}\rangle$,资源状态向量 $r_{i,j}=(r_{i,j}^1,r_{i,j}^2,r_{i,j}^3,r_{i,j}^4)$,资源成本 $c_{i,j}=\sum_{\alpha=1}^{4}c_{i,j}^{\alpha}$, n_i 表示 S_i 的备选逻辑服务数量.根据资源定价模型可知,云资源是按使用时长收费的.因此,SBS 云资源优化分配问题是指如何为各组件服务选择一个合适的备选逻辑服务,使得单位时间内的总体资源成本最小,同时满足端到端响应时间约束.另外,与基本的服务选取问题不同,在该基于服务选取的 SBS 云资源分配模型中,还要求最优备选逻辑服务组合 $(\langle S_1,C_{1,j_1}\rangle,\langle S_2,C_{2,j_2}\rangle,...,\langle S_m,C_{m,j_m}\rangle)$ 中各组件服务的资源分配量满足当前云环境的可用资源约束,即:至少存在一种 SBS 的部署方式,使得云环境中各物理机上为组件服务分配的资源量之和不超过其可分配资源量 R_k^{α}.

根据上述问题定义可知,资源优化分配模型中包含两类决策变量,分别以 $x_{i,j}$,$y_{i,j,k}$ 表示.

- $x_{i,j}$=1 表示组件服务 S_i 选择逻辑服务 $C_{i,j}$,否则 $x_{i,j}$=0;
- $y_{i,j,k}$=1 表示根据 $C_{i,j}$ 创建的虚拟机部署在物理机 pm_k 上,否则 $y_{i,j,k}$=0.

令 $\boldsymbol{X}=(x_{i,j})$,$\boldsymbol{Y}=(y_{i,j,k})$,则资源分配的优化模型为

$$\text{Min } F(\boldsymbol{X})=\sum_{i=1}^{m}\sum_{j=1}^{n_i}c_{i,j}x_{i,j} \tag{2}$$

$$\text{s.t.} \quad Q(\boldsymbol{X})\leqslant Q_{\max} \tag{3}$$

$$\sum_{j=1}^{n_i}x_{i,j}=1,i=1,2,...,m \tag{4}$$

$$\sum_{j=1}^{n_i}\sum_{k=1}^{n}y_{i,j,k}=1,i=1,2,...,m \tag{5}$$

$$y_{i,j,k}\leqslant x_{i,j},i=1,2,\ldots,m,j=1,2,\ldots,n_i,k=1,2,\ldots,n \tag{6}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i}r_{i,j}^{\alpha}y_{i,j,k}\leqslant R_k^{\alpha},k=1,2,...,n,\alpha=1,2,3,4 \tag{7}$$

$$y_{i,j,k},x_{i,j}\in\{0,1\},i=1,2,\ldots,m,j=1,2,\ldots,n_i,k=1,2,\ldots,n \tag{8}$$

公式(2)表示 $\boldsymbol{X}$ 下 SBS 各组件服务资源分配量的总成本;公式(3)为 $\boldsymbol{X}$ 下 SBS 的端到端响应时间约束;公式(4)表示任意组件服务只能选择一个逻辑服务,也就是说,只能为一个组件服务创建一个虚拟机;公式(5)、公式(6)表示根据逻辑服务创建的虚拟机只能部署在云环境中的一个物理机上;公式(7)表示在物理机 pm_k 上为组件服务分配的资源总量不能超过其最大可用资源数量.

在上述模型中,本文为了便于描述所提出的资源优化分配方法,仅考虑了单个目标,即,部署 SBS 的资源成本.实际上,可以根据云服务提供商的需要在该模型中增加其他目标,如最小化组件服务占用的物理机总数,从而可以关闭闲置物理机以实现节能,或者在最小化资源成本的同时,尽可能地提高 SBS 的性能.此时,该模型描述为一个多目标优化问题,可采用某些多目标优化算法求解,本文限于篇幅,不再赘述.

求解上述资源优化分配模型得到 SBS 的最优逻辑服务组合后,即可根据各逻辑服务中的资源状态向量为相应组件服务分配资源.由于服务选取问题等价于多维多选择背包问题,因此是 NP 困难的[27],无法在多项式时间内求得精确解,故本文采用遗传算法求解该优化模型的近似最优解.

4.3 混合遗传算法求解优化模型

遗传算法(genetic algorithm,简称 GA)是一种通过群体迭代求解优化问题的元启发式算法,能够有效地求解 NP 问题[28].为了获得优化问题的一个近似最优解,首先要为 GA 选取合适的个体编码方式.本文采用整数编码形式,个体的每个基因位取值为十进制整数,与二进制编码相比,该编码方式具有编码长度短、求解效率高等特点.为了使 GA 能够收敛到全局最优解,在标准遗传算法(canonical genetic algorithm,简称 CGA)的选择算子中引入了精英保留策略[29],从而避免最优个体在杂交操作中被破坏.同时,采用局部搜索代替标准的变异算子以增强 GA 的局部搜索能力,克服早熟问题并加快收敛到最优解的速度.

4.3.1 个体编码

为了在选择逻辑服务时判断其能否满足当前可用云资源状态约束,令个体长度等于 $2m$,并采用十进制整数进行编码.基因位 $g_i(1\leqslant i\leqslant m)$的取值是 S_i 的备选逻辑服务索引,其范围为$[1,n_i]$,其中,n_i 为 S_i 的备选逻辑服务个数;基因位 $g_j(m+1\leqslant j\leqslant 2m)$的取值是云环境中的物理机索引,其范围为$[1,n]$,其中,$n$ 为物理机个数.可知,该编码方式使得任意个体都能保证组件服务与虚拟机的部署关系为 1:1.图 8 给出了个体编码原理.

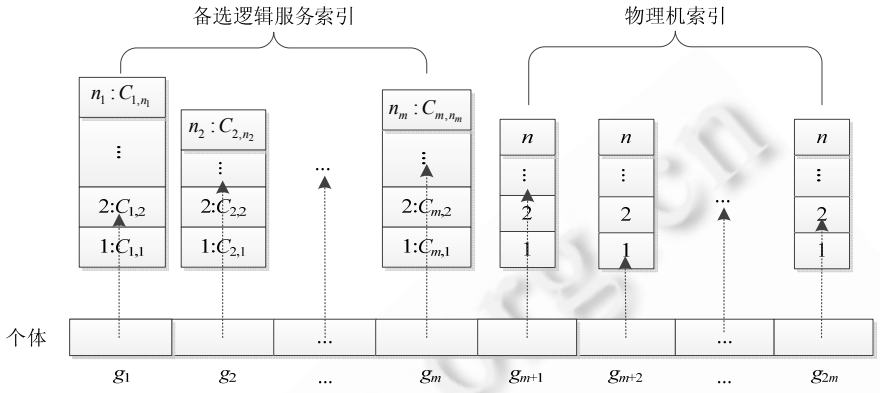


Fig.8 Individual encoding

图 8 个体编码

4.3.2 遗传算子

选择(selection)算子采取精英保留策略,即,将当前种群中适应度值最大的个体(称作精英个体 elitist)直接复制到下一代,从而保证GA的全局收敛性.同时,采用随机遍历抽样法(stochastic universal sampling)选择进行交叉变异的个体.该方法是经典轮盘赌选择法(roulette wheel)的一种改进,其特点是只需一次轮盘旋转,能够在一定程度上防止早期的高适应度个体迅速占据种群,从而避免收敛到局部最优解.

交叉(crossover)算子采用标准的两点交叉,即:对于任意个体 g_1,根据一定的交叉概率从其他被选择出的个体中随机选择一个配对个体 g_2,随机选择两个不同的交叉点 $o_1,o_2(1\leqslant o_1<o_2\leqslant 2m-1)$,以 g_2 中 o_1 和 o_2 之间的基因片段替换 g_1 中相应位置的片段,从而形成新的个体.

为了提高 GA 的局部搜索能力,类似文献[30],采用基于局部搜索的变异(mutation)算子,即:对新的种群中的任意适应度值大于或等于种群平均适应度值的个体 g,从其邻域 $N(g)$中随机选择 l 个候选个体加入该种群.本文采用海明距离等于 1 来定义领域,即:对于一个个体,其领域是指所有与其具有 1 个不同基因位的个体构成的集合.此时,为了维持种群规模不变,假设群体规模为 NP,将精英个体加入该种群,并根据适应度值从高到低对个体进行排序,选取前 NP 个个体作为下一代种群.

4.3.3 适应度函数

为了评价个体 g 的质量,考虑到其必须满足端到端响应时间约束和物理机资源约束,因此在适应度函数中引入罚函数,以降低在解空间中无对应可行解的个体的适应度,减少其遗传到下一代群体的概率,进而加快算法收敛速度.这里,采用界限构造法定义保证取值大于 0 的适应度函数:

$$Fit(g)=(4n+1)+F'(\boldsymbol{X})-\delta_1\mathcal{Q}(\boldsymbol{X})-\delta_2\sum_{\alpha=1}^{4}\sum_{k=1}^{n}R_{\alpha,k}(\boldsymbol{Y}) \tag{9}$$

其中,$F'(\boldsymbol{X})$为与 $\boldsymbol{X}$ 对应的资源成本归一化后的值,即:

$$F'(\boldsymbol{X})=(F_{\max}-F(\boldsymbol{X}))/(F_{\max}-F_{\min}) \tag{10}$$

其中,$F_{\max}=\sum_{i=1}^{m}\max_{1\leqslant j\leqslant n_i}\{c_{i,j}\},F_{\min}=\sum_{i=1}^{m}\min_{1\leqslant j\leqslant n_i}\{c_{i,j}\}$.而$\mathcal{Q}(\boldsymbol{X}),R(\boldsymbol{Y})$分别表示 $\boldsymbol{X},\boldsymbol{Y}$ 违反优化模型的约束(3)和约束(7)时的惩罚函数,有:

$$\mathcal{Q}(\boldsymbol{X})=\begin{cases}(\mathcal{Q}(\boldsymbol{X})-\mathcal{Q}_{\max})/(\mathcal{Q}'_{\max}-\mathcal{Q}_{\max}), & \mathcal{Q}(\boldsymbol{X})>\mathcal{Q}_{\max}\\ 0, & \mathcal{Q}(\boldsymbol{X})\leqslant\mathcal{Q}_{\max}\end{cases} \tag{11}$$

其中, $\mathcal{Q}'_{\max}$ 表示由各组件服务的最大响应时间备选逻辑服务构成的 SBS 的端到端响应时间.

$$R_{\alpha,k}(\boldsymbol{Y})=\begin{cases}(\Omega(\boldsymbol{Y},\alpha,k)-R_k^{\alpha})/(\Omega_{\max}^{\alpha,k}-R_k^{\alpha}), & \Omega(\boldsymbol{Y},\alpha,k)>R_k^{\alpha}\\ 0, & \Omega(\boldsymbol{Y},\alpha,k)\leqslant R_k^{\alpha}\end{cases} \tag{12}$$

其中, $\Omega(\boldsymbol{Y},\alpha,k)=\sum_{i=1}^{m}\sum_{j=1}^{n_i}r_{i,j}^{\alpha}y_{i,j,k}$, 且 $\Omega_{\max}^{\alpha,k}=\sum_{i=1}^{m}\max_{1\leqslant j\leqslant n_i}\{r_{i,j}^{\alpha}\}$.另外,$\delta_1,\delta_2(\delta_1,\delta_2\in(0,1))$为惩罚因子,分别表示违反端到端响应时间约束和物理机资源约束的惩罚程度.

4.3.4 算法终止条件

本文采用 De Jong 提出的离线性能函数(off-line performance)评估准则来衡量算法的收敛性,并将其作为遗传迭代的终止条件之一.以 $X(T)$表示第 T 次迭代的离线性能,则对于给定阈值ε,当 $X(T)$–$X(T$–1)$\leqslant\varepsilon$时,终止搜索.同时,为了避免搜索时间过长,设定一个最大遗传代数 NG 为终止条件.当算法满足两者中的任意条件时,则终止搜索.

该算法结合了遗传算法和局部搜索特点,因此是一种混合遗传算法(hybrid GA,简称 HGA).具体算法描述如下.

算法 2. 面向逻辑服务选取的混合遗传算法(HGA).

输入:备选逻辑服务集$\{Z_i\}$,最大端到端响应时间$\mathcal{Q}_{\max}$,可用资源 R_k^{α} ,种群规模 NP,

交叉概率 P_c,局部搜索的候选个体数量 $NumOfCandidates$;

输出:任意组件服务 S_i 的最优备选逻辑服务 LS_i^* :$\{\langle S_i,LS_i^*\rangle\}$.

过程:

1. $X(0)\leftarrow-\infty$; /*置初始离线性能函数值为无穷小*/
2. $Pop\leftarrow Initialize$('*IntegerCoding*',NP); /*按照整数编码随机产生初始种群*/
3. **while** $T\leqslant NG$ **do**
4. **foreach** g in Pop **do**
5. $f^g\leftarrow Fit(g)$; /*计算任意个体 g 的适应度值*/
6. **end for**
7. $g_{best}\leftarrow\text{argmax}_{g\in Pop}(f^g)$; /*保留当前种群中适应度值最高的个体*/
8. $SelPop\leftarrow Select$('SUS',Pop); /*采用随机遍历抽样(SUS)选择个体*/
9. $SelPop\leftarrow Crossover$('2-Point',$SelPop$,$P_c$); /*两点交叉*/
10. /*基于局部搜索的变异*/
11. $SelPop\leftarrow LocalSearchMutation(SelPop,NumOfCandidates)$;
12. /*将保留的精英个体加入变异种群,并按适应度值从高到底排序(descend)*/
13. $SelPop\leftarrow FitnessRanking$('Descend',$SelPop\cup\{g_{best}\}$);
14. $Pop\leftarrow GetPopulation(SelPop,NP)$; /*选择前 NP 个个体作为新一代种群*/
15. $X(T)\leftarrow ComputeOffline(Pop)$; /*计算离线性能*/
16. **if** $X(T)$–$X(T$–1)$\leqslant\varepsilon$ **then**
17. **break**; /*满足终止条件,跳出循环*/
18. **end if**
19. $T\leftarrow T$+1;
20. **end while**
21. $\{\langle S_i,LS_i^*\rangle\}\leftarrow Decoding(\text{argmax}_{g\in Pop}(f^g))$; /*最优解解码*/

22. **return** $\{\langle S_i, LS_i^* \rangle\}$;

/*子函数:基于局部搜索的变异,返回变异后的种群*/

function *LocalSearchMutation*(*Current_Pop*,*NumOfCandidates*) **returns** *MutatedPop*

1. *MutatedPop*←*Current_Pop*;
2. $\overline{f} \leftarrow MeanFitness(Current\_Pop)$; /*计算当前种群的平均适应度值*/
3. **foreach** *g* in *Current_Pop* **do**
4. **if** $f^g \geqslant \overline{f}$ **then** /*从个体领域 *Neighbor*(*g*)中随机选择候选个体*/
5. *Cand*(*g*)←*RandomSelect*(*Neighbor*(*g*),*NumOfCandidates*);
6. *MutatedPop*←*MutatedPop*∪*Cand*(*g*); /*将候选个体加入变异种群*/
7. **end if**
8. **end for**
9. **return** *MutatedPop*;
10. **end function**

5 实验分析

本节通过仿真实验,分析验证所提出的基于服务选取的 SBS 资源优化分配模型及其求解算法在确定资源成本最小的组件服务最优资源分配量方面的可用性和有效性.实验由两部分构成:

(1) 实验环境的相关配置;

(2) 实验结果与分析,包括:(i) 不同惩罚因子和候选个体数量下的HGA算法收敛性分析;(ii) 不同资源划分策略对CLSG算法和HGA算法求解效率以及解的质量的影响;(iii) 本文的HGA算法与其他算法在解的质量和求解效率方面的对比.

5.1 实验环境配置

实验以1台HP Z820工作站实现:(1) 组件服务的资源与性能关系基准测试;(2) 模拟SBS部署的云资源环境;(3) 验证算法性能.该工作站的具体配置:Intel Xeon CPU E5-2620 v2@2.10GHz(2 处理器),64GB 内存,1 000Mbps带宽,200MB/s磁盘I/O,Windows 7 Professional操作系统,且采用Matlab 2013a实现和测试算法.同时,利用1台PC部署组件服务资源与性能关系基准测试的请求发生器Httperf.

以图1的例子模拟云环境中的SBS,并根据资源依赖类型将其组件服务实现为4类Web服务:CPU密集型(S_1,S_2)、通信密集型(S_3)、I/O密集型(S_4,S_5)和其他型(S_6,S_7),从而形成多样性资源需求,更好地仿真复杂的云应用.

为了降低组件服务性能模型建模的工作量,并且不影响实验结果,本文设置SBS的系统负载阈值L_{sbs}=6000,基于此,计算得到各组件服务的负载阈值.在获取组件服务执行日志时,首先创建1个KVM虚拟机,并为其分配不同数量的资源组合(CPU个数、内存大小、网络带宽、磁盘I/O);然后,以等于负载阈值的并发请求率持续30s调用该组件服务,并监测每次请求的响应时间;最后统计所有请求的平均响应时间,即可获得组件服务在该负载阈值和资源状态下的1条执行日志.实验中,对于S_1~S_7,每个组件服务平均获得150条执行日志.

基于获取的组件服务执行日志,采用 SVR 算法建立资源与组件服务响应时间的关系,算法主要参数设定:Puk(Pearson VII function-based universal kernel)核函数[31],其中,ω=1.0,σ=1.0.

为了验证SBS的资源优化分配模型及HGA算法的可用性和有效性,实验以虚拟机模拟了20个云环境中的物理机,各物理机的资源总量取值范围及不同类型资源的单位价格见表2.另外,设置SLA约束的最大端到端响应时间$Q_{\max}$=90s.

HGA算法的主要参数包括:种群大小*NP*=50,交叉概率P_c=0.7,离线性能终止阈值ε=1.0×10^{-4},最大终止代数*NG*=100,而违反约束的惩罚因子取值以及局部搜索变异算子的候选个体数量通过实验确定.另外,由于算法搜索解时的随机性,每次实验均重复执行20次,并将最大适应度值的平均值对应的个体作为最优解.

Table 2 Ranges of resource cost and available physical resources

表 2 单位资源价格与物理机可用资源范围

	CPU(个)	内存大小(MB)	网络带宽(Mbps)	磁盘 I/O(MB/s)
单位资源价格	10	0.02	2.5	3
资源取值范围	1~12	500~16×1024	2~80	50~200

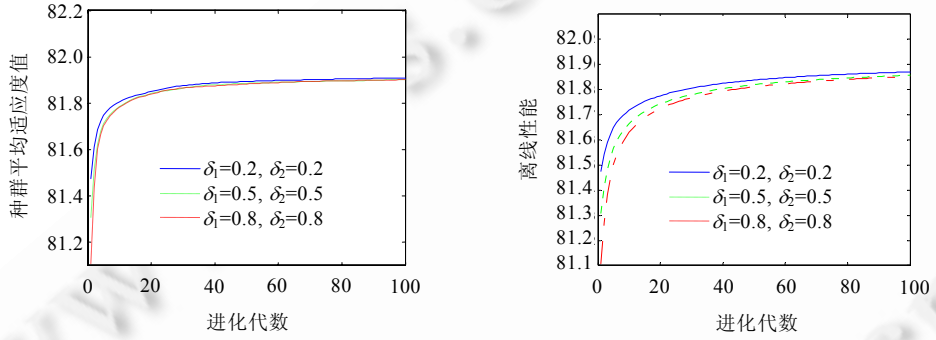
5.2 实验结果与分析

5.2.1 HGA 算法收敛性

算法的收敛性对于是否能够在合理时间内找到近似最优解至关重要,直接影响算法的可用性,本组实验考察了影响 HGA 算法收敛速度的主要参数(包括惩罚因子和局部搜索变异算子的邻域候选个体数量)在不同取值时的收敛曲线.为了比较全面地评价算法收敛性,实验采用种群平均适应度值和离线性能两个指标.

(1) 惩罚因子的影响

实验中,违反端到端响应时间约束和物理机资源约束的惩罚因子分别取 3 组不同的值:(0.2,0.2),(0.5,0.5)和(0.8,0.8),且局部搜索变异的候选个体数量 *NumOfCandidates*=20,收敛曲线如图 9 所示.根据图 9(a)、图 9(b)可知:HGA 算法平均迭代 70 次即可收敛到最优解,且惩罚因子越小,收敛速度越快.但实验中发现,此时更容易违反约束.因此在下面的实验中,惩罚因子的组合采用(0.5,0.5),从而兼顾算法收敛速度和对约束条件的满足.



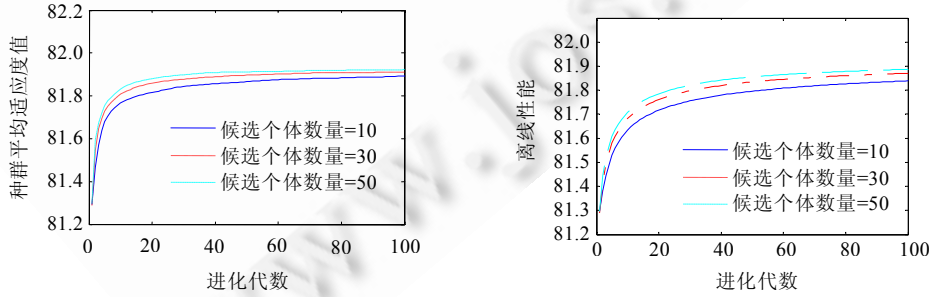
(a) 不同惩罚因子的种群平均适应度值变化 (b) 不同惩罚因子的离线性能变化

Fig.9 Algorithm convergence for different penalty factors

图 9 不同惩罚因子的算法收敛性

(2) 候选个体数量的影响

实验中,候选个体数量 *NumOfCandidates* 分别取值为 10,30,50,结果如图 10 所示.



(a) 不同候选个体数量的种群平均适应度值变化 (b) 不同候选个体数量的离线性能变化

Fig.10 Algorithm convergence for different numbers of candidate individuals

图 10 不同候选个体数量的算法收敛性

由图10可知:与图9的结果类似,HGA算法在最多迭代70次时近似收敛到最优解,而且局部搜索变异时的候选个体越多,算法的收敛速度越快,更容易以较少的进化代数获得近似最优解.但是,当候选个体数量过大时,HGA 算法容易演变为局部搜索算法,进而影响其全局搜索能力.因此,为了同时保证算法收敛性和全局搜索能力,在下面的实验中折中设置 *NumOfCandidates*=30.

由本组实验可知:HGA 算法具有良好的收敛性,且不同参数取值对于算法收敛性具有比较显著的影响.

5.2.2 不同资源划分策略的影响

本组实验以内存资源的划分为例,对本文提出的两种资源划分策略在备选逻辑服务集生成算法(CLSG)和HGA 算法的求解效率以及最小资源成本方面的影响进行评价.对于等宽资源划分策略,设置其他资源的划分宽度分别为 CPU:1,带宽:10,磁盘 I/O:30,实验结果见表 3.

Table 3 Effect of equal-width resource partition

表 3 等宽资源划分策略的影响

内存大小属性划分宽度	平均备选逻辑服务数(个)	CLSG 算法执行时间(s)	HGA 算法执行时间(s)	最小资源成本
512	1413	54.623 9	2.671	1 332.16
768	914	36.256 8	2.193 6	1 373.6
1 024	684	29.087 2	2.013 8	1 393.84
1 280	531	23.967 9	1.814	1 399.2
1 536	430	20.188 0	1.671 9	1 420.76
1 792	364	19.694 0	1.554 3	1 445.88
2 048	330	16.306 7	1.407 4	1 481.72
2 304	265	13.782 6	1.153 4	1 517.56
2 560	**243**	**12.502 8**	**1.134 3**	**1 558.4**
2 816	219	13.476 5	1.360 6	1 579.24

根据表3可知:与第3.4节时间复杂度的分析结论一致,内存大小的划分宽度显著影响到平均备选逻辑服务数和CLSG算法的时间开销.随着划分宽度的增加,CLSG算法的执行时间呈指数减少,而HGA算法的执行时间则呈线性递减.同时,最小资源成本随着划分宽度的增加而增加,这是由于划分宽度增加导致可行解减少,进而降低了更好的资源分配量出现的概率.

对于 Ent-MDLP 资源划分策略,响应时间属性划分区间数取值不同时,实验结果见表 4.由表 4 可知:与等宽资源划分策略相比,不同划分区间数获得的平均备选逻辑服务数量明显减少,且基本不影响 CLSG 算法的执行时间.由于不同划分区间数对应的平均备选逻辑服务数差别不大,因此 HGA 算法的执行时间基本没有变化.另外,不同于等宽资源划分策略,最小资源成本与划分区间数并不具有线性关系.

Table 4 Effect of Ent-MDLP resource partition

表 4 Ent-MDLP 资源划分策略的影响

响应时间属性划分区间数	平均备选逻辑服务数(个)	CLSG 算法执行时间(s)	HGA 算法执行时间(s)	最小资源成本
2	3	5.988 4	0.838 68	1 897.32
4	8	6.427 4	1.047 6	1 822.4
6	9	6.512 8	1.051 4	1 630.7
8	7	6.744 8	1.040 9	1 698.24
10	6	6.391 2	1.038 1	1 762.66
15	7	6.478 3	1.041 8	1 663.88
30	7	6.461 1	1.042 5	1 728.34
50	8	6.599 9	1.045 1	1 809.86
80	**9**	**6.722 1**	**1.044 7**	**1 562.56**
100	7	7.000 2	1.041 5	1 775.46

比较表3和表4可知:Ent-MDLP资源划分策略得到的最小资源成本高于较小划分宽度的等宽资源划分策略;但是当后者的资源划分宽度增加时,这种差别变得越来越不明显;特别是当前者的划分区间数等于 80 时,其最小资源成本接近于后者的划分宽度为2 560时的结果.而此时,前者的CLSG算法时间开销仅近似为后者的一半,且 HGA 算法的时间开销也略低于后者.值得注意的是:在 Ent-MDLP 资源划分策略中,由于划分区间数与最小资源成本不具有线性关系,因此在实际应用中,需要谨慎地选取合适的划分区间数.

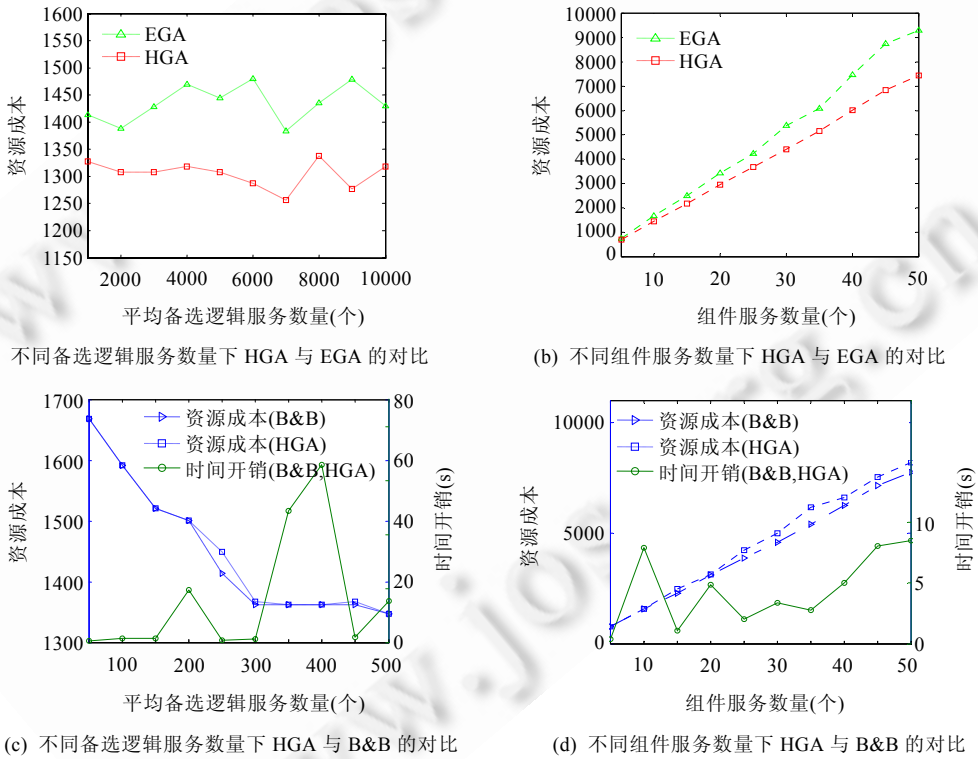
5.2.3 与其他算法相比较

为了保证收敛到全局最优解,并提高遗传算法的局部搜索能力,本文在基于精英保留策略的遗传算法(EGA)的基础上引入了局部搜索变异,进而提出了 HGA 算法.因此,这里将该算法与 EGA 算法进行解的质量(即最小资源成本)和求解效率方面的比较,以验证这种改进的有效性.目前,求解多维多选择背包问题大多采用分支定界法(branch and bound,简称 B&B),其可扩展性较差,仅适用于求解规模较小的优化问题.但由于该方法是一种确定算法,能够求得最优解,因此实验中将 HGA 算法与 B&B 算法在小规模可行解空间的解相比较,从而验证其对最优解的近似程度.另外,为便于在不同可行解规模上比较不同算法,这里采用等宽资源划分策略,并且 CPU、内存、带宽和磁盘 I/O 的划分宽度分别为 1 个、512MB、10Mbps 和 30MB/s.

(1) 最小资源成本

本组实验分别对比不同平均备选逻辑服务数量和组件服务数量下 HGA 算法、EGA 算法以及 B&B 算法求得的最小资源成本.对于不同平均备选逻辑服务数量,依然采用图 1 所示的 SBS 示例,并从等宽资源划分获得的整个备选逻辑服务集中随机抽取一定数量的逻辑服务;而对于不同组件服务数量,则随机地从该示例的组件服务集合中抽取一定数量的组件服务,并根据顺序结构组合成 SBS.这里均采用增量抽样方式,而且 SBS 的结构只会影响端到端响应时间的计算,并不直接影响优化模型的求解和对比.因此为了简单起见,实验采用顺序结构.

在 HGA 算法与 EGA 算法的对比实验中,设置 EGA 算法的变异概率 P_m=0.07,其他参数与 HGA 算法相同.在不同平均备选逻辑服务数量和组件服务数量下,两种算法求得的最小资源成本分别如图 11(a)、图 11(b)所示.



(a) 不同备选逻辑服务数量下 HGA 与 EGA 的对比

(b) 不同组件服务数量下 HGA 与 EGA 的对比

(c) 不同备选逻辑服务数量下 HGA 与 B&B 的对比

(d) 不同组件服务数量下 HGA 与 B&B 的对比

Fig.11 Minimal resource cost comparison of different algorithms

图 11 不同算法的最小资源成本对比

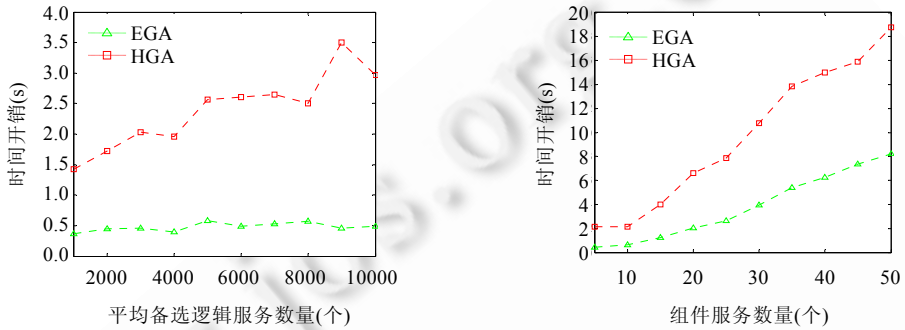
由图 11(a)可知:在不同平均备选逻辑服务数量下,HGA 算法求得的最小资源成本明显低于 EGA 算法.而根据图 11(b):随着组件服务数量(平均备选逻辑服务数量固定为 5 000)的增加,HGA 算法的最小资源成本与 EGA

算法结果的差距逐渐增大,表明本文的 HGA 算法在引入局部搜索变异算子后,解的质量明显优于 EGA 算法.

在 HGA 算法与 B&B 算法的对比实验中,以后者求得最优解时消耗的时间为前者执行的终止条件,并且对比不同平均备选逻辑服务数量和组件服务数量下两种算法求得的最小资源成本,结果如图 11(c)、图 11(d)所示.由图 11(c)可知:当平均备选逻辑服务数量较小或 B&B 算法时间开销较大时,HGA 算法的解的质量与 B&B 算法非常近似,可以认为此时 HGA 算法能够求得最优解.同样,由图 11(d)可知:当组件服务数量(平均备选逻辑服务数量固定为 300)较小时,HGA 算法求得的解接近最优解.随着组件服务数量的增加,其求解的最小资源成本与最优解的差值平均为 440 左右,且并未呈现出明显的差距增大趋势,表明本文的 HGA 算法能够在不同问题的规模上求得质量良好的近似最优解.

(2) 时间开销

图 11(a)、图 11(b)对比了 HGA 算法与 EGA 算法在不同平均备选逻辑服务数量和组件服务数量下的最小资源成本,这里给出了两种算法在求得近似最优解时消耗的时间对比结果,如图 12(a)、图 12(b)所示.



(a) 不同备选逻辑服务数量下的时间开销对比 (b) 不同组件服务数量下的时间开销对比

Fig.12 Time overhead comparison of different algorithms

图 12 不同算法的时间开销对比

图 12(a)表明:HGA 算法求得近似最优解的时间开销高于 EGA 算法,且随着平均备选逻辑服务数量的增加近似呈线性增长;而 EGA 算法则具有更好的求解效率,时间开销随平均备选逻辑服务数量的增加并无明显变化.而由图 12(b)可知:在不同组件服务数量下,HGA 算法的时间开销仍然高于 EGA 算法.同时,随着组件服务数量的增加,二者的时间开销均呈线性增加,且前者的增加趋势稍高于后者.本组实验说明:局部搜索变异在显著提高解的质量的同时,也增加了一定的时间代价.但是,由于该求解过程是在离线阶段完成的,因此对于成本敏感的 SBS 资源分配来说,增加求解代价是值得的.

6 结束语

本文探讨了一种适用于云环境中 SBS 的资源优化分配方法.该方法基于 SBSE 思想,将资源优化分配策略的确定转换为服务选取问题,从而采用混合遗传算法搜索最优解.在转换问题时,定义了逻辑服务的概念,使得问题不仅从形式上与服务选取相匹配,而且还考虑了云环境中的资源约束,因此更具实际应用价值.实验证明了提出的模型和算法在确定 SBS 各组件服务资源分配量方面的有效性,且在遗传算法中引入精英保留策略与局部搜索变异,对于全局收敛性和加快收敛速度具有显著作用.另外,结果表明:提出的资源划分策略对优化算法求解效率和解的质量均具有一定的影响,因此有助于在实际问题中指导如何选取合适的资源分配粒度.本文扩展了 SBSE 方法在云计算领域优化 SBS 系统设计方面的具体应用,且所得优化结果对于提高 SBS 运行时的环境适应能力具有一定的帮助.在实际中,由于运行环境的动态性,如负载减少或增加、主机或网络故障导致的资源不可用等,确定的最优资源分配策略可能失效,因此在下一步工作中,有必要深入探索能够适应环境变化的 SBS 云资源动态分配方法.

References:

[1] Durao F, Carvalho JFS, Fonseka A, Garcia VC. A systematic review on cloud computing. The Journal of Supercomputing, 2014, 68(3):1321−1346. [doi: 10.1007/s11227-014-1089-x]

[2] Chen K, Zheng WM. Cloud computing: System instances and current research. Ruan Jian Xue Bao/Journal of Software, 2009,20(5): 1337−1348 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[3] Yau SS, Ye N, Sarjoughian HS, Huang D, Roontiva A, Baydogan MG, Muqsith MA. Toward development of adaptive service-based software systems. IEEE Trans. on Services Computing, 2009,2(3):247−260. [doi: 10.1109/TSC.2009.17]

[4] Alhamad M, Dillon T, Chang E. A survey on SLA and performance measurement in cloud computing. In: Meersman R, Dillon T, Herrero P, eds. Proc. of the Move to Meaningful Internet Systems (OTM 2011). LNCS 7045, Heidelberg: Springer-Verlag, 2011. 469−477. [doi: 10.1007/978-3-642-25106-1_4]

[5] Jennings B, Stadler R. Resource management in clouds: Survey and research challenges. Journal of Network and Systems Management, 2014. [doi: 10.1007/s10922-014-9307-7]

[6] Harman M, Mansouri SA, Zhang Y. Search-Based software engineering: Trends, techniques and applications. ACM Computing Surveys, 2012,45(1):11−61. [doi: 10.1145/2379776.2379787]

[7] Zeng L, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H. QoS-Aware middleware for Web services composition. IEEE Trans. on Software Engineering, 2004,30(5):311−327. [doi: 10.1109/TSE.2004.11]

[8] Zhu Q, Agrawal G. Resource provisioning with budget constraints for adaptive applications in cloud environments. IEEE Trans. on Services Computing, 2012,5(4):497−511. [doi: 10.1109/TSC.2011.61]

[9] Pandey S, Wu L, Guru SM, Buyya R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Proc. of the 24th IEEE Int'l Conf. on Advanced Information Networking and Applications (AINA). Perth, 2010. 400−407. [doi: 10.1109/AINA.2010.31]

[10] Yau SS, An HG. Adaptive resource allocation for service-based systems. Int'l Journal of Software Informatics, 2009,3(4):483−499. [doi: 10.1145/1640206.1640209]

[11] Hossain MS, Hassan MM, Qurishi MA, Alghamdi A. Resource allocation for service composition in cloud-based video surveillance platform. In: Proc. of the 2012 IEEE Int'l Conf. on Multimedia and Expo Workshops (ICMEW). Melbourne, 2012. 408−412. [doi: 10.1109/ICMEW.2012.77]

[12] Chen Y, Iyer S, Liu X, Milojicic D, Sahai A. SLA decomposition: Translating service level objectives to system level thresholds. In: Proc. of the 4th Int'l Conf. on Autonomic Computing (ICAC). Florida, 2007. 3. [doi: 10.1109/ICAC.2007.36]

[13] Schulte S, Schuller D, Hoenisch P, Lampe U, Steinmetz R, Dustdar S. Cost-Driven optimization of cloud resource allocation for elastic processes. Int'l Journal of Cloud Computing, 2013,1(2):1−14.

[14] Canfora G, Penta MD, Esposito R, Villani ML. An approach for QoS-aware service composition based on genetic algorithms. In: Proc. of the Conf. on Genetic and Evolutionary Computation (GECCO). Washington, 2005. 1069−1075. [doi: 10.1145/1068009.1068189]

[15] Ma Y, Zhang C. Quick convergence of genetic algorithm for QoS-driven Web service selection. Computer Networks, 2008,52(5): 1093−1104. [doi: 10.1016/j.comnet.2007.12.003]

[16] Martens A, Koziolek H, Becker S, Reussner R. Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In: Proc. of the 1st Joint Int'l Conf. on Performance Engineering (WOSP/SIPEW). 2010. 105−116. [doi: 10.1145/1712605.1712624]

[17] Wada H, Suzuki J, Yamano Y, Oba K. Evolutionary deployment optimization for service-oriented clouds. Software: Practice and Experience, 2011,41(4):469−493. [doi: 10.1002/spe.1032]

[18] Frey S, Fittkau F, Hasselbring W. Search-Based genetic optimization for deployment and reconfiguration of software in the cloud. In: Proc. of the 2013 Int'l Conf. on Software Engineering (ICSE). San Francisco, 2013. 512−521. http://dl.acm.org/citation.cfm?id=2486856

[19] Shi XL, Xu K. Utility maximization model of virtual machine scheduling in cloud environment. Chinese Journal of Computers, 2013,36(2):252−262 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2013.00252]

[20] Fayyad UM, Irani KB. Multi-Interval discretization of continuous-valued attributes for classification learning. In: Proc. of the 13th Int'l Joint Conf. on Artificial Intelligence. Chambery, 1993. 1022−1027. http://ijcai.org/Past%20Proceedings/IJCAI-93-VOL2/PDF/022.pdf

[21] Chen B, Peng X, Zhao W. Towards runtime optimization of software quality based on feedback control theory. In: Proc. of the 1st Asia-Pacific Symp. on Internetware. Beijing, 2009. 1−8. [doi: 10.1145/1640206.1640216]

[22] Kousiouris G, Menychtas A, Kyriazis D, Gogouvitis S, Varvarigou T. Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in cloud platforms. Future Generation Computer Systems, 2014,32:27−40. [doi: 10.1016/j.future.2012.05.009]

[23] Rao J, Wei Y, Gong J, Xu CZ. QoS guarantees and service differentiation for dynamic cloud applications. IEEE Trans. on Network and Service Management, 2013,10(1):43−55. [doi: 10.1109/TNSM.2012.091012.120238]

[24] Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V. Support vector regression machines. Advances in Neural Information Processing Systems, 1997,9:155−161. http://ece.ut.ac.ir/classpages/F83/PatternRecognition/Papers/SupportVectorMachine/support-vector.pdf

[25] Cardellini V, Casalicchio E, Grassi V, Lannucci S, Presti FL, Mirandola R. MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. IEEE Trans. on Software Engineering, 2012,38(5):1138−1159. [doi: 10.1109/TSE.2011.68]

[26] Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient *k*-means clustering algorithm: Analysis and implementation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002,24(7):881−892. [doi: 10.1109/TPAMI.2002.1017616]

[27] Ardagna D, Pernici B. Adaptive service composition in flexible processes. IEEE Trans. on Software Engineering, 2007,33(6):369−384. [doi: 10.1109/TSE.2007.1011]

[28] Radcliffe NJ, Surry PD. Formal memetic algorithms. Evolutionary Computing, 1994,865:1−16. [doi: 10.1007/3-540-58483-8_1]

[29] Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Trans. on Neural Networks, 1994,5(1):96−101. [doi: 10.1109/72.265964]

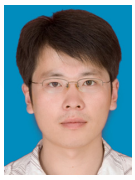
[30] Leitner P, Hummer W, Dustar S. Cost-Based optimization of service compositions. IEEE Trans. on Services Computing, 2013,6(2):239−251. [doi: 10.1109/TSC.2011.53]

[31] Ustun B, Melssen WJ, Buydens LMC. Facilitating the application of support vector regression by ssing a universal Pearson VII function based kernel. Chemometrics and Intelligent Laboratory Systems, 2006,81:29−40. [doi: 10.1016/j.chemolab.2005.09.003]

附中文参考文献:

[2] 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. http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[19] 师雪霖,徐恪.云虚拟机资源分配的效用最大化模型.计算机学报,2013,36(2):252−262. [doi: 10.3724/SP.J.1016.2013.00252]



赵秀涛(1985－),男,山东临沂人,博士生,主要研究领域为服务计算,自适应软件系统.

张长胜(1980－),男,博士,副教授,主要研究领域为服务计算,进化计算.

张斌(1964－),男,博士,教授,博士生导师,CCF高级会员,主要研究领域为Web信息处理,服务计算,数据挖掘.