doi:10.3969/j.issn.1672-1292.2016.03.013

云数据中心面向实时任务的节能调度算法

鲍文燕,张银娟,李 晨,李 云

(扬州大学信息工程学院,江苏 扬州 225127)

[摘要] 针对云计算环境下的独立实时任务的节能调度问题进行了研究,设计了一种基于松弛时间的任务调度算法,该算法由实时任务的分配、虚拟机资源的动态扩展以及虚拟机的动态整合3个部分组成,通过计算任务的松弛时间保证任务在截止期限内完成,保证任务的时效性.同时提出了一种基于多阈值的虚拟机整合策略,以平衡系统负载并降低系统完成任务集合的能耗.实验表明,与其他算法相比,该算法在保证了任务能够按时完成的基础上,有效降低了系统的整体能耗.

[关键词] 云数据中心,实时任务,任务松弛时间,节能调度

[中图分类号]TP393 [文献标志码]A [文章编号]1672-1292(2016)03-0081-07

Energy-Efficient Scheduling Algorithm for Real-Time Tasks in Cloud Data Center

Bao Wenyan,Zhang Yinjuan,Li Chen,Li Yun

(College of Information Engineering,Yangzhou University,Yangzhou 225127,China)

Abstract: In this paper,we study the problem of energy-efficient scheduling for real-time tasks. A task scheduling algorithm based on the slack time of tasks is designed. Three components of the algorithm are the distribution of real-time tasks,dynamically expanding virtual machine resource and integration of virtual machine resource. By computing the slack time of tasks,tasks can be completed within the deadline to ensure the timeliness of the task. With a multi-threshold-based virtual machine integration strategy,the system load is balanced and energy consumption of the system to complete the task set is reduced. Experiments show that,comparing with two other scheduling algorithms,the algorithm in this paper ensures that tasks can be completed on time,and energy consumption of the system can be efficiently reduced.

Key words: cloud data center,real-time tasks,slack time of tasks,energy-efficient scheduling

数据中心的节能管理工作中,最大的挑战在于要在保证服务质量的基础上进一步实施节能策略,来降低数据中心的能耗[1].目前被广泛研究和实践的主要节能策略,如对CPU频率进行动态调整、关闭空闲机器或使其进入睡眠状态、虚拟机的迁移合并等等,这些策略的实质都是通过对数据中心资源的动态调度和整合,来使能源利用率最大化.

赵彬等人[2]通过在调度任务时优先考虑处于运行状态的服务器,并以任务响应时间为约束,按照最小能耗原则将任务分配到相应的服务器上执行.只有当处于运行状态的服务器不满足任务对响应时间的要求时,考虑休眠状态的服务器,但没有考虑到根据系统的运行状态将虚拟机进行合理的迁移.谭一鸣等[3]提出了一个满足性能约束的最小期望执行能耗调度算法,该算法可以根据所有服务器的不同负载情况调度策略.何丽等人[4]提出了基于能耗优化的最早完成时间任务调度策略,该调度策略能够根据任务的截止时间要求和服务器的执行能耗,尽可能将任务映射到执行能耗最小的服务器上执行.但是这两种算法都未考虑到物理机和虚拟机的整合问题.

收稿日期:2016-07-17.

基金项目:江苏省自然科学基金(BK20161338)、江苏省"六大人才高峰"高层次人才项目(2012-WLW-024)、江苏省产学研联合创新资金(前瞻性联合研究)项目(BY2013063-10)、扬州大学研究生创新项目(CXLX_1415)、扬州市"绿扬金凤计划"创业创新领军人才项目(2013-50).

通讯联系人:鲍文燕,工程师,研究方向:计算机应用技术.E-mail:wybao@yzu.edu.cn

本文对影响数据中心服务器能耗的各项因素进行逐一分析,并结合实验数据,建立服务器能耗模型,提出了面向实时任务的节能算法,该算法由实时任务的分配、虚拟机资源的动态扩展和虚拟机的整合3个部分组成.本文详细描述了该算法的应用场景和任务调度模型,并设计了一种基于松弛时间的任务调度策略以及基于多阈值的虚拟机整合策略.

1　实时任务调度模型

1.1　系统模型

在本文中,云数据中心被描述为一个由 m 个异构物理服务器组成的集合 $H=\{h_1,h_2,\cdots,h_m\}$,对于系统中任意一台物理机 j,可表示为一个三元组 $h_j=\{C_j,N_j,P_j\}$,其中,C_j 表示该物理机的CPU计算能力,N_j 则表示该物理机的网络带宽,P_j 是该物理机满负载时的功率大小.

物理机 j 上的 k 个虚拟机集合则表示为 $\{vm_{j1},vm_{j2},\cdots,vm_{jk}\}$,物理机上每台虚拟机可表示为 $vm_{jk}=\{c_{jk},n_{jk}\}$,与物理机类似,$c_{jk}$ 和 n_{jk} 分别表示该虚拟机的被分配的CPU计算能力和网络带宽.任务调度模型如图1所示.

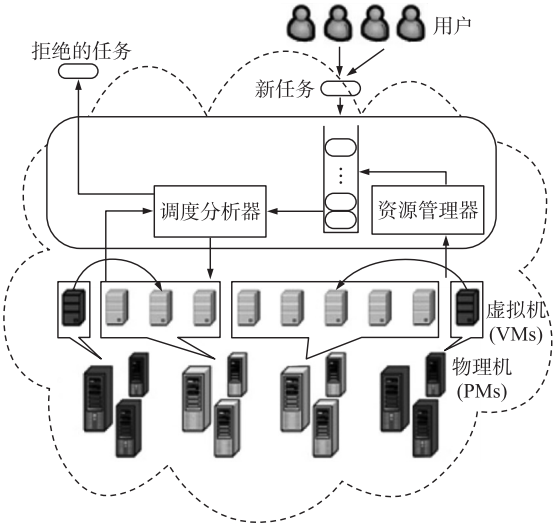

图1　云数据中心任务调度模型

Fig.1　Tasks schedule model in cloud data center

面向实时任务的调度系统模型主要由以下3部分组成:

等待队列WQ(Waiting Queue):在本章节的调度模型中,每当一个新任务到达系统时,该任务将被存放于等待队列WQ中.

调度分析器(Schedulablity Analyzer):调度分析器在新任务到达时,对当前等待队列中所有任务进行分析,计算当前所有等待执行任务的松弛时间,根据任务紧急程度不同决定任务的调度顺序,拒绝无法按时完成的任务.

资源管理器(Resource Monitor):资源管理器主要负责对当前系统负载情况的检测,并对系统中的计算资源进行调整,监测系统中物理主机的闲置时间,关闭闲置时间过长的物理主机,同时通过虚拟机迁移技术对计算资源进行整合,将任务集中于尽可能少的物理主机上执行.

1.2　调度问题描述

在本文中,云数据中心要完成的是一组包含 n 个独立实时任务的任务集合 T,其中每个任务可表示为 $t_i=\{a_i,l_i,d_i,s_i\}$,其中,a_i 表示该任务的到达时间,l_i 表示该任务的计算长度(任务所需的基本操作数),d_i 表示该任务的截止期限,s_i 表示该任务的传输数据大小.

本文使用 X_{ijk} 来表示任务 t_i 与虚拟机 vm_{jk} 的分配关系:

$$X_{ijk}=\begin{cases}1, & \text{if } t_i \text{ is allocated to } vm_{jk};\\ 0, & \text{otherwise.}\end{cases} \tag{1}$$

当任务 t_i 被分配到虚拟机 vm_{jk} 上执行时,X_{ijk} 的值被置为1,否则置为0.

使用 st_{ijk}、et_{ijk} 和 ft_{ijk} 分别表示任务 t_i 在虚拟机 vm_{jk} 上的开始时间、执行时间和完成时间,则任务 t_i 在 vm_{jk} 上执行的完成时间可表示为:

$$et_{ijk}=l_i/c_{jk}, \tag{2}$$

$$ft_{ijk}=st_{ijk}+et_{ijk}+s_i/n_{jk}. \tag{3}$$

使用 $O_j(t)$ 来表示在任意时刻 t 物理机 j 的使用状态:

$$O_j(t)=\begin{cases}1, & \text{host } j \text{ is active at time } t;\\ 0, & \text{otherwise.}\end{cases} \tag{4}$$

对于任意时刻 t,若物理机 j 处于开启状态,则 $O_j(t)$ 的值为 1,否则为 0.

当任务的完成时间 ft_{ijk} 小于任务的截止期限 d_i 时,任务才算成功完成 . 此处使用 Y_{ijk} 表示任务 t_i 在虚拟机 vm_{jk} 上的执行结果 . 若 Y_{ijk} 的值置为 1,表示任务 t_i 在虚拟机 vm_{jk} 上按时完成,否则 Y_{ijk} 的值被置为 0.

$$Y_{ijk}=\begin{cases}1, & \text{if } (ft_{ijk}\leqslant d_i)\text{ and }(X_{ijk}=1);\\ 0, & \text{otherwise}.\end{cases} \tag{5}$$

为了满足用户的需求,对于该任务集合,在设计调度算法时首要目标是最大化任务集合的完成率,表示如下:

$$\text{Maximize}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{|vm_j|}\frac{Y_{ijk}}{n}\ . \tag{6}$$

在优先保证任务完成率的基础上,本文的另一个优化目标是降低云数据中心完成这一任务集合的总的能源消耗,根据文献[5]的服务器能耗模型,可得:

$$\text{Minimize}\sum_{j=1}^{m}\int_{ST}^{FT}((k\cdot P_{\max}\cdot O_j(t))+(1-k)\cdot P_{\max}\cdot u(t))\mathrm{d}t\ , \tag{7}$$

式中,$P_{\max}$ 代表服务器的最大功耗,u 为服务器 CPU 的资源利用率,k 为静态功耗占总功耗的比重,k 的值为 0.7,ST 和 FT 为该任务集合的开始时间和结束时间 .

2 基于松弛时间的任务调度策略

2.1 任务的松弛时间

为了决定任务的调度顺序,本文通过计算每个任务的松弛时间(slack-time)衡量其紧急程度 .

定义 1 任务的松弛时间:任务的松弛时间为其截止期限与其最快预期执行时间以及当前时间的时间差 . 一个任务的松弛时间表示为:

$$\mathrm{SL}_i=d_i-ct-\frac{l_i}{\mathrm{Max}\{c_{jk}\}}-\frac{s_i}{\mathrm{Max}\{n_{jk}\}}\ , \tag{8}$$

式中,d_i 为任务 t_i 的截止时间,ct 为系统当前时间,$\max\{c_{jk}\}$ 表示当前系统中单个虚拟机的最大 CPU 能力,l_i 表示任务 t_i 的计算长度,s_i 为任务 t_i 的传输数据大小 .

定义 2 任务的实际松弛时间(real-slack-time):当任务 t_i 被调度到虚拟机 vm_{jk} 上执行时,若虚拟机已存在,该任务的实际松弛时间为该任务的截止期限与该任务在虚拟机 vm_{jk} 上执行的预期执行时间以及当前时间的差值,若虚拟机需要重新部署即资源需要扩展,则需考虑部署虚拟机的时间,可表示为:

$$\mathrm{RSL}_i=\begin{cases}d_i-ct-l_i/c_{jk}-s_i/n_{jk}, & \text{虚拟机已存在;}\\ d_i-ct-l_i/c_{jk}-s_i/n_{jk}-T_{vm\text{on}}, & \text{需开虚拟机;}\\ d_i-ct-l_i/c_{jk}-s_i/n_{jk}-T_{vm\text{on}}-T_{host\text{on}}, & \text{需开物理机.}\end{cases} \tag{9}$$

式中,$T_{vm\text{on}}$ 及 $T_{host\text{on}}$ 分别表示资源扩展时的虚拟机开启时间和主机开启时间,其值主要由物理机的性能决定,在本文中假定其均为常量值 .

在调度任务时,将任务的实际松弛时间最小化作为调度目标,既保证了任务的时效性,也使得任务的截止期限得到最大化利用 . 同时,由于使用了尽可能少的计算资源完成任务,使得系统中可同时运行的任务数量最大化,提高了系统的负载能力 .

2.2 计算资源的扩展

当系统中当前的计算资源无法满足 WQ 的任务的计算需求时,需要对计算资源进行扩展,即部署新的虚拟机,此时存在两种情况:

(1)在已经开启的物理机上部署一个满足任务计算需求的虚拟机,此时任务的开始时间需考虑部署一个新的虚拟机所需要的时间 $T_{vm\text{on}}$.

(2)若系统中已经开启的物理机上无法部署满足要求的虚拟机,此时需要新开启一台物理机,并在其

上部署新的虚拟机,此时任务的开始时间需要加上开启物理机所耗的时间 $T_{host\,\mathrm{on}}$ 以及开启虚拟机所耗的时间 $T_{vm\,\mathrm{on}}$.

部署新虚拟机的伪代码可表示如下:

算法 1　CreateNewVM()

```
Select a VM vm_jk with minimal MIPS that can finish task t_i before its deadline considering the delay of creating vm_jk
if vm_jk! =NULL then
 for each active host h_j do
if VM vm_jk can be created on host h_j then
 Create VM vm_jk on host h_j; return vm_jk;
end if
 end for
else
 Select a VM vm_jk with minimal MIPS that can finish task t_i before its deadline considering the delays of turning on a host and creating VM vm_jk
  if vm_jk! =NULL then
  Turn on a host h_j and then create VM vm_jk on it;
  return vm_jk;
  end if
  end if
```

2.3　最小松弛时间调度算法

最小松弛时间调度算法(Minimize Slack-time Scheduling Algorithm,MSLA)的主要调度步骤如下:

算法 2　MSLA

```
for each new task t_i do
Delete all the allocating relationships between tasks in WQ and VM
WQ←t_i;
Sort all the tasks in the WQ by their SL_i in ascending order;
 for each task in WQ do
 Select a VM vm_jk with minimal MIPS that can finish task t_i before its deadline
 if vm_jk! =NULL then
 allocate task t_i to vm_jk
 else
  vm_jk←CreateNewVM();
  if vm_jk! =NULL then
  allocate task t_i to vm_jk
  else
  reject task t_i
  end if
  end if
  end for
  end for
```

3　多阈值虚拟机整合策略

随着任务到达速率的变化,系统负载变轻时,为了降低系统的能耗,首先将系统中空闲时间超过设定阈值的物理主机关闭,然后使用一种多阈值虚拟机整合策略(Multi-threshold VM Integration Policy,MTIP),对系统中的虚拟机资源进行整合,平衡各物理主机的负载,降低系统中开启的物理主机的数量.

在对云数据中心虚拟机进行整合时,首先设置 3 个物理主机 CPU 利用率阈值 α、β 和 γ,3 个阈值的大小关系为 $\alpha<\beta<\gamma$.

MTIP 策略通过设置 CPU 利用率阈值,将物理主机的 CPU 利用率划分为几个区间,并根据某一台 CPU 利用率所位于的区间,判断其负载情况,并以此为根据对部署于其上的虚拟机使用不同的处理方式.

通过阈值判断物理主机负载情况的标准如下:

(1)当物理主机 CPU 的利用率小于或等于 α 时,该主机负载过低;

(2)当物理主机 CPU 的利用率大于 α 小于等于 β 时,该主机负载较低;

(3)当物理主机 CPU 的利用率大于 β 小于等于 γ 时,该主机负载正常;

(4)当物理主机 CPU 的利用率大于 γ 时,该主机负载过高.

多阈值虚拟机整合策略的主要思想如下:

(1)负载过低的主机上的虚拟机被迁移至负载较低的主机上,然后将迁移后的空闲物理主机切换到睡眠模式,或将其关闭以降低能耗.

(2)负载正常的物理主机上的虚拟机保持不变,即不发生迁移.

(3)负载过高的物理主机上的部分虚拟机,被迁移至负载较低的物理主机上.当系统中某一台主机负载过高时,MTIP 首先选择该物理主机中占用物理主机 CPU 利用率最低的虚拟机进行迁移,然后选择该物理主机中占用 CPU 利用率最高的虚拟机进行迁移,如此循环反复,直至该物理主机 CPU 的利用率小于 γ.

4 面向实时任务的节能调度算法仿真实验

本文提出了面向实时任务的节能调度算法,基于实时任务的分配、虚拟机资源的动态扩展和虚拟机的整合,设计了一种结合多阈值的虚拟机整合策略 MTIP 和基于松弛时间的任务调度策略 MSLA 的实时任务节能调度算法,简记为 M-MSLA 算法.

4.1 实验参数设置

针对面向实时任务的节能调度,Li 和 Muthucumaru 分别提出了 EST 算法以及 MCT 算法[6-7]. EST 算法在选择虚拟机时,选择将任务调度到有最早开始时间的虚拟机上执行.MCT 算法是一种实时任务调度中常用的贪婪调度算法,该算法在选择虚拟机时,在保证任务截止期限得到满足的基础上,优先选择具有最早完成时间的虚拟机进行调度,该算法没有对虚拟机资源进行整合.

本文主要根据以下几个指标,通过模拟实验测试对 M-MSLA、EST 和 MCT 调度算法的性能进行比较:

(1)开启的物理主机数量:完成某一任务集合,所开启的物理主机总数.

(2)能量消耗[8]:完成某一任务集合所消耗的总能量.

实验中物理主机、虚拟机和任务的主要参数设置如表 1 所示.

4.2 实验结果与分析

在通过模拟实验对 M-MSLA、EST 和 MCT 调度算法的性能进行比较时,本文分别对不同任务数量和不同的任务截止期限这两种情况下的算法性能进行了实验验证.

4.2.1 不同任务数量情况下算法性能的对比

在第一组实验中,为了测试任务数量对 M-MSLA 算法性能的影响,分别使用 M-MSLA、EST 和 MCT 调度算法对任务数为 500 到 4 000 的 8 个不同的任务集合进行调度,将任务的截止期限设置为任务的到达时间加上 250 s,对开启物理主机的数量及消耗的能量分别进行测试.对 M-MSLA、EST 以及 MCT 算法在不同任务数量情况下调度使用的物理机数量进行对比,实验结果如图 2 所示.

表 1 实验参数设定

Table 1 Experimental parameters determination

参数	设定值
物理主机的 CPU 性能(MIPS)	(1 000,1 500,2 000)
物理主机最大功率(W)	(250,300,400)
虚拟机的 CPU 性能(MIPS)	(250,500,750,1 000)
任务数量	(500,1 000,1 500,…,3 500)
任务计算长度(MI)	$[1\times10^5,2\times10^5]$
任务到达时间间隔(s)	[0,4]
任务最大执行时间(s)	(150,200,…,500)

从图 2 可以发现,随着任务数量的增加,开启的物理主机数量也随之增加,以保证任务执行结果的时效性.同时,使用 M-MSLA 算法进行调度开启的物理主机数量明显少于其他两种算法,这是因为 M-MSLA 算法采用了 MTIP 策略对虚拟机进行动态整合,用更少的物理主机完成任务集合,提高了系统的资源利用率,同时也提高了数据中心的最大负载能力.

对 M-MSLA、EST 及 MCT 算法在不同任务数量情况下的总能量消耗进行对比实验的结果如图 3 所示 . 从图 3 可以看出,相对于 EST 算法和 MCT 算法,M-MSLA 算法完成同一任务集合的能耗更低,这是因为 M-MSLA 算法使用了虚拟机整合策略,提高了资源的利用率,且随着任务数量的增加,资源整合的效果更加显著 .

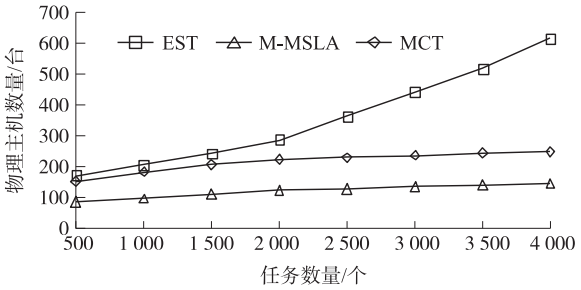

图2　不同任务数量情况下开启的物理主机数量

Fig.2　The number of the started physical machines under different number of tasks

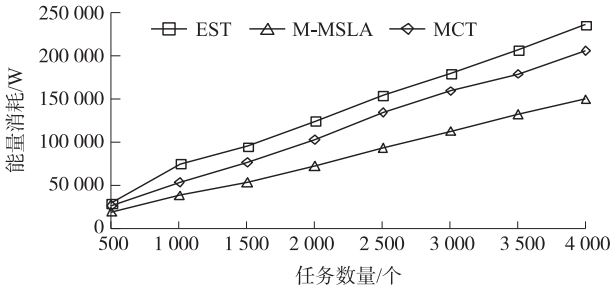

图3　不同任务数量情况下的能量消耗

Fig.3　Energy consumption under different number of tasks

4.2.2　不同任务截止期限下算法性能的对比

在第二组实验中,为了测试任务到达率对 M-MSLA 算法性能的影响,对于任务数量为 1 000 的任务集合,将任务的截止期限设置为任务的到达时间加上 150 s,200 s,250 s,…,500 s,分别使用 M-MSLA、EST 和 MCT 算法对任务集合进行调度,对开启物理主机的数量及能量消耗进行测试 .

对 M-MSLA 算法、EST 算法和 MCT 算法在不同任务截止期限下开启的物理主机总数进行对比,实验结果如图 4 所示 . 图 4 显示了随着任务截止期限的不断放宽,EST 算法和 MCT 算法需要的物理主机数量随之增加,这是因为截止期限的增加使得更多的任务可以被完成,因此需要更多的虚拟机和物理主机 .

对 M-MSLA 算法、EST 算法和 MCT 算法在不同任务截止期限下完成任务集合的总能量消耗进行对比实验的结果如图 5 所示 . 从图 5 可以发现,随着任务截止期限的不断放宽,3 个算法消耗的能量也随之增加,这是由于更长的截止期限使得系统能够完成更多的任务,从而增加系统中虚拟机和物理主机的数量和工作时间,但 M-MSLA 算法在能量消耗上明显少于 EST 算法和 MCT 算法 .

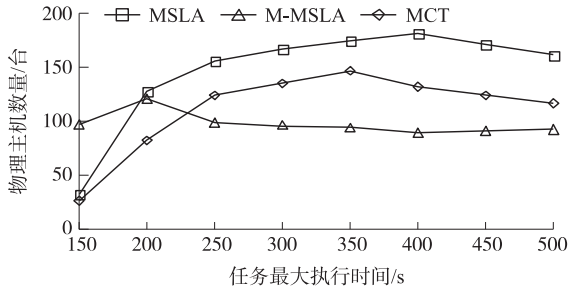

图4　不同任务截止期限下开启的物理主机数量

Fig.4　The number of the started physical machines under different number of tasks

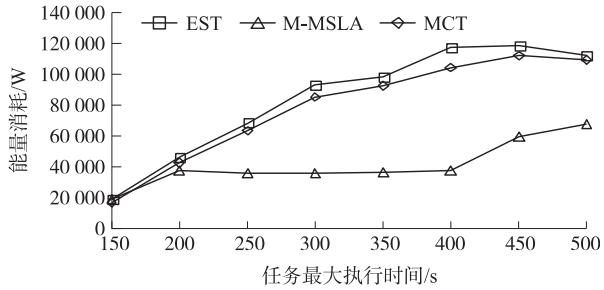

图5　不同任务截止期限下的能量消耗

Fig.5　Energy consumption under different deadline of tasks

通过对比实验结果可以看出,本文提出的算法在任务完成率及节能效果上都优于 EST 算法和 MCT 算法,在很好地保证了任务时效性的基础上降低了系统完成任务集合的能耗 .

5　结语

本文通过计算任务的松弛时间,将任务调度到合适的虚拟机执行以保证任务的时效性,同时通过最小化任务的实际松弛时间,最大化利用系统计算资源,通过多阈值虚拟机整合策略平衡系统负载,降低系统中物理主机的数量,达到降低系统整体能耗的效果 . 但该算法中对于任务模型的定义较为简单,与真实数据中心中处理的任务情况存在较大差别,今后将会对云数据中心的能耗模型进行进一步完善,在已知数据中心硬件架构的情况下,在对任务进行分配和对虚拟机进行整合时,可考虑将计算任务集中在尽可能少的架构上执行,从而减少通信和制冷设备的能耗 .

[参考文献](References)

[1] 巫晨云 . 数据中心能效影响因素及评估模型浅析[J]. 电信工程技术与标准化,2014(1):46-49.
WU C Y. A brief analysis of the factors and evaluation models of data center energy efficiency[J]. Telecom engineering technics and standardization,2014(1):46-49. (in Chinese)

[2] 赵彬,王淖,王高才 . 云计算环境下的节能任务调度策略的随机 Petri 网分析[J]. 计算机科学,2015,42(8):112-117.
ZHAO B,WANG N,WANG G C. Analysis on energy-saving task scheduling strategy based on stochastic Petri net for cloud computing.[J]. Computer science,2015,42(8):112-117. (in Chinese)

[3] 谭一鸣,曾国荪,王伟 . 随机任务在云计算平台中能耗的优化管理方法[J]. 软件学报,2012,23(2):266-278.
TAN Y M,ZENG G S,WANG W. Policy of energy optimal management for cloud computing platform with stochastic tasks[J]. Journal of software,2012,23(2):266-278. (in Chinese)

[4] 何丽,饶俊,赵富强 . 一种基于能耗优化的云计算系统任务调度方法[J]. 计算机工程与应用,2013,49(20):19-22.
HE L,RAO J,ZHAO F Q. Task scheduling method based on energy optimization in cloud computing system[J]. Computer engineering and applications,2013,49(20):19-22. (in Chinese)

[5] FAN X,WEBER W D,BARROSO L A. Power provisioning for a warehouse-sized computer[J]. ACM SIGARCH computer architecture news,2007,35(2):13-23.

[6] LI J,MING Z,QIU M,et al. Resource allocation robustness in multi-core embedded systems with inaccurate information[J]. Journal of systems architecture,2011,57(9):840-849.

[7] MUTHUCUMARU M,SHOUKAT A,HOWARD H S,et al. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems[C]// Proceeding of the 8th Heterogeneous Computing Workshop(HCW'98),1999,4(3):1-15.

[8] KUO W H,YANG D L. Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect[J]. European journal of operational research,2006,174(2):1 184-1 190.

[责任编辑:严海琳]