基于 SaaS 的弹性云平台优化调度策略设计*

赵少卡[1,2], 李立耀[1], 徐 聪[2], 杨家海[2]

(1. 福建师范大学福清分校 数学与计算机科学系, 福建 福清 350300; 2. 清华大学 网络科学与网络空间研究院, 北京 100084)

摘 要: 云计算平台利用虚拟化技术使软件应用变得更有效率的同时,也给资源管理和服务调度带来了挑战。在研究了软件服务(SaaS)与基础设施服务(IaaS)调度的区别基础上,重点考虑 SaaS 层的资源调度,提出基于随机理论的调度模型,把该层调度描述成一种多目标的优化问题。除了服务质量的要求,还考虑了弹性这一云服务的重要特性,并提供了任务调度与弹性服务副本的匹配策略。实验表明本调度机制的设计优化了云平台的整体性能,达到了较好的负载均衡与资源利用率。

关键词: 云计算; 软件服务; 弹性; 服务副本; 优化调度

中图分类号: TP393 **文献标志码:** A **文章编号:** 1001-3695(2014)02-0422-04

doi:10.3969/j.issn.1001-3695.2014.02.024

Elastic cloud platform optimal scheduling strategy design based on SaaS

ZHAO Shao-ka[1,2], LI Li-yao[1], XU Cong[2], YANG Jia-hai[2]

(1. *Dept. of Mathematics & Computer Science, Fuqing Branch of Fujian Normal University, Fuqing Fujian* 350300, *China*; 2. *Institute for Network Sciences & Cyberspace, Tsinghua University, Beijing* 100084, *China*)

Abstract: The use of virtualization technology makes software applications more effective which deployed over cloud computing platforms, but also brings challenges to the resource management and service scheduling over cloud. This paper focused on novel mechanisms which provide optimal deployment and scheduling of cloud services in SaaS scenarios on the foundation of studying some differences between IaaS and SaaS scheduling, and proposed a stochastic model to describe the scheduling process as a multi-objective optimization problem. Besides some QoS (quality of service) issues, the most significant property of cloud services-elasticity has been considered in this model, and not only the scheduling of SaaS tasks but also the arrangement of elastic service replicas are provided by our mechanisms. Experiment results show that the deployment and scheduling mechanisms optimize the overall performance, load balance condition as well as the resource usage of the cloud computing platform.

Key words: cloud computing; SaaS(software-as-a-service); elasticity; service replica; optimal scheduling

0 引言

云计算平台允许用户通过云服务的形式获得计算资源,根据提供给用户(租户)的资源或服务的类型,云计算架构分为不同的层次。在基础设施服务(infrastructure as a service, IaaS)层,应用虚拟化技术使许多虚拟机(VM)运行在单个物理服务器上,并提供灵活的资源共享;在软件服务(software as a service, SaaS)层,通过使用 IaaS 层模型提供的虚拟资源,云计算平台能产生一系列的虚拟环境,来满足不同的应用需求。此外,灵活的虚拟资源可以组成相应的套餐,为云计算提供带有弹性(elasticity)的服务,使得部署在云上的各种应用更加高效。

云服务提供商为用户提供资源与服务,但随着云规模的增大,资源管理的复杂度也在不断提高,这就给传统的调度机制带来了挑战。在 IaaS 层,一台物理机中可以运行多台 VM,这使得要获得物理服务器上精确的共享资源(CPU、内存等)信息变得更加困难。此外,动态迁移与云的弹性特征也增加了算法设计的动态性。本文将设计一套调度机制保证云计算的服务质量(quality of service,QoS)要求,提升云平台的整体性能。

1 相关工作

现有云平台上的资源调度研究主要集中在 IaaS 层[1~4],这些调度主要是考虑优化资源供给和 VM 的创建,仅有少数的研究进一步讨论了基于 IaaS 层的 SaaS 层调度问题[5,6],但这些研究也仅集中在如何将资源合理分配到云服务上,本质上依然是一个类似于 IaaS 层的资源分配问题。事实上,SaaS 层与 IaaS 层在调度机制上存在差异:在 IaaS 层,被提供的服务是虚拟资源,因此该层调度的主要约束是云平台中的资源总量,调度目标是确保有限的资源能够服务于无限的租户请求,并最终

收稿日期: 2013-05-21; **修回日期:** 2013-07-09 **基金项目:** 国家教育部—中国移动研究基金资助项目(MCM20123041);福建省教育厅科技项目(JB13198,JA13343,JA12352);福建师范大学福清分校科技研究项目(KY2013001)

作者简介: 赵少卡(1980-),男,福建福州人,讲师,硕士,主要研究方向为云计算、软件工程(zska@cernet.edu.cn);李立耀(1970-),男,副教授,硕士,主要研究方向为计算机网络;徐聪(1986-),男,博士研究生,主要研究方向为计算机网络;杨家海(1966-),男,教授,博士,主要研究方向为计算机网络.

降低请求的丢失率。然而,在 SaaS 层,被提供的服务是软件应用而非资源,在该场景下,所有的请求都能被处理,每个服务所分配的资源多少仅仅是造成执行效率的不同,请求的丢失率在 SaaS 层不再是个原则性的目标。

SaaS 平台的服务调度机制主要包括请求的排队策略与请求和服务副本(service replica)之间的匹配策略。对于排队策略而言,现有的研究主要基于确定性代数模型排队机制[7~10],这些模型策略固定,优化目标单一,主要采用以提高云计算系统的总体吞吐率为核心的优化目标,具体包括请求的响应时间、请求处理效率等服务质量指标,以及服务器负载状况、资源利用率等系统性能指标。其中的代表策略有:以优化服务性能为目标的 Min-Min 调度策略、Suffrage 调度策略;以优化系统性能指标为目标的 Best-Fit、First-Fit 等调度策略;其他一些以优化经济收益为目标的调度策略。总体来看,目前大部分云计算平台使用的都是这些基于代数模型的调度策略,虽然能够满足当前的调度达到最优值,但是缺乏对于阶段性调度结果的总体评估,经常导致请求的丢失以及系统整体吞吐率的下降。为了克服代数模型调度策略的缺陷,学界提出了随机模型来描述任务的执行和调度设计策略。引入随机排队的机制[11~13]对确定性模型进行改进,随着调度进程的进行改变优化目标,性能指标的整体效果较好。

现有的先入先出的排队方式尚未很好地将请求的排队与服务副本的匹配机制相结合,不能真正做到优化稳态调度时云服务的整体性能。对于服务副本的匹配策略而言,主要是对服务等级协定(SLA)中的约束条件进行建模[14~16],研究满足 SLA 约束条件的匹配策略,从而对匹配算法效率进行改进[17],并研究分布式匹配算法,实现服务副本的负载均衡。但在此过程中,弹性这一云计算的重要特性被忽略,匹配机制中没有考虑云服务弹性的优化分配。因此,这样的调度机制也并不完善。

本文将克服上述研究的缺陷,在排队机制的设计上考虑与匹配机制的结合,并在服务副本的匹配机制上考虑云服务弹性的优化分配,以提供一种随机模型来描述云中的资源分配与调度过程,并将其归结为一个多目标的优化问题进行模型设计,从而实现云服务的最优调度。

2　云服务优化调度方案的设计

对于部署在云平台中的服务,通过设计合适的调度策略,规划服务请求与服务副本之间的匹配关系、服务的执行顺序、服务对底层资源的占用情况等,进一步提升云端资源管理服务的质量。

云平台中服务调度场景如图 1 所示。网管服务调用的请求到达云平台后,会先存放在调度队列当中,之后云平台的服务调度模块会根据调度策略,以优化服务执行效率为目标,在队列中选取最合适的请求进行处理,并将该请求分配到最合适的服务副本中进行处理,完成网络资源管理的功能。其中,mapping policy analyzer 负责将任务分配到合适的服务副本以优化平均响应时间,而 elasticity controller 负责云服务的弹性分配以优化资源的利用率。

具体地,在调度方案中需要依次完善服务请求的优化排队策略、请求与服务副本之间的匹配策略以及服务弹性分配策略的实现。采用基于排队论的随机模型,对调度进行到稳定状态时的服务性能进行模型设计。

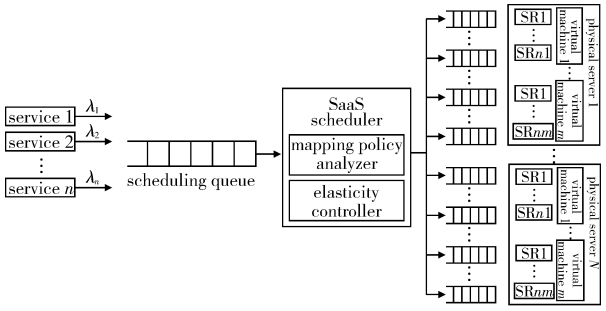

图 1　SaaS 场景下任务调度过程

表 1 介绍了调度模型使用的主要参数及相关定义。

表 1　主要参数及相关定义

参数	定义
λ_m	第 m 个服务请求到达的泊松强度
S_{VM}	云平台中可用虚拟资源的集合
S_{SR}	云平台中所有服务副本的集合
p_{im}	第 i 个服务请求被分配到第 m 个服务副本的概率
μ_{im}	第 i 个服务请求在第 m 个服务副本中的执行效率
ρ_{SRm}	第 m 个服务副本的资源利用率
T_{SRm}	第 m 个服务副本执行一个服务的耗时
L_{total}	调度队列的长度
I_{total}	云平台单位资源处理的任务量
T_{total}	稳定状态时,云平台处理单个服务的平均耗时
M	云服务的种类数目

2.1　云服务执行效率的优化

假定服务请求的到达服从泊松分布,第 i 个服务的请求到达等待队列的强度为 λ_i,则服务到达具体副本 m 的强度为

$$\lambda_{\mathrm{SRm}} = \sum_{i=1}^{M} \lambda_i p_{\mathrm{im}}$$

使用排队论的模型,可以得出调度处于稳定状态时,副本 m 的资源使用率表达式为

$$\rho_{\mathrm{SRm}} = \frac{\lambda_{\mathrm{SRm}}}{\mu_{\mathrm{SRm}}} = \sum_{i=1}^{M} \frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}}$$

根据副本的资源使用率,得到稳态时副本 m 执行一个服务的平均耗时 T_{service} 以及请求排队的平均耗时 T_{wait}:

$$T_{\mathrm{service}} = \frac{1}{\mu_{\mathrm{SRm}}} = \frac{\rho_{\mathrm{SRm}}}{\lambda_{\mathrm{SRm}}} = \frac{1}{\sum_{i=1}^{M} \lambda_i p_{\mathrm{im}}} \cdot \sum_{i=1}^{M} \frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}}$$

$$\sigma_{\mathrm{service}} = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \left(\frac{1}{\mu_{\mathrm{im}}} - T_{\mathrm{service}}\right)^2}$$

$$T_{\mathrm{wait}} = \frac{\rho_{\mathrm{SRm}} T_{\mathrm{service}} \left[1 + \left(\frac{\sigma_{\mathrm{service}}}{T_{\mathrm{service}}}\right)^2\right]}{2(1 - \rho_{\mathrm{SRm}})} = \frac{\sum_{i=1}^{M} \left(\frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}}\right) T_{\mathrm{service}} \left[1 + \left(\frac{\sigma_{\mathrm{service}}}{T_{\mathrm{service}}}\right)^2\right]}{2\left(1 - \sum_{i=1}^{M} \frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}}\right)}$$

达到稳定状态时,副本 m 执行一个服务的总耗时为排队耗时及服务处理耗时的和,因此定位到副本 m 的服务响应时间期望为

$$T_{\mathrm{SRm}} = T_{\mathrm{service}} + T_{\mathrm{wait}} =$$

$$\frac{1}{\sum_{i=1}^{M} \lambda_i p_{\mathrm{im}}} \cdot \sum_{i=1}^{M} \frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}} + \frac{\sum_{i=1}^{M} \left(\frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}}\right) T_{\mathrm{service}} \left[1 + \left(\frac{\sigma_{\mathrm{service}}}{T_{\mathrm{service}}}\right)^2\right]}{2\left(1 - \sum_{i=1}^{M} \frac{\lambda_i p_{\mathrm{im}}}{\mu_{\mathrm{im}}}\right)}$$

进一步得到整个云平台处理服务的整体响应时间,这同时也是该模型的优化目标之一:

$$T_{\mathrm{total}} = \sum_{i=1}^{M} \left(\lambda_i \cdot \frac{\sum_{m=1}^{|S_{\mathrm{SR}}|} p_{\mathrm{im}} T_{\mathrm{SRm}}}{\sum_{m=1}^{|S_{\mathrm{SR}}|} \lambda_{\mathrm{SRm}}}\right) \tag{1}$$

2.2 云服务的弹性分配

除了优化服务的执行效率,提高云平台工作的整体响应时间之外,本调度策略还进一步考虑了云服务的弹性分配,提高资源的使用效率,避免资源的过量和欠量分配。关于云服务弹性分配部分的具体建模方式如下:

首先使用排队模型,得到稳态时副本 m 积压的请求量为

$$L_{\mathrm{SRm}}=\lambda_{\mathrm{SRm}}T_{\mathrm{SRm}}=\sum_{i=1}^{M}\lambda_i p_{\mathrm{im}}(T_{\mathrm{wait}}+T_{\mathrm{service}})$$

接着利用资源利用率,得到稳态时副本 m 单位资源处理的请求量为

$$I_{\mathrm{SRm}}=\frac{L_{\mathrm{SRm}}}{\rho_{\mathrm{SRm}}}=\frac{\sum_{i=1}^{M}\lambda_i p_{\mathrm{im}}(T_{\mathrm{wait}}+T_{\mathrm{service}})}{\rho_{\mathrm{SRm}}}$$

进一步得到整个云平台单位资源处理的任务量,这同时也是本模型的另一个优化目标:

$$I_{\mathrm{total}}=\frac{L_{\mathrm{total}}}{\rho_{\mathrm{SRm}}|S_{\mathrm{SR}}|}=\frac{\sum_{i=1}^{M}\sum_{m=1}^{|S_{\mathrm{SR}}|}\frac{\lambda_i p_{\mathrm{im}}T_{\mathrm{SRm}}}{\rho_{\mathrm{SRm}}}}{|S_{\mathrm{SR}}|}\tag{2}$$

综合式(1)和(2),将云服务的优化调度问题抽象为一个多目标的优化模型:找到最优的分配概率 p_{im},从而最小化服务的响应时间 T_{total};同时,依据服务的弹性最大化平均执行任务量 I_{total},即

$$\min\ T_{\mathrm{total}}=\sum_{i=1}^{M}\left(\lambda_i\cdot\frac{\sum_{m=1}^{|S_{\mathrm{SR}}|}p_{\mathrm{im}}T_{\mathrm{SRm}}}{|S_{\mathrm{SR}}|}\right)$$

$$\max\ I_{\mathrm{total}}=\frac{\sum_{i=1}^{M}\sum_{m=1}^{|S_{\mathrm{SR}}|}\frac{\lambda_i p_{\mathrm{im}}T_{\mathrm{SRm}}}{\rho_{\mathrm{SRm}}}}{|S_{\mathrm{SR}}|}$$

$$\text{s.t.}\begin{cases}\sum_{m=1}^{|S_{\mathrm{SR}}|}p_{\mathrm{im}}=1,\ \forall 1\leqslant i\leqslant M,1\leqslant m\leqslant|S_{\mathrm{SR}}|\\ \forall m\in S_{\mathrm{SR}},\forall 1\leqslant i\leqslant M,\ \sum_{m}^{p_{\mathrm{im}}>0}\mu_{\mathrm{SRm}}\propto\sum_{m}^{p_{\mathrm{im}}>0}L_{\mathrm{SRm}}\\ \sum_{m}^{p_{\mathrm{im}}>0}|S_{\mathrm{SRm}}|\leqslant|S_{\mathrm{VM}}|\\ |S_{\mathrm{SR}}|\geqslant M\end{cases}$$

该优化策略的主要约束条件是:分配概率的总和为 1;单个服务的弹性与服务的负载正相关;分配的服务副本总量不能超过云平台所能提供的虚拟资源总量;云平台中所有服务副本集不能小于服务请求的种类总量。

调度算法的核心思想是合理调整服务排队、服务匹配以及服务弹性分配策略,使上述优化模型达到最优取值,提高服务响应时间,同时最大化单位资源的工作效率。

2.3 弹性云服务的调度算法

算法由两部分组成:第一部分(6 ~ 14 行)是在服务请求与服务副本间找到一个优化的映射关系,从而最小化 SaaS 平台的平均响应时间 T_{total};第二部分(15 ~ 32 行)是设计弹性优化分配机制,从而最大化平台的执行任务量,提高资源的利用率。其中,第二部分又可以划分为两大步骤:第 15 ~ 26 行解决当云平台有空闲的资源时,如何增加合适的服务弹性来提高执行效率;第 27 ~ 32 行解决当平台资源满载时,如何对每个云服务进行合理的弹性调度安排从而达到平台的负载均衡。

具体的实现过程的伪代码如下:

```
1  分别初始化 S_SR 和 S_VM,且|S_SR| = M,说明初始时每个服务仅有一个副本;
2  初始化任务集 S_T 与服务副本集 S_SR 之间的映射表,最大化各个服务副本的队列长度 L_max;
3  初始化优化目标 T_optimal = INFINITE;
4  初始化优化目标 I_optimal = 0;
5  for 调度队列队头的每个任务 i , do
6     根据映射表创建服务副本 S_A 的候选集;
7     for 每个在 S_A 中的服务副本 m
8           令 p_im = 1,根据式(1)计算 T_total;
9           if T_total < T_optimal and L_SRm < L_max
10                T_optimal = T_tatal, dest = m;
11          end if
12    end for
13    将当前的任务 i 分配到服务副本 dest;
14    L_SRdest = L_SRdest + 1;
15    if L_total/(L_max · |S_SR|) > |S_SR|/|S_VM| and |S_SR| < |S_VM| · M
16       for 每个在 S_SR 中的服务副本 m do
17          if L_SRm > L_max · |S_SR|/|S_VM|
18             创建被 m 映射的任务 i 的新副本,计算 I_total;
19             if I_total > I_optimal and Σ_m^{p_im>0} |S_SRm| ≤ |S_VM|
20                   I_optimal = I_total, aim = i;
21                end if
22             end if
23             创建任务 aim 的新副本;
24             更新映射表;
25          end for
26    end if
27    if |S_SR| = |S_VM| · M
28          for 在 S_T 中的每个任务 i,do
29             Σ_m^{p_im>0} |S_SRm| = |S_VM| · M · Σ_m^{p_im>0} L_SRm/L_total;
30             更新映射表;
31          end for
32    end if
33 end for
```

3 实验与结果分析

为了验证算法的有效性,在云端部署五种不同的软件应用服务,每一服务请求的到达满足不同强度的泊松分布,根据算法 1 的策略,在考虑弹性云服务策略的同时,使得每一服务请求最优地分配至服务副本。在 6 台物理机上创建 10 台具有相同虚拟资源的虚拟机,在每一虚拟机上最多创建三种不同的服务,这样,在云端最多有 30 个服务副本,某个特定的软件应用的弹性增长最多是 10 个副本。

将本调度机制模型与被广泛使用的代数模型调度机制,还有一般的普通随机调度模型进行比较。图 2 和 3 表示在四种调度机制下,任务的积压量与平均响应时间的变化状况。

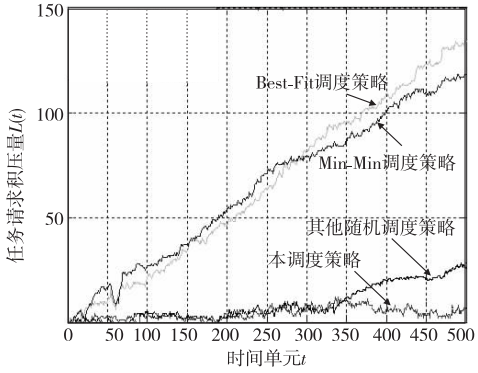

图 2 任务积压量曲线

使用代数模型(Min-Min、Best-Fit 等算法)的调度机制,任务的积压量高。原因是这些算法在调度进行前,很多需要事先判断并选择最早完成时间和最大吞吐率的任务来执行,这就需

要消耗时间来寻找在调度队列中的“优质”任务,但是,如果“优质”任务没有被执行,其他任务也不能被执行,这就导致了任务的大量积压。如果没有合理选择每个服务副本等待队列的队长,将会引起大量任务请求的丢失,造成无限等待。然而,使用随机模型,每个任务都有可能被执行,通过设计最合适的任务与服务副本间的映射关系,而不是通过挑选“优质”的任务来达到优化目标,因此未被处理的任务积压量就相当少。所以,比起代数模型,随机模型在设计调度机制上是一个比较好的选择。

从图 2 中还可以看出,虽然同样是随机模型,比起一般的随机模型,尤其是随着执行时间的增加,本算法在降低任务的积压上效果更明显。这是由于采用了服务的弹性安排策略,根据云服务的负载对每个服务的弹性作了优化的安排:当负载重时,增加这些服务的副本来提高执行效率,从而使未被处理的积压任务数自然地迅速减少。

此外,虽然所有的服务都会在有限的时间内被处理,但是每一服务的平均响应时间在不同算法模型下是不同的,如图 3 所示。这一现象的主要原因是不恰当的服务弹性分配策略,若没有考虑到弹性就很有可能造成服务分配上的负载不均,从而导致等待队列与响应时间的增长。本算法引入了弹性,从而将根据实际需要增加负载较重的服务副本并减少轻量级服务的副本数量,从而在云服务的负载均衡上也显现出优势,确保了服务的负载在一个较低的水平并降低任务的积压量。

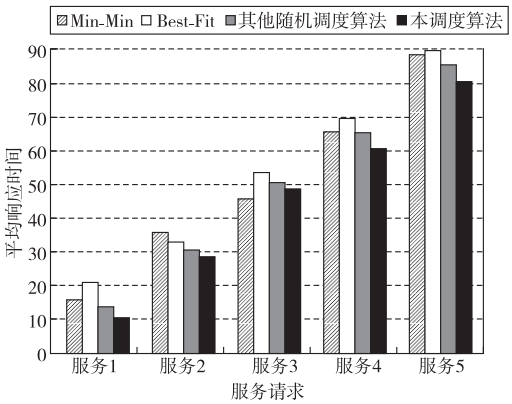

图 3　服务平均响应时间图

实验结果表明:本调度机制能有效地改进服务的执行效率,降低服务的积压,并实现 SaaS 平台中云服务的负载均衡。

4　结束语

本文提出了在 SaaS 场景下的云服务优化部署与调度机制,提出一种随机模型将 SaaS 的调度描述成多目标的优化问题。设计出的新的 SaaS 调度机制,综合考虑了服务的性能和弹性分配。根据实验结果,本调度机制根据服务的弹性安排,有效地改进了 SaaS 平台任务的平均执行量,从而保障了云平台的高性能。

在后续研究中,将继续考察云服务的更多问题,如应用的安全约束、迁移的开销、不同服务间的数据共享和数据耦合等,都需要将现有的随机模型进行进一步设计与完善。此外,弹性云服务的调度机制也有待于继续改进。

参考文献:

[1] BREITGAND D, EPSTEIN A. SLA-aware placement of multi-virtual machine elastic services in compute clouds[C]//Proc of IFIP/IEEE International Symposium on Integrated Network Management. 2011: 161-168.

[2] WOOD T, TARASUK-LEVIN G, SHENOY P J, *et al*. Memory buddies: exploiting page sharing for smart colocation in virtualized data centers[C]//Proc of ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. New York: ACM Press, 2009:31-40.

[3] SUDEVALAYAM S, KULKARNI P. Affinity-aware modeling of CPU usage for provisioning virtualized applications[C]//Proc of IEEE International Conference on Cloud Computing. 2011:139-146.

[4] KHULLER S, LI Jian, SAHA B. Energy efficient scheduling via partial shutdown[C]//Proc of the 21st ACM-SIAM Symosium on Discrete Algorithms. 2010:1360-1372.

[5] LI Hui, CASALE G, ELLAHI T. SLA-driven planning and optimization of enterprise applications[C]//Proc of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering. New York:ACM Press, 2010:117-128.

[6] ALI-ELDIN A, TORDSSON J, ELMROTH E. An adaptive hybrid elasticity controller for cloud infrastructures[C]//Proc of IEEE Network Operations and Management Symposium. 2012:204-212.

[7] SPEITKAMP B, BICHLER M. A mathematical programming approach for server consolidation problems in virtualized data centers [J]. Trans on Services Computing,2010,3(4):266-278.

[8] BELOGLAZOV A, BUYYA R. Energy efficient allocation of virtual machines in cloud data centers[C]//Proc of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. 2010: 577-578.

[9] QIN Xiao, XIE Tao. An availability-aware task scheduling strategy for heterogeneous systems[J]. IEEE Trans on Computers, 2008,57(2):188-199.

[10] SONG S, HWANG K, KWOK Y. Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling[J]. IEEE Trans on Computers, 2006,55(6):703-719.

[11] MAGULURI S T, SRIKANT R, YING Lei. Stochastic models of load balancing and scheduling in cloud computing clusters[C]//Proc of IEEE INFOCOMM. 2012:702-710.

[12] BRAMSON M, LU Yi, PRABHAKAR B. Randomized load balancing with general service time distributions[C]//Proc of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York: ACM Press, 2010:275-286.

[13] MAGULURI S T, SRIKANT R, LEI Ying. Heavy traffic optimal resource allocation algorithms for cloud computing clusters[C]//Proc of the 24th International Teletraffic Congress. 2012:25.

[14] BONVIN N, PAPAIOANNOU T G, ABERER K. Autonomic SLA-driven provisioning for cloud applications[C]//Proc of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2011:434-443.

[15] NANDI B B, BANERJEE A, GHOSH S C. Elastic cloud resource management with dynamic SLA: a SaaS perspective[C]//Proc of the 13th IFIP/IEEE International Symposium on Integrated Network Management. 2013.

[16] MOON H J, CHI Y, HACIGUMUS H. A SLA-aware profit optimization in cloud services via resource scheduling[C]//Proc of the 6th IEEE World Congress on Services. 2010.

[17] WENDEL P, JIANG J W, FREEDMAN J. DONAR: decentralized server selection for cloud services[C]//Proc of ACM SIGCOMM. 2010:231-242.

[18] 林伟伟,齐德昱.云计算资源调度研究综述[J].计算机科学,2012,39(10):1-6.

[19] 田文洪.云计算资源调度管理[M].北京:国防工业出版社,2011.

[20] 左利云,曹志波.云计算中调度问题研究综述[J].计算机应用研究,2012,29(11):4023-4027.