基于强化学习的 SBS 云应用自适应性能优化方法

闫永明 张 斌 郭 军 孟 煜
(东北大学计算机科学与工程学院 沈阳 110819)

摘 要 自适应的调整云应用所占用的资源是一种有效的保障云应用性能的方法,但传统的决策方法面向基于服务的系统(Service-Based System,SBS)时会存在一些问题,例如基于应用系统性能模型的决策方法不能很好适应云环境下 SBS 的动态变化,基于智能优化算法的决策方法效率较低.该文提出了一种基于强化学习的 SBS 云应用自适应性能优化方法.在该方法中,该文建立了自适应基本要素之间相互关系的特征描述框架,利用高层次的系统行为指标(如响应时间、用户并发量、资源量等)来描述系统性能的优化目标等.为了应对云环境以及 SBS 的动态变化,该文的方法采用了无模型(model-free)的在线学习算法,当用户并发量发生变化导致系统的预期行为发生偏差时,该方法通过不断重复"执行-积累-学习-决策"的过程,可以不断的积累经验数据并优化决策结果.为了保证自适应优化的高效性,该文提出了一种引导算子,可以有效的缩小候选自适应动作的范围,提高算法的学习效率.该文实现了以一个 SBS 为例的原型框架,使用该框架的实验结果证明了该文提出方法的有效性.

关键词 自适应;强化学习;资源调整;云应用;基于服务的系统;云计算
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2017.00464

A Reinforcement Learning-Based Self-Adaptation Performance Optimization Approach for SBS Cloud Application

YAN Yong-Ming ZHANG Bin GUO Jun MENG Yu
(*School of Computer Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract Adaptive adjustment of cloud applications resources is an effective way to guarantee the performance of cloud applications. There are some problems of the traditional decision method for SBS (Service-based System), such as the decision method based on application system performance model cannot adapt to the dynamic change of SBS in cloud environment, the efficiency of the decision making method based on intelligent optimization algorithm is not high. To solve the disadvantages of the existing methods, a self-adaptation performance optimization approach for SBS cloud application based on reinforcement learning is proposed by this paper. In this approach, the feature description framework of the relationship of the basic key adaptation elements is established, and the system's optimization target for system performance and key elements of learning algorithm are described as high-level system behavior indicators (such as response time, user concurrency, resource quantity, etc.). To cope with dynamic change of the cloud environment and SBS, this paper adopts a model-free online learning algorithm. When the cloud application deviates from the desired behavior caused by the changes in the number of user concurrency, the approach will repeat the "execution-accumulation-learning-decision" process, which can

收稿日期:2016-05-31;在线出版日期:2016-09-19.本课题得到国家自然科学基金(61572117,61300019,61370155)、省科技项目攻关项目(2015302002)、中央高校东北大学基本科研专项基金(N140406002,N150404008)资助.**闫永明**,男,1981 年生,博士研究生,主要研究方向为云服务性能优化.E-mail:yym_sy@163.com.**张 斌**(通信作者),男,1964 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究方向为云计算、服务计算.E-mail:zhangbin@mail.neu.edu.cn.**郭 军**,男,1974 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为云计算、服务计算.**孟 煜**,男,1990 年生,博士研究生,中国计算机学会(CCF)会员,主要研究领域为云计算、服务预测及优化.

accumulate the corresponding empirical data in the process of dynamic decision, and optimize the decision-making results continuously through empirical data. To ensure the high efficiency of the adaptation optimization, this paper puts forward a shaping operator which can effectively narrow the scope of the candidate adaptation actions and improve the learning efficiency of the algorithm. Experimental results using a prototype framework in the context of a SBS application demonstrate the effectiveness of this approach.

Keywords　self-adaptation; reinforcement learning; resource adjustment; cloud application; service-based system; cloud computing

1　引　言

当前很多云计算平台(Amazon EC2、Microsoft Azure、Rackspace、阿里云等)都为部署在上面的应用系统提供了通过资源调整进行云应用系统性能优化的能力,这种能力使得云应用系统可以更加有效地应对云环境以及用户请求的实时变化,为云应用系统的性能保障提供了更加有效的支持[1-3].

基于服务的系统(Service-Based System,SBS)在各种应用领域(如科研、通信、金融)发挥着越来越重要的作用[4].同时,也有越来越多的服务提供者选择将 SBS 部署在云环境中(通常采用将具体组件服务部署到多台虚拟机(Virtual Machine,VM)上的方式[5]),以减少硬件的运营成本.为了保障部署在云环境下的 SBS 的服务质量,则可以通过调整组件服务所在的虚拟机的资源(CPU、内存等)来改善组件服务的服务能力,从而实现保障 SBS 整体性能的目的[6-7].

然而对组件资源进行自适应调整需要确定调整动作(如组件服务需要增加内存或增加 CPU、给组件服务增加 300 MB 或 500 MB 内存).但云环境下资源的使用状态是动态变化的,一个虚拟机能够增加多少资源(如 CPU、内存等)是受其所在物理机的资源使用情况所左右的,如果在系统运行之前预定义自适应调整,很有可能会出现自适应调整无法执行、不能实现预期优化目标等情况.例如预定义的自适应调整是给组件 S_1 增加 300 MB 内存,若 S_1 所在物理机的剩余内存不足 300 MB 时,则该自适应调整无法执行;若 S_1 需要增加 500 MB 内存才能使系统的响应时间满足服务等级协议(Service Level Agreement,SLA)的要求,则该自适应调整不能实现预期的优化目标.所以,为了在云环境下对 SBS 进行更有效的自适应性能优化,需要基于动态的自适应决策.

动态的自适应决策通常可以利用应用系统性能模型[8-11]或者智能优化算法[12-15]来实现.其中,利用应用系统性能模型进行自适应决策是在离线阶段建立应用系统资源、负载、性能三者关系的模型,然后在线阶段以系统当前的负载和性能目标作为性能模型的输入,计算得到资源的调整量;采用智能算法进行决策是将自适应方案的生成转换为最优化问题,即从所有可能的调整动作中选出满足优化目标且调整代价最小的动作组合.但是云环境以及部署在云环境中的 SBS 是动态变化的(如虚拟机资源增减、组件服务发生替换等),离线模型可能会存在不能很好地适应变化后环境的情况.而利用智能优化算法求解的效率通常很难满足在线自适应调整在高效性方面的要求(尽量减少 SLA 违例的时间),例如一个 SBS 由 i 个组件服务组成,每个组件服务所在的虚拟机都可以调整 CPU 和内存资源,CPU 的调整方案为 P 种(可调整的 CPU 数量为 $0\sim P$),内存的调整方案为 Q/M 种(可调整的内存总量为 Q,最小调整量为 M).那么智能优化算法需要在 $(P\times Q/M)^i$ 种动作组合中进行选择,这将耗费大量的计算时间.

针对这些问题,本文提出了一种基于强化学习的云应用自适应方法,该方法在动态决策的过程中积累相应的经验数据,并利用历史经验在线学习到资源调整的效果,通过不断重复"执行-积累-学习-决策"的过程,从而生成更好的资源调整方案,即通过在线学习不断优化决策结果.本文采用了 Q-learning 算法,该算法是一种无模型(model-free)的在线学习算法,在自适应的过程中不需要建立应用系统的性能模型,可以更好地应对自适应过程中的动态变化.此外,本文提出了一种引导算子,可以有效地缩小候选自适应动作的范围,从而提高算法的学习效率,保证自适应优化的高效性.

本文的主要贡献有:

(1)本文提出了一种基于强化学习的自适应方法,可以在动态决策的过程中积累相应的经验数据,并通过经验数据不断地优化决策结果,能够在自适应优化的过程中更好地应对云环境以及 SBS 的动态性;

(2)本文采用形式化的描述方法构建了云服务运行时自适应优化的表示模型,建立了可以完备描述自适应基本要素之间相互关系的特征描述框架;

(3)为了保证应用系统运行时自适应优化的高效性,本文提出了一种引导算子,该算子通过自适应动作所包含的特征对自适应动作进行筛选,可以有效地缩小候选自适应动作的范围,达到提高算法学习效率的目的.

本文第 2 节对所提出的方法进行了概述;第 3、4、5 节分别对自适应基础信息、自适应动态决策和在线学习过程进行描述;第 6 节给出相关的实例分析;第 7 节为实验及其结果分析;第 8 节介绍本文相关研究的现状;最后总结全文.

2 基于强化学习的自适应

2.1 基于强化学习的自适应模型

基于强化学习的决策可以看作是一个 Agent(应用系统也可以扮演 Agent 的角色)与部署环境之间的互动模型(interaction model)[16],如图 1 所示.在该模型中,Agent 可以立即监测到软件系统运行环境的状态,并获得当前环境的回报值(根据当前环境状态计算得到的一个数值).基于当前的环境状态(State) s_t 和回报值 r_t,Agent 选择一个动作(Action) a_t,并通过执行动作 a_t 来影响环境.在动作执行完成后,环境将转换到下一状态 s_{t+1},对应一个新的回报值 r_{t+1},Agent 将基于 s_{t+1} 和 r_{t+1} 选出一个新的动作 a_{t+1}.这种 Agent 和环境之间周期性的持续互动是实现自适应自主性的基础.

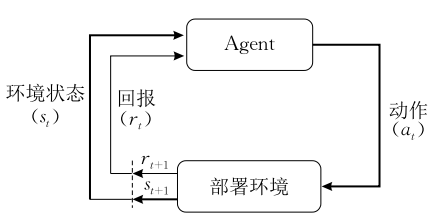


图 1 Agent 和环境的互动模型

但是上述模型应用在面向云应用的自适应性能优化中会存在以下问题:(1)在云应用系统性能符合 SLA 要求时,持续的频繁调整会导致不必要的成本,包括决策和执行时所使用的计算资源和额外增加的资源的成本;(2)由于基于资源调整的自适应优化过程中存在数量众多的调整动作,且在动作选择时有可能会选择对云应用系统性能造成不良影响的动作(例如在 SBS 系统正常运行时,选择并执行减少 VM 资源的动作可能会导致云应用系统性能下降,发生 SLA 违例),从而影响强化学习算法的收敛速度和自适应的优化效果.

为了解决上述问题,本文采用如图 2 所示的模型.针对第一个问题,该模型采用了事件触发机制(如图 2 中的 $|E|$ 所示),即 Agent 根据环境状态 s_t 和回报值 r_t 选择一个动作 a_t 前,需要先判断是否有触发事件(Event)被触发,如果触发事件被触发,则开始执行系统的自适应调整,从而避免了云应用系统的性能符合 SLA 要求后,还继续进行"无效"的调整.针对第 2 个问题,本文利用被触发的事件对所有可执行的 Action 进行筛选,保留与触发事件有相同目标趋势的 Action 作为候选集(例如云应用的响应时间超过 SLA 规定的上限时触发事件 e 被触发,e 的目标趋势是提升组件服务的性能,所以将增加资源类的 Action 作为候选集),即对 Agent 进行了引导(shaping),限定其在特定范围的 Action 集合中进行选择,从而提升强化学习算法的学习效率.

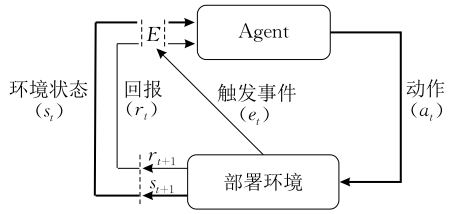


图 2 改进后的互动模型

为了将上述的 Agent-环境互动模型应用到面向 SBS 云应用的自适应资源调整中,本文假设 SBS 系统是由 i 个组件服务 $c_1, c_2, \cdots, c_i$ 构成,其部署的云环境包含 j 个物理机 $pm_1, pm_2, \cdots, pm_j$,各物理机上的可分配的资源量为 $R_k^{\alpha}(1\leq\alpha\leq n, 1\leq k\leq j)$,$\alpha$ 为资源类型.并设触发事件的有限集合为 E,自适应方案的有限集合为 P,每个自适应方案($p\in P$)包含 m 个自适应调整$\{a_1, a_2, \cdots, a_m\}$,对于每一个 a_m,其调整的资源量为 $r_{m,k}$(即 a_m 调整的资源需要由物理机 pm_k 提供).当一个触发事件($e\in E$)被触发时,如果 Agent 总是可以选出一个最佳的自适应方案($p\in P$),那么可以认为 Agent 拥有一组最优的自适应方案,使得 SBS 系统的响应时间满足 SLA 的约束时,自适应调整的资源量满足当前云环境的可用

资源约束. 则该模型的形式化描述如下：

$$D: E \rightarrow P \tag{1}$$

$$\text{s.t. } RT_{sla\_low} \leq RT \leq RT_{sla\_up} \tag{2}$$

$$\sum_{l=1}^{m} r_{m,k} \leq R_k^{\alpha},\ k=1,2,\cdots,j;\ \alpha=1,2,\cdots,n \tag{3}$$

式(1)中的 D 为一个决策器，表示在 SBS 运行期间可以将每一种触发事件($e \in E$)映射到一个合适的自适应方案上($p \in P$)；式(2)为 SBS 系统响应时间 RT 的约束；式(3)表示一台物理机内的自适应调整的资源量不能超过其最大可用资源数量. 所以，面向 SBS 云应用的自适应决策的主要目标是找到一个最优的决策器 D，在面对环境和用户访问动态变化时，决策器 D 可以自动地制定自适应优化方案.

2.2　方法概述

在 Agent-环境互动模型的基础上，本文提出的基于强化学习的 SBS 云应用自适应的基本过程如图 3 所示.

主要分为监测、触发、决策、执行、评估和学习 6 个阶段：监测阶段主要是获得与 SBS 系统相关的云环境状态信息(如 VM 占用的资源量等)以及 SBS 系统的运行状态信息(如响应时间、用户并发量等)，然后将这些信息传递到触发阶段. 在触发阶段时，需要判断监测信息是否满足触发事件中的触发条件，若满足触发条件则进入决策阶段. 另外，如果有自适应调整动作未执行完成时，则不触发新的触发事件.

在决策阶段时，Agent 根据监测信息和决策经验信息选出最优的自适应调整动作. 在执行阶段时，Agent 执行上一阶段选择的自适应调整动作，完成对云环境中虚拟机资源的调整. 当自适应调整动作执行完成后，Agent 将利用适应度函数(fitness function)评估调整后的云环境所能获得的回报，并利用 Q-learning 对所执行的自适应调整的经验(即评估阶段获得的回报值)进行累积，为以后的决策提供经验.

在上述过程中，监测和执行均可以通过使用现有的软件来完成，例如利用 collectd 获得应用系统性能信息，利用 libvirt API 获得 VM 资源占用量和调整 VM 的资源. 触发则是依次判断监测的数据是否满足相应触发事件. 所以，在接下来的第 3、4、5 节中将重点介绍自适应基础信息形式化描述、如何在决策的过程中使用 Q-learning 学习到的经验和如何累积自适应调整之后的经验.

3　自适应基础信息

本节介绍了基于强化学习自适应过程中的基础信息，包括：系统行为的描述指标(Key Performance Indicator，KPI)、SBS 系统的部署信息(包括物理机、虚拟机和组件服务的部署信息)、自适应优化目标(Goal)和触发事件(Event)，并在此基础上对 Q-learning 的 State 和 Action 进行了形式化描述.

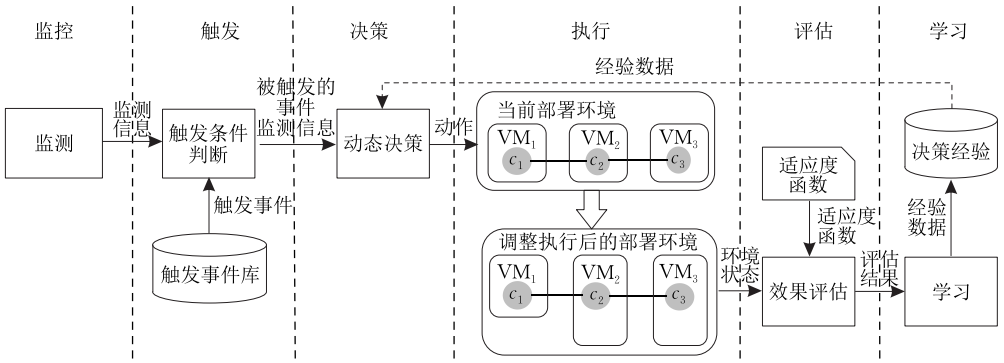


图 3　自适应基本过程

3.1　关键性能指标

假设预期系统行为可以通过一系列的指标来进行描述，这些指标通过获取系统性能特定的方面获得，例如 CPU 利用率、内存占用率、系统响应时间、服务等级(SLA)等[6]，并将这些指标称为关键性能指标. KPI 可以用来描述单个组件或整个系统的状态，例如可以使用 c. KPI 来表示组件服务 c 的 KPI 值.

KPI 的定义包括名称、数据类型、组合方程和边界值(Relevance Margin). 其中组合方程用来说明如何计算系统全局或部署了组件服务的单个虚拟机的 KPI 值，而边界值则用来说明两个同类型的 KPI 值是否相同，即当两个同类型的 KPI 值的差值小于边界值时，认为这两个 KPI 值相同. 本文中所定义的 KPI 如下所示：

KPI cpu_use：double **CombFunc** Sum **RMargin** 0.01

KPI mem_use：double **CombFunc** Sum **RMargin** 0.1

KPI user_con：int **CombFunc** Sum **RMargin** 0

KPI resp_time：double **CombFunc** Tree **RMargin** 0.01

在本文中，我们假设系统级的 KPI 值都可以通过系统中每个单独的组件服务的 KPI 值组合获得，即系统级的 KPI 值可以通过 c.KPI 对应的组合方程计算得到，其中组合方程是单调递增或单调递减的，如 CPU、内存、网络和用户并发量的 KPI 值.但对于响应时间的 KPI 值，由于本文讨论的云应用系统是由多个组件服务构成的 SBS 系统，其业务流程中会存在顺序、循环、并行和选择这 4 种结构，可以采用标记树对 SBS 的组合流程进行等价转换并计算其响应时间[9].

3.2 系统部署信息

系统部署信息描述了云环境中与 SBS 云服务部署相关的物理机、虚拟机以及 SBS 系统中组件服务的部署信息，以虚拟机的部署信息为例，其具体形式如下：

Virtual Machine Specification VM_name

Deployment：(component$_1$，…，component$_n$)

Resource：((Resource$_1$，Count)，…，(Resource$_n$，Count))

Status：[RUNNING/UNUSED/UNALLOCATED]

Deployment 和 Resource 分别描述了该虚拟机上部署的组件服务的情况和该虚拟机的资源信息.Status 描述了虚拟机的运行状态，共有 3 种类型：RUNNING、UNUSED 和 UNALLOCATED，分别表示该虚拟机处于运行、停止和未分配状态，其中，处于 UNALLOCATED 状态表示未创建该虚拟机的运行实例.

3.3 自适应优化目标和触发事件

每个自适应目标使用 KPI 来描述特定的自适应优化目标，并使用 Above、Below 和 Between 关键字来确定自适应目标可接受的指标的范围，每个自适应目标可以通过人工设定或从 SLA 约束中自动抽取等方式获得，本文采用人工设定的方式.具体形式如下：

Goal goal_name：kpi_name **Above** thr_lower

Goal goal_name：kpi_name **Below** thr_upper

Goal goal_name：kpi_name **Between** thr_lower thr_upper

其中：Above 目标表示应该使 KPI 的值高于所给定的阈值 thr_lower；Below 目标表示应该使 KPI 的值低于所给定的阈值 thr_upper；between 目标表示应该使 KPI 的值保持在所给定的阈值区间.

触发事件为自适应调整的触发条件，每一个触发事件关联着一个特定的自适应目标，并说明事件触发后自适应调整所应达到的目标趋势，具体形式如下：

Event goal_name.event_name：kpi_name [>/<] thr

Target target_name

其中：goal_name.event_name 为自适应目标和触发事件名称的组合，描述了两者的对应关系；kpi_name [>/<] thr 为判定条件，表示当 KPI 的值大于/小于阈值 thr 时满足触发条件；Target 表示触发事件触发后自适应调整所应达到的目标趋势，包括两种类型：IMPR_PERF 和 DEGR_PERF，分别表示触发事件触发后应执行提升或降低组件服务性能的自适应调整.

在实际系统中使用上述触发事件时，为了避免系统 KPI 值在短时间内超过阈值所造成的不必要的触发.本文的触发机制采用基于时间窗的方法，即当时间窗内的系统 KPI 值有 $x\%$ 超过触发事件中的阈值时，才认为触发事件会被触发.其中，时间窗的宽度和 x 的值可以根据经验来设定.

根据自适应目标抽取对应的触发事件，触发事件的判定条件可以依据自适应目标中的关键字(Above、Below 和 Between)和阈值来确定，具体的对应规则如表 1 所示.在确定触发事件的判定条件后，可以根据判定条件确定触发事件的目标趋势，具体为当判定条件为“>”时，目标趋势为“IMPR_PERF”；当判定条件为“<”时，目标趋势为“DEGR_PERF”.

表 1 自适应目标生成触发事件规则

自适应目标	触发事件 1	触发事件 2
Above thr_lower	< thr_lower	
Below thr_upper	> thr_upper	
Between thr_lower thr_upper	< thr_lower	> thr_upper

3.4 State 和 Action 的形式化表示

State 和 Action 的形式化表示定义了系统的问题空间和解决方案空间，是设计性能优化方法中基于 Q-learning 动态决策的关键步骤.本文设环境状态的集合为 $S=\{s_1,s_2,\cdots,s_n\}$，每个环境状态 s_i 包括云环境中与 SBS 系统相关的 VM 的资源拥有量和 SBS 系统当前的负载所处的区间，其形式可以表示为

State state_name

Resource：($\text{Resource}_{\text{vm}_1}$，$\text{Resource}_{\text{vm}_2}$，…，$\text{Resource}_{\text{vm}_n}$)

Load：load_type

其中，Resource 为与 SBS 系统相关的 VM 的资源拥有量，本文为了减少环境状态的数量，将 VM 的资源拥有量设定为固定的 6 种：NANO、MICRO、SMALL、MEDIUM、LARGE 和 EXLARGE. 采用这种设定，一方面是参考亚马逊 EC2 中虚拟机实例的设定，另一方面是通过对 SBS 系统进行基准测试，参考基准测试数据所设定给的(例如，通过基准测试可以得到当 VM 的 CPU 数量为 8，内存为 8GB 时，再继续增加 VM 的资源不会使 SBS 的性能有明显的提升)，具体如表 2 所示. Load 是 SBS 系统当前的负载类型，即当前的并发用户数量所处的区间，其设定类似于虚拟机资源类型的设定，具体如表 3 所示.

表 2　虚拟机资源类型

类型	CPU 数量	内存数量/GB
NANO	1	0.8
MICRO	2	1.0
SMALL	3	2.0
MEDIUM	4	3.0
LARGE	6	4.0
EXLARGE	8	8.0

表 3　负载类型

负载类型	区间
TYPE_I	[1，100]
TYPE_II	(100，150]
TYPE_III	(150，180]
TYPE_IV	(180，200]
TYPE_V	(200，+∞]

对于 Action，本文设自适应动作的集合为 $A=\{a_1,a_2,\cdots,a_m\}$，每个自适应动作 a_j 包括自适应原语和动作执行后的目标趋势，具体形式如下：

Action action_name

Primitive：primitive_name(object，ori_res，des_res)

Target：[IMPR_PERF/DEGR_PERF]

其中，Primitive 为自适应原语，即每个自适应调整方式的具体执行指令，其第 1 个参数为调整的目标 VM，第 2、3 个参数分别为 VM 调整前后所占有的资源数量，例如 addRes(VM1，SMALL，MEDIUM 是将 VM_1 的资源从 SMALL 调整为 MEDIUM. Target 是自适应原语执行后所实现的目标趋势，与触发事件中的目标趋势相同.

当 SBS 由 i 个组件服务构成时，环境状态的集合 S 中共有 $x^i\times y$ 个环境状态，动作集合 A 中共有 $i\times x(x-1)\times y$ 个自适应动作. 其中，x 为虚拟机资源类型，y 为负载类型. 在本文中 x 和 y 的值分别为 6 和 5，即根据本文的设定，共有 $6^i\times 5$ 个环境状态以及 $30i\times 5$ 个自适应动作.

在确定所有环境状态和自适应动作后，Agent 需要生成 $(6^i\times 5)\times(30i\times 5)$ 维的矩阵 $\boldsymbol{Q}$，其元素 $\boldsymbol{Q}(s,a)$ 为环境处于环境状态 s 时选择自适应动作 a 的经验值. 在选择自适应动作时，由于计算所有自适应动作的选择概率需要耗费大量时间和资源，所以本文通过两个步骤来减少候选自适应动作的范围：首先在确定当前的环境状态后，可以选出当前环境状态下可以执行的自适应动作作为候选自适应动作集，该候选集中共有 $30i$ 个自适应动作；然后利用引导算子(具体见第 4 节)对候选集进行筛选，最终的候选集中共包括 $15i$ 个自适应动作.

4　自适应动态决策

当触发条件被触发后，自适应动态决策过程开始执行，即根据当前的环境状态和决策经验选择合适的 Action，主要分为两个步骤：首先，利用环境状态和引导算子从自适应动作集合 A 中选出候选动作集合 A_c ($A_c\subset A$)，以减少候选 Action 的搜索空间. 然后，利用软最大化(soft-max)方法在 A_c 中选择合适的待执行 Action.

其中，引导算子从自适应动作集合 A 中选出候选动作集合 A_c ($A_c\subset A$)的原则为：对于任意 $a_j\in A$，若 a_j 中的 Target 与被触发的触发事件中的 Target 相同，则将 a_j 加入候选集 A_c 中. 例如，当触发事件 **Event** $\text{resp\_time\_goal}_1.\text{event}_2$：resp_time＞19 s **Target** IMPR_PERF 被触发时，从集合 A 中挑选 Target 为 IMPR_PERF 的 Action 放入候选动作集合 A_c. 构建候选集 A_c 可以视为对 Agent 进行引导(shaping)，限定了 Agent 在特定范围内选择 Action，避免随机选择 Action 降低 Q-learning 的学习效率.

利用软最大化方法选择待执行 Action 时，认为在环境状态 s 时选择自适应动作 a ($a\in A_c$)的概率与 $e^{\frac{Q(s,a)}{\tau}}$ 成正比，具体为

$$p_{s,a}=e^{Q(s,a)/\tau}\Big/\sum_a e^{Q(s,a)/\tau} \tag{4}$$

式中：$\boldsymbol{Q}(s,a)$ 为环境处于环境状态 s 时选择自适应动作 a 的经验值；τ ($\tau>0$)是选择参数，当 τ 较大时表示任意自适应调整被选中的概率几乎一样，减少 τ 则拥有最大值的自适应调整被选中的可能性更大，若 $\tau\to 0$ 则总是选择最优的行为. 利用式(4)，

Agent 将计算动作候选集 A_c 中每个自适应动作的选择概率，并选择概率值最大的自适应动作作为结果. 若多个自适应动作的选择概率相同时，则随机选择一个自适应动作，被选择的自适应动作将在执行阶段被执行.

5 在线学习

当选择的自适应动作 a 执行完成后，环境状态由 s 转换为 s'，Agent 可以利用适应度函数来获得相应的回报 $r(s,a)$，并利用获得的回报 $r(s,a)$ 来更新 Q 值(Q-values)，以完成在线学习过程.

在线学习过程中，合理高效的适应度函数使自适应动作的选择准确、高效，本文将自适应动作 a 执行后的代价和收益作为其执行后的回报，即利用自适应动作执行所需要付出的成本和对系统性能影响来评价自适应动作是否合理. 适应度函数跟所调整的资源的成本、系统性能的变化量以及所调整资源是否满足资源量的约束这 3 个属相相关，可以定义为

$$r(s,a) = \omega_c c_{\Delta R} + \omega_w \Delta W + RP \tag{5}$$

其中：ω_c 和 ω_w 是适应度函数相关属性的权重；$c_{\Delta R}$ 是调整资源的成本；ΔW 是 SBS 系统性能的变化量；RP 是资源调整量超过可利用资源量时的惩罚.

调整资源的成本 $c_{\Delta R}$ 是指自适应调整在执行前后不同资源变化量的费用之和，即

$$c_{\Delta R} = \sum_{r \in \{\mathrm{CPU},\mathrm{Mem}\}} U_r(m - m') \tag{6}$$

式中：CPU、Mem 分别表示处理器、内存等资源；U_r 表示每种资源的单位价格，可通过建立资源的定价模型来确定，如市场上一条容量为 4GB 的 DDR3 内存条的价格是 220 元左右，则可以算出内存的单位价格为 220 元/4096 MB=0.054 元/MB；m，m' 分别表示自适应调整在执行前和执行后VM占有的各类资源的总量. 当增加资源时，$c_{\Delta R}$ 为负数，释放资源时，$c_{\Delta R}$ 为正数. SBS 系统性能变化量 ΔW 是指自适应调整导致的 SBS 系统的不同性能指标(如响应时间、可靠性和吞吐量等)的变化，在本文中主要是指响应时间的变化量. 当自适应动作调整的资源量超过可用的资源量时，惩罚量 RP 为一个负数，否则 RP 为 0.

在学习阶段，利用获得的回报 $r(s,a)$ 来更新 Q 值的过程如式(7)所示：

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a')) \tag{7}$$

式中：$\alpha(0 \leq \alpha \leq 1)$ 是一个常数步长参数，用来控制已有经验和当前所获得经验之间的权重，其中，$(1-\alpha)Q(s,a)$ 为已有经验，$\alpha(R(s,a) + \gamma \max_{a'} Q(s',a'))$ 目前回报和未来经验折扣估计值的和，即当前所获得的经验；$\gamma(0 \leq \gamma \leq 1)$ 是折扣因子，用来控制未来动作选择的权重. 更新后的 Q 值将会在下一次决策中被使用，使 Agent 能够选出最优的自适应动作.

6 实例分析

本文以一个景点语音导游系统(地听)为例，该系统分为服务器端和移动端两部分，其中服务器端是一套 SBS 系统，共由 7 个组件服务构成，分别是：景区定位(c_1)、上传图像解析(c_2)、图像特征匹配(c_3)、GPS 定位识别景点(c_4)、当前景点语音播放列表确定(c_5)、当前景点语音播放传送(c_6)和后续路线推荐(c_7)，整体的业务流程如图 4 所示. 在本文中，将重点关注地听系统服务器端的自适应性能优化过程.

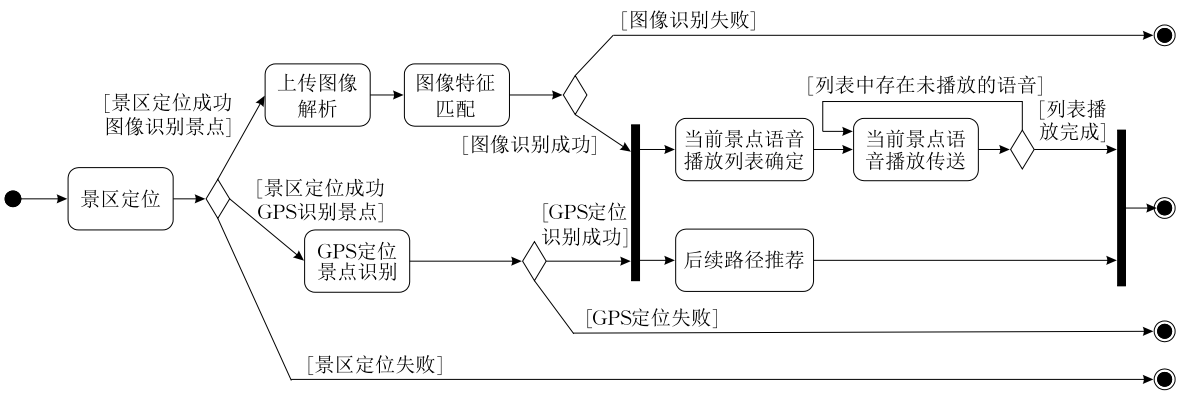


图 4 地听服务器端业务流程

6.1　系统部署信息

系统部署在由 6 台物理机(Physical Machine，PM)组成的云环境中，其中 3 台物理机为配置相同的服务器，标记为 PM_1～PM_3，并在这 3 台物理机上共创建 7 台配置相同的虚拟机，分别标记为 VM_1～VM_7(在 PM_1 上创建 VM_1 和 VM_2，在 PM_2 上创建 VM_3～VM_5，在 PM_3 上创建 VM_6 和 VM_7)，然后在每台虚拟机上分别部署 SBS 系统的 1 个组件服务，如在 VM_1 上部署 c_1. 另外 3 台物理机则分别用来进行自适应决策、存储 SBS 系统的业务数据及运行状态的监控数据和产生访问负载及监测 SBS 系统的运行状态. 以 PM_1，VM_1 和 c_1 为例，相关的系统部署信息如下所示：

Physical Machine Specification PM_1

　Deployment：(VM_1，VM_2)

　Resource：((CPU，2.5 GHz×2×4)，(MEM，8 GB))

Virtual Machine Specification VM_1

　Deployment：(c_1)

　Resource：((CPU，1×2)，(MEM，2 GB))

Component Specification c_1

　Deployment：VM_1，

　Status：[USING]

6.2　自适应优化目标设定和触发事件生成

在地听系统自适应优化的过程中，自适应优化的目标是要保证系统的响应时间不能超过 SLA 中规定的阈值. 而与该目标相关的指标，除了系统的响应时间外，还有用户的并发访问量以及虚拟机的资源利用率，如 CPU 和内存的占用率. 对于用户并发访问量，当系统当前的用户并发访问量超过系统所能承受的上限时，则很容易导致系统的响应时间变长，所以需要设定用户并发量相关的自适应目标，以保证在出现并发用户数超过系统当前承载能力的时候可以进行及时的调优. 对于虚拟机的资源利用率，若虚拟机的资源利用率一段时间内都处于较高的状态时，如 CPU 利用率大于 80%，则说明部署在虚拟机上的组件服务已经处于较高的负载状态，很有可能会出现或已经出现过载的情况，从而导致服务响应时间违反 SLA 规定，所以也需要设定虚拟机资源利用率的自适应目标，使虚拟机可以保持在适当的负载状态.

根据上述因素，本文将设定 3 类自适应目标：响应时间类、用户并发量类和虚拟机资源利用率类. 其中，响应时间类的自适应目标可以根据 SLA 中关于系统响应时间的约束进行设定，以生成系统级的自适应目标. 以系统的响应时间自适应目标为例：

Goal sys_resp_time_goal：resp_time **Between** 7 s 19 s

用户并发量类的自适应目标可以分为系统级和组件服务级两类，每类的设定方式也与响应时间类的自适应目标类似. 以系统的用户并发量自适应目标和组件服务 c_1 的用户并发量自适应目标为例：

Goal sys_user_con_goal：user_con **Below** 300

Goal $comp\_user\_con\_goal_1$：c_1.user_con **Below** 300

对于虚拟机资源利用率的自适应目标，其设定完全是根据专家经验，例如服务器的 CPU 利用率应保持在 20%～80%之间. 此外，本文中主要考虑 CPU 和内存指标，结合系统部署信息，可以为每一个虚拟机设定相应的自适应目标，即人工设定每类资源利用率的具体范围，所有虚拟机的资源利用率类的自适应目标都以该范围为标准进行设定. 以虚拟机 VM_1 为例，其资源利用率类的自适应目标如下：

Goal $vm\_cpu\_usage\_goal_1$：vm_1.cpu_usage

　Between 20% 80%

Goal $vm\_mem\_usage\_goal_1$：vm_1.mem_usage

　Between 20% 80%

根据上述的自适应目标，按照 3.3 节所述的自适应触发事件抽取规则，可以生成相应的触发事件. 以系统响应时间的自适应目标为例，抽取的自适应触发事件如下：

Event $sys\_resp\_time\_goal.event_1$：resp_time＜7 s

　Target DEGR_PERF

Event $sys\_resp\_time\_goal.event_2$：resp_time＞19 s

　Target IMPR_PERF

当运行阶段所获得监测指标使得多个触发事件被触发时，将按照响应时间类、用户并发量类和虚拟机资源利用率的优先级进行触发. 若还存在系统级和组件服务级的触发事件同时被触发，则系统级的自适应目标相关的触发事件将优先触发，例如：当通过监测发现系统的用户并发量和组件服务 c_1 的用户并发量都不符合要求时，则优先触发系统级的触发事件(resp_user_con_goal. event)，首先保证系统的整体性能. 此外，为了避免由于 KPI 的瞬时变化所导致的不必要的优化，本文将根据一段时间内 KPI 值的变化情况对触发事件进行触发，例如在 30 s 内，如果有 80% 以上的系统响应时间监测值超过了 19 s，则触发 $resp\_time\_goal_1.event_2$ 事件，开始对系统进行优化.

6.3 State 和 Action

根据 3.4 节中 State 的形式定义和所设定的虚拟机资源类型以及负载类型，可以生成环境状态的集合. 假设所有虚拟机的初始资源类型为 MICRO，负载为 TYPE_I 时，环境状态为

State s_1

Resource：(MICRO, MICRO, MICRO, MICRO, MICRO, MICRO, MICRO)

Load：TYPE_I

对于 Action，以增加和减少 VM_1 资源的自适应动作为例：

Action a_1

Primitive：addRes(VM_1, MICRO, SMALL)

Target：[IMPR_PERF]

Action a_2

Primitive：reduceRes(VM_1, MICRO, NANO)

Target：[DEGR_PERF]

其中：自适应动作 a_1 是将 VM_1 的资源从 MICRO 类型增加至 SMALL 类型，由于增加资源，所以可以提升系统的性能，即 Target 为 IMPR_PERF；而自适应动作 a_2 是将 VM_1 的资源从 MICRO 类型减少至 NANO 类型，降低系统的性能，Target 为 DEGR_PERF. 其它 State 与 Action 与上述例子类似.

7 实 验

本节将通过实验分析不同参数对本文提出的方法的性能的影响，以及应用于部署在云环境中的 SBS 系统时的有效性，主要包括实验环境配置、实验设计和实验结果与分析三部分内容.

7.1 实验环境配置

实验所用的云环境与第 6 节中的系统部署环境一致，共包括 6 台物理机(Physical Machine，PM)，各物理机通过带宽 1GB/s 的局域网相连，其中物理机 1～3 为硬件配置相同的服务器(CPU：Intel Xeon E5-2620 v2 24 核，64 GB 内存，1000 Mbps 网卡，1 TB 磁盘)，并在这些服务器上创建大小相同的虚拟机(类型为 MICRO)，用于部署 SBS 系统的组件服务. 物理机 4 用于分析 SBS 系统的监控数据，并根据系统的运行状态进行自适应决策，生成相应的自适应调整方案. 物理机 5 用于存储 SBS 系统的业务数据及运行状态的监控数据. 物理机 6 用于监测 SBS 系统的运行状态并产生访问负载. 具体部署结构如图 5 所示.

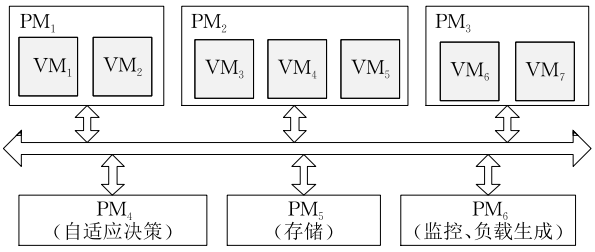


图 5 云环境部署结构

7.2 实验设计

本文的实验主要是为了分析不同参数对基于 Q-learning 的自适应性能优化方法性能的影响以及在 SBS 自适应性能优化中应用的有效性，具体如下：

(1) 分析不同因素对算法收敛性的影响.

① 分析参数设定对算法收敛性的影响. 对 SBS 系统施加恒定的压力，对比在不同参数(α, γ, τ)情况下基于 Q-learning 方法的回报累积情况，以选出较优的参数；

② 分析引导对算法收敛性的影响. 对 SBS 系统施加恒定的压力，对比方法在是否采用引导的情况下的回报累积情况.

(2) 验证基于 Q-learning 的自适应性能优化方法的有效性：

① 在不同压力的场景下，检验本文提出的方法能否使 SBS 系统的响应时间恢复到 SLA 的约束范围内(7 s～19 s)，并对比同样情况下不使用自适应优化方法时 SBS 系统的响应时间状态；

② 在不同压力的场景下，对比本文提出的方法和另外两种动态决策方法(分别为文献[8]中利用应用系统性能模型的方法；文献[12]中基于遗传算法的方法)作用下的系统响应时间的情况以及使用的资源成本 c_o.

对于实验内容 2 的压力场景，本文根据数理统计[17]中对时间序列(用户访问序列可以视为一个时间序列)的分类来设计负载场景. 数理统计中将时间序列分为平稳型和非平稳型，其中非平稳型又可以根据序列变化的趋势分为随机型和具有一定变化规律的类型(如具有线性递增特征的类型、具有线性递减特征的类型和具有季节特征的类型).

对于非平稳随机型场景，本文在 0～60 min 的时间内随机生成 100～200 区间的任意并发数，且持续很短的时间，具体如图 6 所示.

在随机型负载场景中，本文对比了使用本文的方法和不使用时 SBS 系统的响应时间状态，具体如图 7 所示.

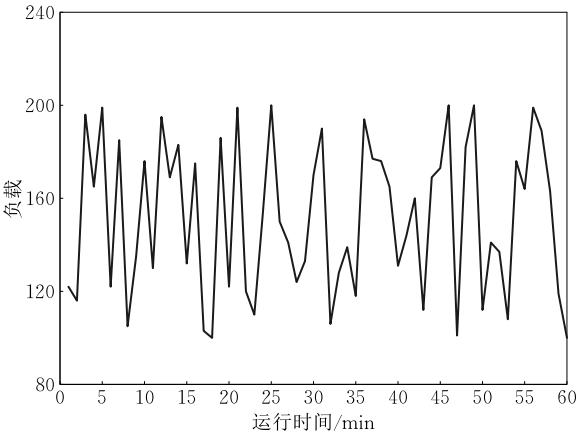


图 6　随机型负载场景

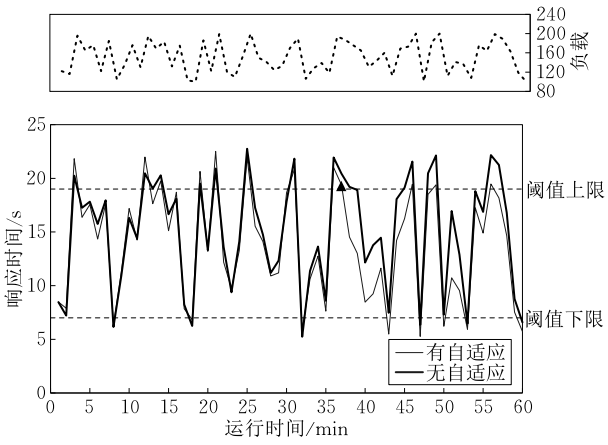


图 7　随机型场景中 SBS 的响应时间对比

在图 7 中，上方的虚线为负载随时间变化的情况，下方为 SBS 的响应时间的变化情况，其中标注“▲”的位置为该时刻进行了资源调整. 由于本文提出的自适应方法为了避免由于 KPI 瞬间变化所导致的不必要的优化，所以在随机型场景中很多时刻并未触发自适应优化. 在不触发自适应优化的情况下，系统的响应时间(Response Time，RT)会随着负载的随机变化而变化，具体分布情况如表 4 所示.

表 4　随机型场景中 SBS 的响应时间分布情况(单位：%)

	$RT>19$ s	$19\text{ s}\geq RT\geq 7$ s	$RT<7$ s
有自适应	18.3	68.3	13.4
无自适应	26.7	63.3	10.0

如表 4 所示，在随机型场景中，有无自适应优化的情况下，系统响应时间符合 SLA 要求的比例($7\leq RT\leq 19$)分别为 68.3% 和 63.3%，满足用户要求($RT\leq 19$)的比例分别为 81.7% 和 73.3%. 采用自适应优化使 SBS 系统响应时间符合 SLA 约束和用户要求的比例分别提升了 5% 和 8.4%. 可见，本文提出的自适应优化方法在随机型负载场景中对 SBS 系统的性能改善不大.

所以，根据随机型场景的实验结果，本文将重点验证所提出的方法在其它负载场景中的有效性，分别是：具有平稳型特征的负载场景(简称平稳型场景)、具有线性递增非平稳型特征的负载场景(简称线性递增型场景)、具有线性递减非平稳型特征的负载场景(简称线性递减型场景)和具有季节性非平稳型特征的负载场景(简称季节型场景). 其中，平稳型场景包括 4 种负载方式，分别标记为负载 I、负载 II、负载 III 和负载 IV. 在虚拟机资源为初始分配量的情况下，负载 I 将使得 SBS 系统的响应时间低于 SLA 约束的下限阈值；负载 II 将使得 SBS 系统的响应时间保持在 SLA 的约束范围之内；负载 III 将使得 SBS 系统的响应时间超过 SLA 约束的上限阈值；负载 IV 将使得 SBS 系统的响应时间比在负载 III 情况下更多地超过 SLA 约束的上限阈值. 具体并发数量见表 5.

表 5　实验负载

负载 I	负载 II	负载 III	负载 IV
100	150	180	200

对于平稳型场景，将按照 II→III→IV→I 的顺序对 SBS 系统进行施压，且一段时间内一直持续施加一种负载；线性递增型场景是在 0～60 min 期间内，用户并发量从负载 I 增长到负载 IV，总体处于线性递增的趋势. 线性递减型则与之相反，总体处于线性递减的趋势；季节型场景是用户并发量具有季节波动性，一个周期为 30 min. 在一个周期内，两种类型的用户并发量的最小值为负载 I，最大值为负载 IV，具体如图 8 所示.

对于使用的资源成本 c_o，具体如式(8)所示.

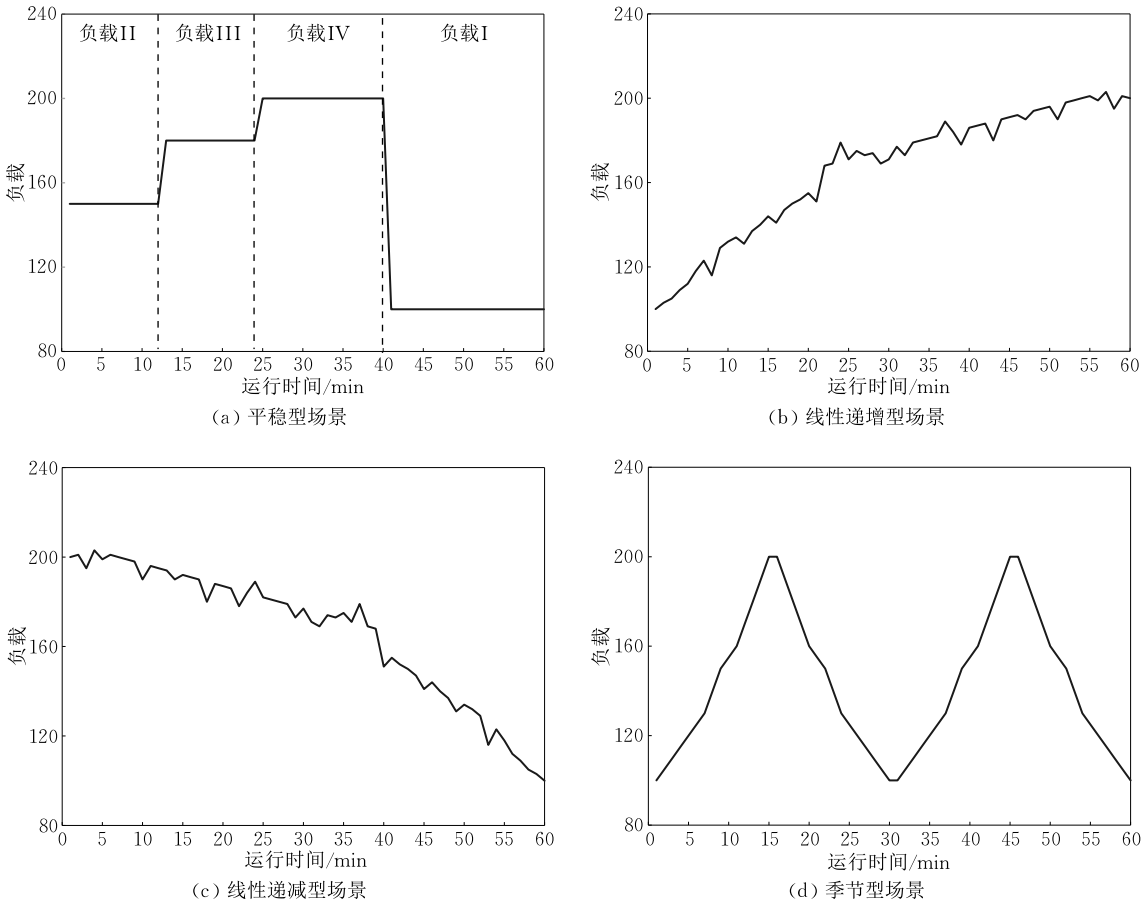
$$c_o=\sum_{l=1}^{i}\sum_{\alpha=1}^{n}(RV_L^{\alpha}\times U_r\times T) \tag{8}$$

其中：$RV_l^{\alpha}(1\leq\alpha\leq n,1\leq k\leq i)$为第 l 个虚拟机所占用的资源量；U_r 表示每种资源的单位价格；T 为使用该虚拟机资源的时间.

7.3　实验结果与分析

7.3.1　收敛性分析

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 Q-learning 算法收敛速度的主要参数(学习步长参数 α、折扣因子 γ 和选择参数 τ)在取不同值时以及是否对算法进行引导时，算法的回报累积情况.



(a) 平稳型场景 (b) 线性递增型场景

(c) 线性递减型场景 (d) 季节型场景

图 8 4 种负载场景

(1) 参数设定对算法收敛性的影响

本实验对于参数的设定将分为 6 种情况,具体如表 6 所示.

表 6 参数设定

	α	γ	τ
设定 I	0.3	0.7	0.8
设定 II	0.3	0.7	10.0
设定 III	0.3	0.7	50.0
设定 IV	0.5	0.8	0.8
设定 V	0.5	0.8	10.0
设定 VI	0.5	0.8	50.0

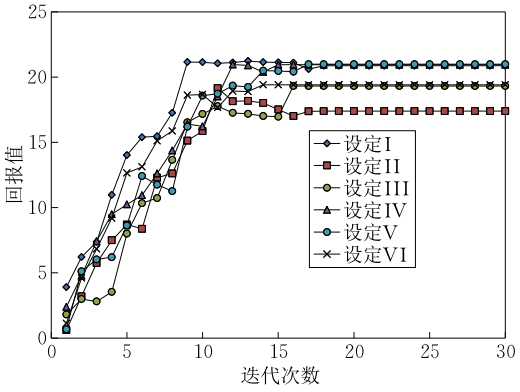


图 9 不同参数设定时的回报累积

在实验中,将对 SBS 系统施加恒定的压力(用户并发量为 200),SBS 系统的响应时间将会超过 SLA 的约束,使得自适应优化被触发,在迭代 30 次后累积回报值的具体情况如图 9 所示.

从图 9 中可以看出,参数设定为(0.3,0.7,0.8)时算法的回报值累积曲线在其它参数设定曲线的上方,则认为该参数设定使得算法具有较好的收敛性,所以本文将采用设定 I.

(2) 引导对算法收敛性的影响

本实验主要是在恒定负载的情况下,所有虚拟机的初始资源类型为 SMALL 时,对比有无引导时基于 Q-learning 决策算法的回报累积情况. 其中,无引导是指在选择 Action 时,从当前 State 对应的所有action中进行选择;有引导是指在选择 Action 时,从与被触发 Event 有相同 Target 的 Action 集合中进行选择. 在迭代 30 次后回报的具体情况如图 10 所示.

从图 10 中可以看出,有引导时的算法的回报值累积曲线在没有引导的回报值累积曲线上方,则说明有引导时算法具有更好的收敛性.

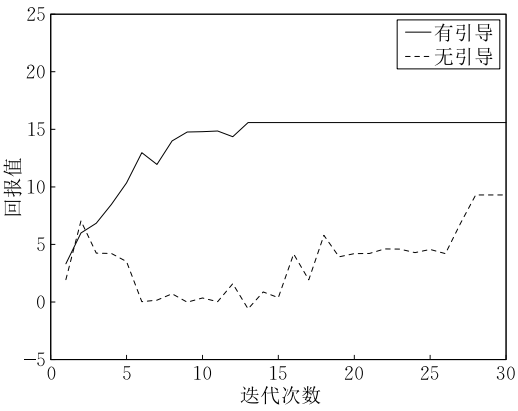


图 10　有无引导时的回报累积

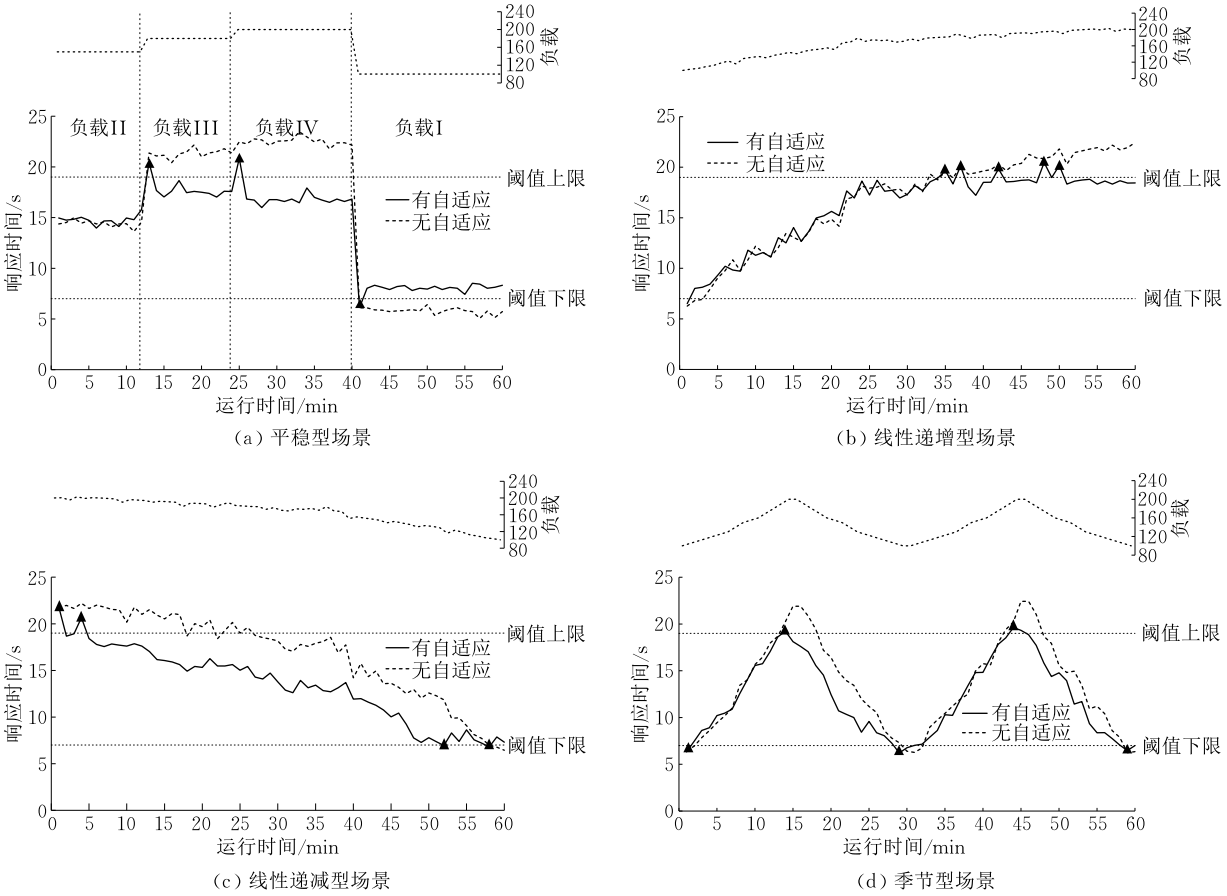
7.3.2　有效性验证

为了验证本文提出的自适应优化方法的有效性，本文将对比 4 种压力场景中，有自适应优化和无自适应优化情况下 SBS 系统的响应时间 RT，以及对比本文提出的方法和另外两种动态决策方法作用下的系统响应时间的情况以及使用的资源成本，具体如下：

(1) 有无自适应情况对比

在 4 种压力场景中，SBS 系统在有自适应优化和无自适应优化情况下的系统响应时间情况如图 11 所示. 与图 7 类似，在图 11(a)～(d)中，上方的虚线为负载随时间变化的情况，下方为 SBS 的响应时间的变化情况，其中标注“▲”的位置为该时刻进行了资源调整.

从图 11(a)中可以看出，在平稳型的负载场景下，当 SBS 系统的并发访问量从负载 II 增加到负载 III 后，系统的响应时间也随之增加，并超过了 SLA 约束的上限阈值. 在对 SBS 系统进行优化后，其响应时间在第 14 min 时恢复到 SLA 的约束范围内. 当 SBS 系统的并发访问量从负载 III 增加到负载 IV 时，以及并发访问量从负载 IV 减少到负载 I 后，对系统进行自适应优化后均可以使系统的响应时间恢复到 SLA 的约束范围内. 而未进行自适应优化时，在施加负载 III、负载 IV 和负载 I 期间，系统响应时间一直处于高于上限阈值和低于下限阈值的状态. 与在平稳型场景中的情况类似，SBS 系统的响应时间在其它 3 种场景中的情况如图 11(b)、(c)和(d)所示，当有自适应优化发挥作用时，SBS 系统的响应时间会很快地恢复到 SLA 的约束范围内，而没有自适应优化时，SBS 系统的响应时间会长时间地处于 SLA 违例状态. 每种场景中响应时间的具体分



(a) 平稳型场景　(b) 线性递增型场景

(c) 线性递减型场景　(d) 季节型场景

图 11　有无自适应时 SBS 的响应时间对比

布情况如表 7 所示.

如表 7 所示,在不同的负载场景中,采用自适应优化时,SBS 系统响应时间符合 SLA 约束($7 \leq RT \leq 19$)的比例分别提升了 75%、38.3%、40%和 15%,符合 SBS 系统使用用户要求($RT \leq 19$)的比例分别提升了 43.4%、35%、38.4 和 10%.可见,本文提出的方法是有效的.

表 7 有无自适应时 SBS 响应时间分布情况(单位:%)

		$RT>19$ s	19 s$\geq RT \geq$7 s	$RT<7$ s
平稳型	有自适应	3.3	95.0	1.7
	无自适应	46.7	20.0	33.3
线性递增型	有自适应	8.3	90.0	1.7
	无自适应	43.3	51.7	5.0
线性递减型	有自适应	3.3	93.3	3.4
	无自适应	41.7	53.3	5.0
季节型	有自适应	5.0	88.3	6.7
	无自适应	15.0	73.3	11.7

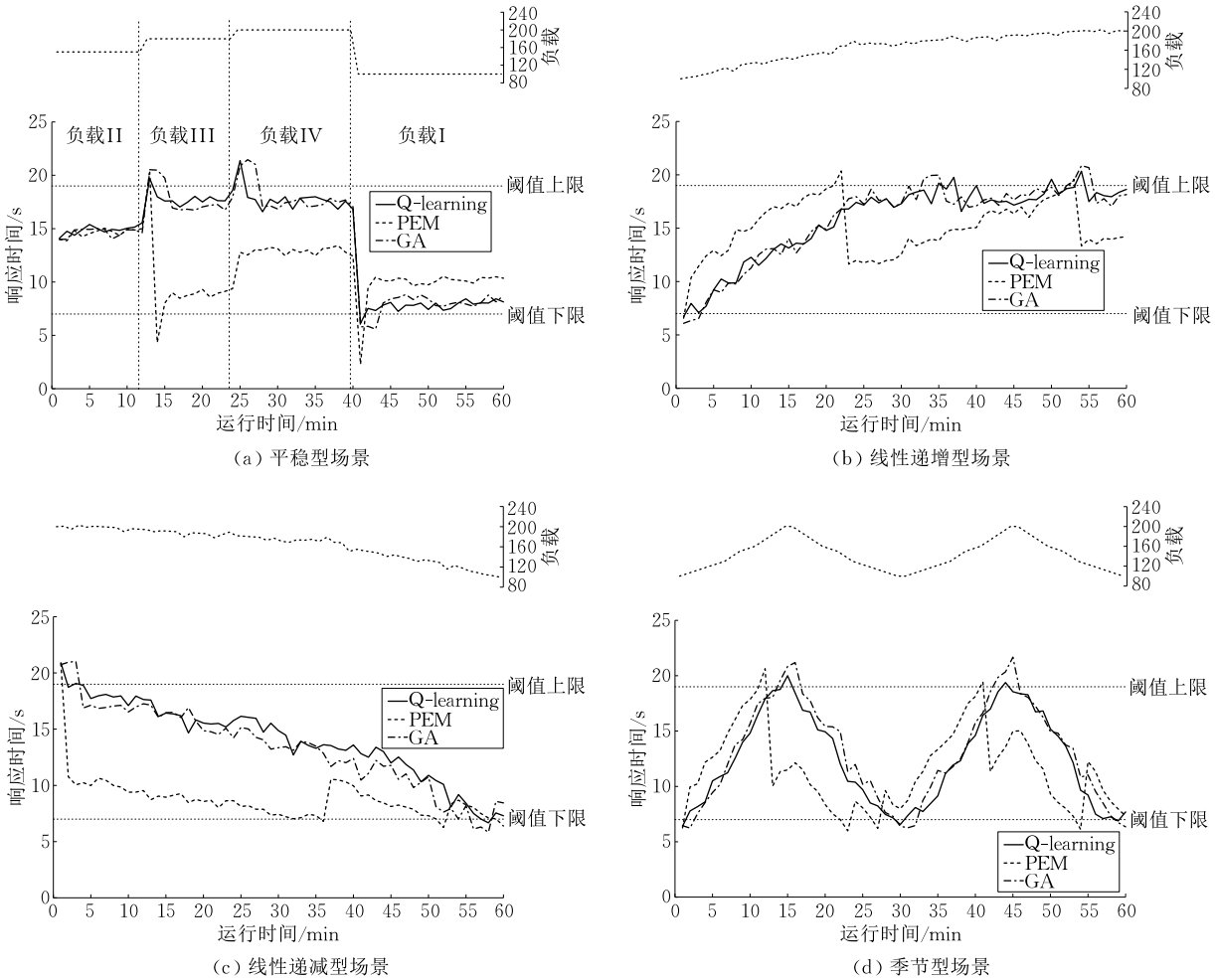
(2)不同决策方法对比

在本小节实验中,将在不同的压力场景下,对比本文提出的方法和另外两种动态决策方法(分别为文献[8]中利用应用系统性能模型的方法,简称 PEM;文献[12]中基于遗传算法的方法,简称 GA)作用下的系统响应时间的情况以及使用的资源成本 c_o,具体见图 12、表 8 和表 9.

表 8 不同方法作用下 SBS 响应时间分布情况(单位:%)

		$RT>19$ s	19 s$\geq RT \geq$7 s	$RT<7$ s
平稳型	Q-learning	3.3	95.0	1.7
	PEM	1.7	95.0	3.3
	GA	10.0	85.0	5.0
线性递增型	Q-learning	6.7	91.7	1.6
	PEM	3.3	95.0	1.7
	GA	10.0	85.0	5.0
线性递减型	Q-learning	3.3	95.0	1.7
	PEM	1.7	93.3	5.0
	GA	5.0	90.0	5.0
季节型	Q-learning	3.3	91.7	5.0
	PEM	3.3	90.0	6.7
	GA	10.0	78.3	11.7

从图 12 可以看出,在 4 种不同的压力场景中,本文提出方法和其它两种方法均可以使 SBS 系统的响应时间满足 SLA 的约束.其中,本文提出的方



(a)平稳型场景 (b)线性递增型场景

(c)线性递减型场景 (d)季节型场景

图 12 不同方法作用下 SBS 的响应时间对比

法和基于 PEM 的方法都可以很快地使系统响应时间恢复到 SLA 的约束范围内，如表 8 所示，两种方法使得 SBS 系统响应时间符合 SLA 约束和 SBS 系统使用用户要求的比例相差不大. 但基于 PEM 的方法由于离线建立的性能模型不能很好地适应环境的动态变化，每次调整后系统的响应时间变化幅度都较大，而这也代表着使用了更多的资源. 如表 9 所示，在 4 种场景中，基于 PEM 的方法的资源成本 c_o 比本文方法分别高出 57.74%、55.22%、69.64%和 83.35%. 可以看出，本文提出的方法对比基于 PEM 的方法在决策效率方面相差不大，但使用的资源成本更低.

表 9　不同方法的资源成本

	Q-learning	PEM	GA
平稳型	3597	5674	3368
线性递增型	2617	4062	2482
线性递减型	4546	7712	4340
季节型	2949	5907	2781

综上，本文提出的方法在 4 种典型负载场景中均可以达到优化系统响应时间的目的，且在拥有较高决策效率的同时，可以保证云应用占用较少的资源即可以满足性能需求. 综合考虑效率和资源成本时，本文提出的方法优于其它两种方法.

8　相关研究

自适应决策是自适应过程中的关键步骤，可以通过多种方式实现，一种方式是依据预先设定的自适应策略进行决策，还可以利用人工智能、控制论等技术进行决策.

(1) 基于策略的自适应决策方面

自适应策略的描述通常采用 ECA (Event-Condition-Action) 的形式[18]，或者采用更为简单的 if-else 形式，连同系统的因果网络一起构建基于模型的推理机制[19]. 此外，IBM 还提出了 ACPL (Automatic Computing Policy Language) 语言，用来描述自主计算中的策略[20]. 在此基础上，文献[6]的自适应决策机制包括离线和在线两个阶段，在离线阶段生成一组 ECA 形式的自适应规则，在线阶段时根据当前的系统状态和策略中所定义的目标对自适应规则集合进行匹配的方式生成自适应调整方案. 文献[9]提出了一种 SBS 运行时自适应框架 MOSES (MOdel-based SElf-adaptation of SOA Systems)，利用为每个抽象服务建立的最优自适应策略模型选取动作模式，从而达到最佳性能. 文献[10]提出了一种可执行的建模语言 EUREMA (ExecUtable RuntimE MegAmodels)，可以按照模型驱动的工程方法来简化自适应工程的开发. 文献[11]给出了一组特定的 MAPE-K 模板 (Monitor-Analyze-Plan-Execute plus Knowledge)，这些模板包括特定行为模板和特定属性模板，前者用来对 MAPE-K 循环回路中的不同的组件进行建模，后者用来验证自适应行为的正确性，并利用这些模板构建了一个自适应系统. 文献[8]则通过建立应用系统的负载、资源和性能之间的关系模型，来制定应用系统的优化策略. 但是这类方法大多需要对系统或者环境进行建模，而在离线阶段建立的模型不能很好地适应发生变化后的环境或应用系统.

(2) 结合人工智能技术进行自适应决策方面

智能优化算法在自适应决策中得到了较多的应用，例如文献[13]在决策时考虑了工作流使用资源的成本和所花费时间之间的平衡，采用基于启发式的多完整关键路径方法进行求解；文献[14]通过提出的声明式服务编排语言 DSOL (Declarative Service Orchestration Language) 将自适应调整方案的生成问题表示为自动规划问题，采用智能规划技术进行求解；文献[12]在自适应决策过程中同时考虑了违反 SLA 约束的代价及执行自适应调整动作的代价，将自适应调整方案的生成问题建模为单目标优化问题，通过智能算法对该问题进行求解. 文献[15]提出了一种基于服务选取的 SBS 资源分配模型，并采用支持向量回归 (Support Vector Regression，SVR) 构建组件服务的性能模型，并以性能模型为基础进行分配方案的求解. 但利用智能优化算法求解通常都会存在计算效率问题，需要进行相应的优化.

还有学者利用规划、模糊推理和基于用例的推理 (Case-Based Reasoning，CBR) 方法来进行自适应决策，例如文献[21]将特征模型和在线机器学习结合起来，即将领域专家的知识体现在特征模型中，然后通过在线学习来改进自适应决策的准确性和效率；文献[22]构建了一种自调整的模糊控制器 (Self-Tuning Fuzzy Controller，STFC) 方法，STFC 利用模糊规则推理得到资源的调整量；文献[23]通过计算监测到的系统状态和用例库中用例的相似度，找到最相似的用例，并执行用例中的自适应动作来对系统进行优化，然后将优化的结果更新至用例库中. 而这类方法优化效果的好坏很大程度取决于人工设定知识 (如规则、用例库等) 的质量.

与本文的方法类似的还有文献[24-26],这些方法都采用了基于强化学习的方法来调整系统的资源,达到保障应用系统性能的目的.其中,文献[24]所调整的资源是以虚拟机为粒度,即增加、减少虚拟机的个数,而文献[25-26]是通过调整单个虚拟机的资源来增加其处理能力.但是这些方法都是将优化的对象视为一个整体,当这些方法应用于由多个组件服务组成的 SBS 时,由于环境状态和自适应动作的增加,可能会出现决策效率较低的情况.而本文通过环境状态和引导算子对自适应动作集合进行筛选,有效地缩小候选自适应动作的范围,保证自适应决策的高效性.在针对 SBS 的自适应资源分配方面,文献[27]采用了基于反馈控制的方法,并利用径向基函数(Radial Basis Function, RBF)神经网络在应用系统运行时自适应的调整控制参数;文献[28]首先建立每个组件服务的 RAT(Resource-Allocation-Throughput)模型,并将模型扩展至整个 SBS,从而分析 SBS 所承受的负载和分配的资源之间的关系,然后将资源分配问题转化为线性规划的优化问题并进行求解.但在这两个方法中,都需要在离线阶段建立与 SBS 相关的模型,当 SBS 的组件服务发生动态变化时,离线模型可能需要重新训练,而采用 Q-learning 这种无模型的方法则可以避免此类问题.

所以,针对现有的方法中存在的一些问题,本文提出了一种基于强化学习的云应用自适应方法,而强化学习可以较好的满足自适应决策过程对于动态性、高效性等方面的要求[29].本文采用了 Q-learning 算法,该算法是一种无模型(model-free)的在线学习算法,在自适应的过程中不需要建立应用系统的性能模型,可以更好地应对自适应过程中的动态变化.此外,本文还提出了一种引导算子,可以有效地缩小候选自适应动作的范围,从而保障算法的学习效率.

9 结 论

本文提出了一种基于强化学习的 SBS 云应用自适应优化方法,建立了自适应基本要素之间相互关系的特征描述框架,并详细描述了如何在决策的过程中使用经验和如何累积自适应调整之后的经验.通过实验表明,本文提出的方法是有效的,并具有以下优点:(1)采用了事件触发机制,可以避免云应用系统的性能符合 SLA 要求后进行“无效”的调整;(2)采用了无模型的在线学习算法(Q-learning 算法),可以更好地应对自适应过程中的动态变化;(3)提出了一种引导算子,利用被触发的 Event 对所有可执行的 Action 进行筛选,限定其在特定范围的 Action 集合中进行选择,从而提升学习算法的效率.

参 考 文 献

[1] Chun B-G, Ihm S, Maniatis P, et al. CloneCloud: Elastic execution between mobile device and cloud//Proceedings of the 6th Conference on Computer Systems. New York, USA, 2011: 301-314

[2] Sun Da-Wei, Chang Gui-Ran, Chen Dong, et al. Profiling, quantifying, modeling and evaluating green service level objectives in cloud computing environments. Chinese Journal of Computers, 2013, 36(7): 1509-1525(in Chinese)
(孙大为,常桂然,陈东等. 云计算环境中绿色服务级目标的分析、量化、建模及评价. 计算机学报, 2013, 36(7): 1509-1525)

[3] Wood T, Shenoy P, Venkataramani A, et al. Sandpiper: Black-box and gray-box resource management for virtual machines. Computer Networks, 2009, 53(17): 2923-2938

[4] Calinescu R, Grunske L, Kwiatkowska M, et al. Dynamic QoS management and optimization in service-based systems. IEEE Transactions on Software Engineering, 2011, 37(3): 387-409

[5] Zhu Jie-Ming, He Pin-Jia, Zheng Zi-Bin, et al. Towards online, accurate, and scalable QoS prediction for runtime service adaptation//Proceedings of the IEEE International Conference on Distributed Computing Systems. Madrid, Spain. 2014: 318-327

[6] Rosa L, Rodrigues L, Lopes A, et al. Self-management of adaptable component-based applications. IEEE Transactions on Software Engineering, 2013, 39(3): 403-421

[7] Song Ying, Sun Yu-Zhong, Shi Wei-Song. A two-tiered on-demand resource allocation mechanism for VM-based data centers. IEEE Transactions on Services Computing, 2013, 6(1): 116-129

[8] Mega C, Waizenegger T, Lebutsch D, et al. Dynamic cloud service topology adaption for minimizing resources while meeting performance goals. IBM Journal of Research and Development, 2014, 58(2): 1-10

[9] Cardellini V, Casalicchio E, Grassi V, et al. Moses: A framework for QoS driven runtime adaptation of service-oriented systems. IEEE Transactions on Software Engineering, 2012, 38(5): 1138-1159

[10] Vogel T, Giese H. Model-driven engineering of self-adaptive software with EUREMA. ACM Transactions on Autonomous and Adaptive Systems, 2014, 8(4): 1-18

[11] De La Iglesia D G, Weyns D. MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. ACM

Transactions on Autonomous and Adaptive Systems, 2015, 10(3): 1-31

[12] Leitner P, Hummer W, Dustdar S. Cost-based optimization of service compositions. IEEE Transactions on Services Computing, 2013, 6(2): 239-251

[13] Cai Zhi-Cheng, Li Xiao-Ping, Gupta J N. Heuristics for provisioning services to workflows in XaaS clouds. IEEE Transactions on Services Computing, 2014, 9(2): 250-263

[14] Cugola G, Ghezzi C, Pinto L S. DSOL: A declarative approach to self-adaptive service orchestrations. Computing, 2012, 94(7): 579-617

[15] Zhao Xiu-Tao, Zhang Bin, Zhang Chang-Sheng. Service selection based resource allocation for SBS in cloud enviroments. Journal of Software, 2015, 26(4): 867-885(in Chinese)
(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)

[16] Sutton R S, Barto A G. Reinforcement learning: An introduction. Massachusetts: MIT Press, 1998

[17] Chen Pin. Mathematical Statistics. Beijing: China Machine Press, 2008(in Chinese)
(陈平. 应用数理统计. 北京：机械工业出版社，2008)

[18] Lobo J, Bhatia R, Naqvi S. Apolicy description language// Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Conference on Innovative Applications of Artificial Intelligence. Menlo Park, USA, 1999: 291-298

[19] Wang Gui-Jun, Wang Chang-Zhou, Chen A, et al. Service level management using QoS monitoring, diagnostics, and adaptation for networked enterprise systems//Proceedings of the 9th IEEE International EDOC Enterprise Computing Conference. Enschede, Netherlands, 2005: 239-248

[20] Agrawal D, Lee K-W, Lobo J. Policy-based management of networked computing systems. IEEE Communications Magazine, 2005, 43(10): 69-75

[21] Esfahani N, Elkhodary A, Malek S. A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE Transactions on Software Engineering, 2013, 39(11): 1467-1493

[22] Rao Jia, Wei Yu-Di, Gong Jia-Yu, et al. Qos guarantees and service differentiation for dynamic cloud applications. IEEE Transactions on Network and Service Management, 2013, 10(1): 43-55

[23] Maurer M, Brandic I, Sakellariou R. Adaptive resource configuration for cloud infrastructure management. Future Generation Computer Systems, 2013, 29(2): 472-487

[24] Liu Jin-Zhao, Zhang Yao-Xue, Zhou Yue-Zhi, et al. Aggressive resource provisioning for ensuring QoS in virtualized environments. IEEE Transactions on Cloud Computing, 2015, 3(2): 119-131

[25] Rao Jia, Bu Xiang-Ping, Xu Cheng-Zhong, et al. A distributed self-learning approach for elastic provisioning of virtualized cloud resources//Proceedings of the IEEE 19th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Singapore, 2011: 45-54

[26] Nathuji R, Kansal A, Ghaffarkhah A. Q-clouds: Managing performance interference effects for QoS-aware clouds// Proceedings of the 5th European conference on Computer systems. Paris, France, 2010: 237-250

[27] Gong Si-Qian, Yin Bei-Bei, Zhu Wen-Long, et al. An adaptive control strategy for resource allocation of service-based systems in cloud environment//Proceedings of the IEEE International Conference on Software Quality, Reliability and Security. Vancouver, Canada, 2015: 32-39

[28] Yau S S, An H G. Adaptive resource allocation for service-based systems//Proceedings of the 1st Asia-Pacific Symposium on Internetware. Beijing, China, 2009: 483-499

[29] Ding Bo, Wang Huai-Min, Shi Dian-Xi. Constructing software with self-adaptability. Journal of Software, 2013, 24(9): 1981-2000(in Chinese)
(丁博，王怀民，史殿习. 构造具备自适应能力的软件. 软件学报，2013, 24(9): 1981-2000)

YAN Yong-Ming, born in 1981, Ph. D. candidate. His current research interest is cloud service performance optimization.

ZHANG Bin, born in 1964, Ph. D., professor, Ph. D. supervisor. His research interests include service computing and cloud computing.

GUO Jun, born in 1974, Ph. D., associate professor. His research interests include service computing and cloud computing.

MENG Yu, born in 1990, Ph. D. candidate. His current research interests include cloud computing, service prediction and optimization.

Background

In cloud, the characteristics of utility computing and pay-for-use model need the service system not only can satisfy the dynamic resource allocation which the application requires at the minimum costs but also can adjust the resources adaptively when it deviates from the desired behavior to continuously offer the services that user expect. There are

some problems of the traditional decision method for SBS (Service-based System), such as the decision method based on application system performance model cannot adapt to the dynamic change of SBS cloud environment, the efficiency of the decision making method based on intelligent optimization algorithm is not good.

This paper proposes a self-adaptation performance optimization approach for SBS cloud application based on reinforcement learning, and established the feature description framework of the relationship of the basic key elements of the adaptation. The method adopts a model-free online learning algorithm, through repeating the "execution-accumulation-learning-decision" process, the method optimize the decision results continuously, so as to cope with dynamic change of the cloud environment and SBS. This paper puts forward a shaping operator which can effectively narrow the scope of the candidate adaptive action and improve the learning efficiency of the algorithm. This paper implements a prototype framework with a SBS as an example, and proves the validity of the proposed method in 4 different scenarios.

This work was supported by the National Natural Science Foundation Program of China (61572117, 61300019, 61370155), the Provincial Scientific and Technological Project (2015302002), and the Special Fund for Fundamental Research of Central Universities of Northeastern University (N140406002, N150404008).