软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(8):2111−2123 [doi: 10.13328/j.cnki.jos.004670]

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

基于滚动优化的虚拟云中实时任务节能调度方法*

陈 超, 朱晓敏, 陈黄科, 王 吉, 纪浩然, 包卫东

(国防科学技术大学 信息系统工程重点实验室,湖南 长沙 410073)

通讯作者: 朱晓敏, E-mail: xmzhu@nudt.edu.cn, http://www.nudt.edu.cn

摘 要: 目前,节能已成为云数据中心的研究热点.建设节能的云数据中心不仅可以减少用电消耗,而且可以提高系统的可靠性.现有的云中心节能调度算法缺乏在任务调度级别的考虑,使得任务执行效果受到较大影响.为此,首先给出了一种基于滚动优化的实时任务调度器结构,然后详细分析和构建了任务能量消耗模型.在此基础上提出了一种实时非周期任务节能调度算法EARH(energy-aware scheduling algorithm).EARH采用的滚动优化策略能够被拓展并集成其他节能调度算法.此外,提出了资源动态增加与缩减策略,用于在系统可调度性与节能两方面进行权衡.最后,通过大量的模拟实验验证了EARH的性能.与其他3种基准算法相比,其实验结果表明,EARH的调度质量优于其他算法,可有效提高系统性能.

关键词: 虚拟云;调度;节能;滚动优化

中图法分类号: TP316

中文引用格式: 陈超,朱晓敏,陈黄科,王吉,纪浩然,包卫东.基于滚动优化的虚拟云中实时任务节能调度方法.软件学报,2015,26(8):2111−2123. http://www.jos.org.cn/1000-9825/4670.htm

英文引用格式: Chen C, Zhu XM, Chen HK, Wang J, Ji HR, Bao WD. Energy-Efficient scheduling for real-time tasks by rolling-horizon optimization in virtualized clouds. Ruan Jian Xue Bao/Journal of Software, 2015,26(8):2111−2123 (in Chinese). http://www.jos.org.cn/1000-9825/4670.htm

Energy-Efficient Scheduling for Real-Time Tasks by Rolling-Horizon Optimization in Virtualized Clouds

CHEN Chao, ZHU Xiao-Min, CHEN Huang-Ke, WANG Ji, JI Hao-Ran, BAO Wei-Dong

(Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China)

Abstract: Nowadays, energy saving has become a focus in deploying clouds. Developing energy-aware cloud data centers can not only reduce power electricity cost but also improve system reliability. Existing scheduling algorithms developed for energy-aware cloud data centers are commonly lack of consideration of task level scheduling. To address this issue, this paper proposes a novel rolling-horizon scheduling architecture for real-time task scheduling, together with a detailed task energy consumption model. Based on the novel scheduling architecture, this work develops an energy-aware scheduling algorithm EARH (energy-aware scheduling algorithm) for real-time aperiodic tasks. EARH employs a rolling-horizon optimization policy and can be extended to integrate other energy-aware scheduling algorithms. Strategies for the resource scaling up and scaling down are also presented to make a good trade-off between task's schedulability and energy saving. Extensive experiments are conducted to validate the superiority of EARH by comparing it with three baselines. The results show that EARH significantly improves the scheduling quality over the others and it is suitable for real-time task scheduling in virtulized cloud data centers.

Key words: virtulized clouds; scheduling; energy-aware; rolling horizon optimization

* 基金项目: 教育部高等学校博士学科点专项科研基金(20134307110029); 湖南省自然科学基金(2015JJ3023); 西南电子电信技术研究室公开课题(2013001)

收稿时间: 2013-11-06; 定稿时间: 2014-05-21

云计算目前已经成为具有变革性的资源供给模式,在这种模式下,云服务提供商以即用即付(pay-as-you-go)的方式为用户按需提供应用、平台和基础设施资源[1].现如今,越来越多的企业和政府部门把包括商业和科学研究在内的大量应用部署到云数据中心,以减少投资和维护成本[2].

为了更好地提供云服务,大规模数据中心的数量不断增加.值得注意的是,这些数据中心消耗的能量巨大.据统计,目前数据中心服务器消耗的能量大约占全球电力总消耗的 0.5%,如果照此速度继续发展,预计到 2020 年,其能量消耗将翻四番[3].此外,系统运行过程中产生的热量需要制冷设备(如空调)进行降温调节以保证系统的可靠性,而制冷设备的投入,又导致更多的能量消耗[4].与此同时,巨大能耗开销对环境带来了极大的负面影响[5].因此,研究云数据中心的节能调度问题,减少系统能量消耗,已成为当前极具发展前景的研究方向.

在云数据中心中,为应用提供实时保证尤为必要,有时甚至是强制性的.对于实时任务而言,其有效性不仅是保证运行结果的正确性,而且还要保证在某个时刻之前完成.例如,当地震发生后,云数据中心对灾区图像进行处理就要求提供实时保证.这是因为当灾害发生后,需要及时评估毁坏情况和制定救援计划,这就要求图像必须在几个小时甚至十几分钟之内处理完毕以赢得救援时间.如果不能在期望的时间内得到结果,将极大地影响毁坏情况的评估和救援工作的开展,进而造成严重的后果.在这种情况下,保证任务实时响应比系统的节能更为重要.因此,云数据中心应开启更多的主机,以在规定的时间内完成这些实时任务;相反,如果一些任务是非紧急的,那么云数据中心应在满足用户需求的基础上尽量减少工作机器的数量,以达到减少能量消耗的目的.

另一方面,随着虚拟化技术的发展,虚拟机(VMs)可以被动态地创建、迁移和删除,这为云数据中心的管理提供了良好的可伸缩性.然而目前,虚拟化云数据中心的可伸缩性仍然存在很多挑战,这为研究更加高效的实时保证和节能方法留下了巨大的空间.

众所周知,调度是虚拟化云中心中提高任务执行效能的有效方法.但是目前,大多数节能调度算法没有充分考虑任务的实时需求和虚拟化云数据中心的能量消耗,如何在两者之间进行有效的权衡,亟待深入研究.为此,本文提出了一种虚拟化数据中心的调度结构、模型和算法,主要贡献如下:

(1) 构建了一种基于滚动优化的调度器模型;
(2) 提出了虚拟机创建、迁移和删除策略,用于动态调整虚拟云数据中心的资源,并且充分考虑了任务的实时需求和系统节能;
(3) 提出一种基于滚动优化的节能调度策略 EARH(energy-aware scheduling algorithm),用于调度虚拟化云数据中心的实时任务;
(4) 进行了大量的模拟实验,评估了 EARH 算法的性能.

本文第 1 节回顾与本文相关的研究工作.第 2 节建立一种基于滚动优化的节能调度模型.第 3 节介绍本文提出的 EARH 算法,并对该算法进行分析.第 4 节通过大量的模拟实验测试 EARH 算法的性能.第 5 节总结全文并给出下一步的研究设想.

1 相关工作

目前,分布式系统中的节能问题得到了学者们的广泛关注,很多学者在这方面进行了深入研究.例如,Zhu 等人提出了多处理器系统中空闲资源共享的概念以减少能源消耗,并且针对多处理器系统中有任务顺序约束和无任务顺序约束的任务集分别研究了空闲资源共享的任务分配算法[6];Rountree 等人研究了一种线性规划模型,运用 DVFS 技术来产生近似最优调度方案,模型通过考虑允许的延时、通信效率和内存对已有调度方案进行节能优化[7];Yu 等人将任务分配问题看成是可扩展的广义分配问题,并提出了一种非实时节能分配策略[8].虽然前面提到的算法在节能方面都有较好的效果,但都属于静态策略,不适用于动态环境,如云中任务到达时间不确定这一情况.

此外,也有学者提出了减少能耗的动态调度算法.例如,Chase 等人考虑了网络数据中心同构资源的能效管理问题,所提出的算法通过将空闲服务器转成节能模式来减少能源消耗,同时,该算法适用于数据中心层次上的节能调度[9];Zikos 等人针对集群环境下服务时间未知的密集型任务,提出了一种节能调度算法[10];Ge 等人研究

了分布式系统中以性能为导向的 DVFS 调度策略,通过设置不同的调度粒度,可有效节约能源而且不会增加执行时间[11].但这些算法均未考虑系统资源虚拟化技术,未能有效提升虚拟化云数据中心的资源利用率.

虚拟化技术使得云计算能够为每一个用户灵活地提供资源,并使用户与其他用户相互隔离以满足安全性和稳定性要求[12].因此,大多数数据中心都采用虚拟化技术管理资源.相应地,有学者针对虚拟化云数据中心设计了高能效的调度算法.Liu 等人提出了支持虚拟机迁移和虚拟机放置优化的调度策略以减少虚拟化数据中心能耗.该算法还能实现系统更高的自治程度[13].Petrucci 等人在资源整合中运用了虚拟化技术,并提出了一种动态调度算法.该算法考虑了开启和关闭服务器的费用消耗以优化对虚拟化服务器集群的能耗管理[14].Bi 等人针对基于集群的虚拟化多层级应用提出了一种动态调度策略.此方法运用一个混合队列模型来确定不同层级所需的虚拟机数量[15].Verma 等人提出了一种虚拟化异构系统环境下的持续优化策略,在每一时间窗中都进行虚拟机部署优化,以实现最小能耗和最好性能[16].Beloglazov 等人提出了启发式虚拟机动态配置策略.该方法基于实时资源利用率进行虚拟机迁移并将空闲节点转入睡眠模式[17].Goiri 等人提出了一种多因素能效策略,对虚拟化数据中心进行管理,其中,虚拟机的部署方案综合考虑了多个方面的因素,并且追求服务提供商的利益最优化[18].Wang 等人研究了一种脱离模型约束、适应性良好的策略,能够实现变化时间长度负载情形下任务分配和能耗管理.算法中还考虑了异构多层级应用和包括吞吐率、拒绝任务数、队列状态在内的多重因素,设计了资源适应机制[19].本文中,我们研究了一种动态能效调度算法,其中运用了滚动优化方法来确保实时任务的可调度性,同时还通过虚拟机动态整合来减小云数据中心的能耗.

2 数学模型

2.1 调度器模型

本文研究的虚拟云中心可描述为一个物理主机的无限集合 $H=\{h_1,h_2,\ldots\}$,物理主机提供创建满足用户需求的虚拟资源的硬件平台.用有 n 个元素的集合 H_a 表示活跃的主机,$H_a \subseteq H$.任意一台主机 h_k 可描述为用每秒百万条指令(MIPS)定义的 CPU 性能、一定数量的内存、网络带宽,即 $h_k=\{c_k,r_k,n_k\}$,这里 c_k,r_k 和 n_k 分别表示第 k 台主机的 CPU 能力、内存和网络带宽.每台主机都可以容纳一个虚拟机集合 $V_k=\{v_{1k},v_{2k},\ldots,v_{|V_k|k}\}$.对于每台虚拟机 v_{jk},我们分别使用 $c(v_{jk})$,$r(v_{jk})$和 $n(v_{jk})$表示分配给该虚拟机的 CPU 能力、内存和网络带宽.并且根据系统负载,多台虚拟机可以在一台主机上动态地开始和停止.与此同时,虚拟机还能在主机之间进行动态迁移,这样可以合并资源并进一步减少能源消耗.图 1 给出了基于滚动窗口的调度模型.

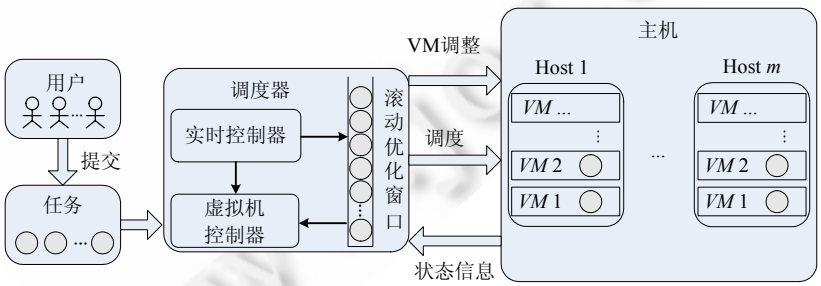


Fig.1 Model of scheduler

图 1 调度器模型

调度器的主要部件为一个滚动优化窗口、实时控制器和虚拟机控制器.调度器把用户的任务分配到不同的虚拟机上.滚动优化窗口容纳着新到达的任务和正在等待执行的任务.当一个新任务到达时就触发一个调度进程,把新到达任务和滚动窗口中所有的等待任务重新调度.

当一个新任务到达时,调度过程包含以下 5 个步骤:

步骤 1. 调度器收集系统状态信息,比如正在执行的任务的剩余时间、活跃的主机、虚拟机的配置和滚动

优化窗口中任务的信息,包括其截止期等.

步骤 2. 对滚动优化窗口中的任务,根据其截止期进行非降序排列来简化调度活动.

步骤 3. 实时控制器确定滚动优化窗口中的任务是否可以在其截止期内完成.如果正在运行的虚拟不能在截止期内完成任务,则虚拟机控制器将通过增加新虚拟机在截止期内完成任务;如果增加虚拟都不能满足任务的时间需求,则系统将拒绝该任务.否则,任务将会在滚动窗口中等待执行.

步骤 4. 更新滚动优化窗口中的调度决策,比如任务的执行次序、开始时间、配置虚拟机和开启主机.

步骤 5. 当滚动优化窗口中的任务准备就绪时,调度器把任务发送到指定的虚拟机上.

另外,当任务的到达率下降、任务有宽松的截止期和任务量减少时,系统的负载变轻,虚拟机控制器根据活跃主机的状态和任务信息决定是否停止某些虚拟机,或者通过动态迁移合并资源,达到节能的目的.

实时控制器和虚拟机控制器协同工作,首先满足用户的时间需求,并在此基础上,通过动态分配任务、调整虚拟机和调整主机尽量减少能量消耗.

2.2 任务模型

在本文中,我们考虑一组动态到达的独立任务集合 $T=\{t_1,t_2,\ldots\}$.一个任务 t_i 由一个用户提交,并且能够通过一个参数集表示,也就是 $t_i=\{a_i,l_i,d_i,f_i\}$,这里 a_i,l_i,d_i 和 f_i 分别表示任务 t_i 的到达时间、长度、截止期和完成时间.令 rt_{jk} 表示在主机 h_k 上的虚拟机 v_{jk} 的准备就绪时间,同样,假设 st_{ijk} 为任务 t_i 在虚拟机 v_{jk} 上的开始时间.由于虚拟机 CPU 处理能力的异构性,假设 et_{ijk} 为任务 t_i 在虚拟机 v_{jk} 上的执行时间:

$$et_{ijk}=\frac{l_i}{c(v_{jk})} \tag{1}$$

假设 ft_{ijk} 为任务 t_i 在虚拟机 v_{jk} 上的完成时间,可由下式计算:

$$ft_{ijk}=st_{ijk}+et_{ijk} \tag{2}$$

另外,我们使用 x_{ijk} 表示任务与虚拟机之间的映射关系,如果任务 t_i 分配到虚拟机 v_{jk} 上,则 $x_{ijk}=1$;否则,$x_{ijk}=0$.任务的结束时间反过来决定任务的截止期约束是否得到满足,即

$$x_{ijk}=\begin{cases}0, & \text{if } ft_{ijk}>d_i\\ 1 \text{ or } 0, & \text{if } ft_{ijk}\leqslant d_i\end{cases} \tag{3}$$

2.3 能量消耗模型

虚拟云中心主机的能量消耗主要由 CPU、内存、磁盘存储器和网络接口决定,其中以 CPU 的消耗为主,因此,本文主要研究 CPU 的能耗,能量消耗模型类似于文献[17].更进一步地,能量消耗可以分为两部分,即动态能量消耗和静态(泄露)能量消耗[20].由于动态能量消耗是主要部分,与此同时,静态能量消耗的趋势类似于动态能量消耗,因此,本文在建立能量消耗模型时主要研究动态能量消耗.令 ec_{ijk} 为任务 t_i 在虚拟机 v_{jk} 上的能量消耗.我们以 ecr_{jk} 表示虚拟机 v_{jk} 的能量消耗率,该能量消耗可以根据以下公式计算:

$$ec_{ijk}=ecr_{jk}\cdot et_{ijk} \tag{4}$$

因此,执行所有任务的整体能量消耗为

$$ec^{exec}=\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}\sum_{i=1}^{|T|}x_{ijk}\cdot ec_{ijk}=\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}\sum_{i=1}^{|T|}x_{ijk}\cdot ecr_{jk}\cdot et_{ijk} \tag{5}$$

其中,公式(5)中假设虚拟机空闲时没有能量消耗.然而,这种假设与现实的虚拟云数据中心不相符.当虚拟机空闲时,能量消耗可分为两种情况,也就是说,主机上所有的虚拟机都是空闲和主机上的一部分虚拟机空闲.

当主机上所有的虚拟机都空闲时,通过(DVFS)技术将该主机的电压设为最低值.因此在这种情况下,我们假设虚拟机 v_{jk} 的能量消耗率为 ecr'_{jk}.令 it_k 表示主机 h_k 上所有虚拟机都空闲的时间,那么这段时间内,该主机的能量消耗为

$$ec^{allIdle}=\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}ecr'_{jk}\cdot it_k \tag{6}$$

如果主机上只有一部分虚拟机空闲,则空闲虚拟机的能量消耗率与它们执行任务时的能量消耗一样,也就是说,虚拟机的能量消耗率为 ecr_{jk}.然后,我们得到这段时间内的能耗为

$$ec^{partIdle}=\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}ecr_{jk}\cdot t_j^{partIdle}=\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}\left(ecr_{jk}\cdot\left(\max_{i=1}^{|T|}\{ft_{ij}\}-it_k-\sum_{i=1}^{|T|}x_{ijk}\cdot et_{ijk}\right)\right) \tag{7}$$

其中,$t_j^{partIdle}$ 表示主机 h_k 上只有一部分虚拟空闲的情况下,虚拟机 v_{jk} 的空闲时间.

因此,从公式(5)~公式(7)中可得到考虑执行时间和空闲时间的能量消耗为

$$\begin{aligned}ecei&=ec^{exec}+ec^{allIdle}+ec^{partIdle}\\&=\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}\sum_{i=1}^{|T|}x_{ijk}\cdot ecr_{jk}\cdot et_{ijk}+\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}ecr'_{jk}\cdot it_k+\sum_{k=1}^{|H_a|}\sum_{j=1}^{|V_k|}\left(ecr_{jk}\cdot\left(\max_{i=1}^{|T|}\{ft_{ij}\}-it_k-\sum_{i=1}^{|T|}x_{ijk}\cdot et_{ijk}\right)\right)\end{aligned} \tag{8}$$

值得注意的是,主机资源可能没有完全使用,也就是说,即使在主机上放置了一些虚拟机,但主机还有一部分资源仍然没有使用.然而,这些没被使用资源仍然消耗能量.假设有个 S 周期,在每个周期中主机 h_k 上的虚拟机个数不同.令 t_p 表示第 p 个周期的时间,然后可得到没有被使用的资源的能量消耗:

$$ecur=\sum_{k=1}^{|H_a|}\sum_{p=1}^{s}\left(ecr(h_k)-\sum_{j=1}^{|V_k(p)|}ecr_{jk}\right)\cdot t_p \tag{9}$$

其中,$ecr(h_k)$表示主机 h_k 的能量消耗率,$|V_k(p)|$表示在第 p 周期内主机 h_k 上的虚拟机个数.

因此,从公式(8)和公式(9)中可以得到执行完所有的分配任务的总能量消耗为

$$ec=ecei+ecur \tag{10}$$

根据以上的能量消耗分析可知,活跃的主机越少,虚拟云数据中心的能量消耗就越少.然而,较少的活跃主机将影响任务的保证率.因此,任务的保证率和节能在虚拟云数据中心调度过程中属于两个冲突目标.EARH 调度算法能够很好地在任务保证率和节能之间进行权衡,这种权衡主要是根据系统负载动态开启和关闭主机以及动态创建、删除和迁移虚拟机来实现.

3 节能调度策略 EARH

本节我们首先介绍滚动优化的原理,然后把思想融合到节能调度算法中.

3.1 滚动窗口优化

在传统的调度策略中,一旦任务被调度,就把该任务发送到虚拟机或主机的本地队列中.然而在 EARH 调度算法中,所有的等待任务都放在一个滚动窗口中,等待任务的调度允许调整,这样可以满足新到达任务的可调度性和尽量降低能量消耗.滚动窗口的优势主要有:一是将动态优化问题转化为多个局部优化问题,并且充分利用已经到达任务信息和系统资源信息,实现了比单个任务直接调度方式更广范围内的优化;二是优先调度最早截止期的任务,提高系统保障实时任务时效性的能力;三是任务执行前均等待在滚动窗口中,在重调度后不产生任务的迁移开销,有利于系统节能.滚动窗口优化算法的伪代码如算法 1 所示.

算法 1. 滚动窗口优化策略伪代码.

```
1.  for each new task t_i do
2.      Q←NULL; R←NULL;
3.      Add new task t_i into set Q;
4.      Update the ready time of v_jk in Vector R;
5.      for each task t_w do
6.          if st_wjk>a_i and x_wjk==1 then
7.              Add task t_w into set Q;
8.          end if
9.          Sort tasks in Q by their deadlines in a non-descending order;
```

10. for each task t_q in set Q do
11. Schedule task t_q by **Different Energy-Efficient Scheduling Algorithm**;
12. if x_{qjk}==0 then
13. Reject task t_q;
14. end if
15. end for
16. Update scheduling decisions;
17. end for
18. end for

在滚动窗口优化策略中,当新任务到达时,把新任务加入到滚动窗口中(见算法 1 第 3 行),滚动窗口的任务集合用 Q 表示.更新虚拟机的准备就绪时间(见算法 1 第 4 行).然后,把所有等待执行的任务都加入到 Q 中(见算法 1 第 6 行~第 8 行).对 Q 中的任务,根据其截止期进行非降序排序(见算法 1 第 9 行).接着,使用不同的节能调度算法调度 Q 中的任务,如果某些任务不能分配就拒绝它们(见算法 1 第 10 行~第 15 行).

3.2 节能调度算法

在我们的节能调度算法中,新任务将被添加在已经传送到虚拟机上的任务后面.因此,任务 t_i 在虚拟机 v_{jk} 上的开始时间 st_{ijk} 可以计算为

$$st_{ijk}=\max\{rt_{jk},a_i\} \tag{11}$$

其中,rt_{jk} 表示虚拟机 v_{jk} 的准备就绪时间,每传送一个任务到虚拟机 v_{jk} 上就更新它,也就是传送新任务 t_q 到虚拟机 v_{jk} 上.虚拟机 v_{jk} 新的准备就绪时间为

$$rt_{jk}=st_{qjk}+et_{qjk} \tag{12}$$

节能调度算法的伪代码如算法 2 所示.

算法 2. 节能调度算法伪代码.

1. for each task t_i in set Q do
2. *findTag*←FALSE; *findVM*←*NULL*;
3. for each VM v_{jk} in the system do
4. Calculate the start time st_{ijk} by Eq.(11) and the execution time et_{ijk} by Eq.(1);
5. if $st_{ijk}+et_{ijk}\leqslant d_i$ then
6. *findTag*←TRUE
7. Calculate ec_{ijk} by Eq.(4);
8. end if
9. end for
10. if *findTag*==FALSE then
11. *scaleUpResource*(·);
12. end if
13. if *findTag*==TRUE then
14. Select v_{sk} with minimal energy consumption to execute t_i; *findVM*←v_{sk};
15. else
16. Reject task t_i;
17. end if
18. Update the scheduling decision of t_i and remove it from Q;
19. end for

节能调度算法采用启发式算法.该算法分配每个任务到虚拟机的过程中,在满足任务截止期的条件下尽可

能地减少能量消耗.节能调度算法首先从任务集合 Q 中选出截止期最早的任务,然后,计算任务在虚拟机上的开始和执行时间(见算法2第4行).如果任务的截止期能够得到满足,则说明任务是可以分配的,接下来计算执行任务的能耗(见算法 2 第 7 行).如果任务不能分配到现有的虚拟机上,则调用函数 *scalUpResource*(·)增加虚拟机尽量完成任务(见算法 2 第 10 行~第 12 行).如果任务能够分配,则选择最小能耗的虚拟机执行该任务;否则拒绝该任务(见算法 2 第 13 行~第 17 行).如果某些虚拟机的空闲时间超过设定的门槛,则调用函数 *scaleDownResource*(·)进行虚拟机合并,从而进一步节能.

当任务不能成功分配到现有的虚拟机上时,调用函数 *scaleUpResource*(·)创建新虚拟机,新虚拟机能够在任务截止期内完成不能分配的任务.本文使用以下 3 个步骤创建新虚拟机:

步骤 1. 在当前活跃的主机上直接创建虚拟机,不对其他虚拟机进行迁移.

步骤 2. 如果步骤 1 中不能创建新虚拟机,则通过虚拟机动态迁移把主机的空闲资源汇聚到某台主机上,然后把虚拟机创建在该主机上.

步骤 3. 如果前两个步骤都不能成功地创建新虚拟机,则开启一台主机,然后在该主机上创建新虚拟机.

$st(h_k)$,$ct(v_{jk})$和 $mt(v_{jk})$分别表示主机 h_k 的开启时间、虚拟机 v_{jk} 的创建时间和虚拟机 v_{jk} 的迁移时间.虚拟机迁移时间的定义类似文献[21],表示如下:

$$mt(v_{jk})=\frac{r(v_{jk})}{n(v_{jk})} \tag{13}$$

值得注意的是,使用不同的步骤创建新虚拟,虚拟机的准备就绪时间是不同的,也就是任务的开始时间 st_{ijk} 也是不同的:

$$st_{ijk}=\begin{cases} a_i+ct(v_{jk}), & \text{if Step 1} \\ a_i+ct(v_{jk})+\sum_{p=1}^{|P|}mt(v_{pk}), & \text{if Step 2} \\ a_i+st(h_k)+ct(v_{jk}), & \text{if Step 3} \end{cases} \tag{14}$$

增加资源的伪代码(函数 *scaleUpResource*(·))如算法 3 所示.

算法 3. 增加资源策略伪代码.

1. Select a kind of VM v_j with minimal MIPS on condition that t_i can be finished before its deadline;
2. Sort the hosts in H_a in the decreasing order of the CPU utilization;
3. for each host h_k in H_a do
4. if VM v_j can be added in host h_k then
5. Create VM v_{jk}; *findTag*←TRUE; break;
6. end if
7. end for
8. if *findTag*←FALSE then
9. Search the host h_s with minimal CPU utilization;
10. Find the VM v_{ps} with minimal MIPS in h_s;
11. for each host h_k except h_s in H_a do
12. if VM v_{ps} can be added in host h_s then
13. Migrate VM v_{ps} to host h_k; break;
14. end if
15. end for
16. if VM v_j can be added in host h_s then
17. Create VM v_{js};
18. if t_i can be finished in v_{js} before its deadline then

19. *findTag*←TRUE;
20. end if
21. end if
22. end if
23. if *findTag*==FALSE then
24. Start a host h_n and put it in H_a;
25. Create VM v_{jn} on h_n;
26. if t_i can be finished in v_{jn} before its deadline then
27. *findTag*←TRUE;
28. end if
29. end if

在函数 *scaleUpResource*(·)中,我们的算法首先选择一种能够在任务截止期内完成任务的虚拟机 v_j(见算法 3 第 1 行),然后选择一个剩余 MIPS 尽可能少的主机容纳虚拟机 v_j(见算法 3 第 2 行~第 7 行).如果找不到这样的主机,则把资源使用最少的主机上 MIPS 最小的虚拟机迁移到剩余资源尽可能少的主机上(见算法 3 第 9 行~第 15 行).当虚拟机迁移后,该算法继续检测虚拟机 v_j 是否可以创建在虚拟机迁出的主机上.如果可行,则创建虚拟机 v_j 并且检测任务 t_i 在虚拟 v_j 上是否可以在截止期内完成(见算法 3 第 16 行~第 21 行).如果不存在虚拟机迁移,或者任务 t_i 不能在截止期内完成,则开启另外一台主机,并在该主机上创建虚拟机 v_j.然后,检测任务 t_i 在虚拟机 v_j 上是否可以在截止期内完成(见算法 3 第 23 行~第 29 行).

当主机为虚拟机提供的资源多余时,虚拟机在理论上可以重新调整或者合并到尽可能少的主机上,然后关闭空闲的主机,这样可以减少能量消耗和减少虚拟云数据中心的总能耗.算法 4 给出了缩减资源的伪代码.

算法 4. 缩减资源策略伪代码.

1. SH←∅; DH←∅;
2. for each VM v_{jk} in the system do
3. if v_{jk}'s idle time>*THRESH* then
4. Remove VM v_{jk} from host h_k and delete it;
5. end if;
6. end for;
7. for each host h_k in H_a do
8. if there is no VM on h_k then
9. Shut down host h_k and remove it from H_a;
10. end if
11. Sort the hosts in H_a in the increasing order of the CPU utilization;
12. SH←H_a; DH←H_a and sort DH inversely;
13. for each host h_k in SH do
14. *shutDownTag*←TRUE; AH←∅;
15. for each VM v_{jk} in h_k do
16. *migTag*←FALSE;
17. for each host h_p in DH except h_k do
18. if v_{jk} can be added in h_p then
19. *migTag*←TRUE; AH←h_p; break;
20. end if
21. end for

22. if then
23. *DownTag*←FALSE; break;
24. end if
25. end for
26. if *migTag*==FALSE then
27. Migrate VMs in h_k to destination hosts; $SH \leftarrow SH-AH-h_k$; $DH \leftarrow DH-h_k$;
28. Shut down host h_k and remove it from H_a;
29. end if
30. end for

在算法 4 中,如果某些虚拟机的空闲时间大于设定的门槛,则关闭这些虚拟机(见算法 4 第 2 行~第 6 行).之后,如果某些主机上不存在运行的虚拟机,则关闭这些主机(见算法 4 第 7 行~第 10 行).算法 4 中,*SH* 和 *DH* 分别表示活跃主机的集合,但是 *SH* 中的主机按照 CPU 使用率是升序,而 *DH* 中的主机排序与 *SH* 相反(见算法 4 第 11 行、第 12 行).如果 *SH* 中的某台主机上所有的虚拟机都能迁移到 *DH* 中的一台或多台主机上,算法 4 将这些虚拟机都迁移到目标主机上,然后关闭迁移源主机.接着,将源主机分别从 *SH* 合 *DH* 中移除,并关闭源主机.如果 *DH* 中某台主机上的部分虚拟机而不是所有的虚拟机都可以迁移到其他主机,则放弃虚拟机迁移(见算法 4 第 13 行~第 30 行).

4 实验测试

本节通过大量模拟实验测试 EARH 调度算法的性能,将其与 non-RH-EARH(NRHEARH)调度算法、non-Migration-EARH(NMEARH)调度算法和 non-RH-Migration-EARH(NRHMEARH)调度算法进行比较.

NRHEARH,NMEARH 和 NRHMEARH 的算法描述如下:

- NRHEARH:该算法与 EARH 算法的区别在于没有采用滚动窗口调度技术.
- NMEARH:该算法与 EARH 算法的区别在于没有采用虚拟机迁移技术.
- NRHMEARH:滚动窗口技术与虚拟机迁移技术均没有在该算法中被采用.

本文主要从以下几个指标比较 EARH,NRHEARH,NMEARH 和 NRHMEARH 的性能:

(1) 完成率(guarantee ratio,简称 GR):*GR*=成功调度的任务数/全部提交任务数×100%;

(2) 平均单任务能耗(energy per task,简称 EPT):*EPT*=成功调度的任务数/总能量消耗;

(3) 总能量消耗(total energy consumption,简称 TEC):所有主机能量消耗之和.

4.1 模拟方法和参数

在本文的实验中,我们采用 CloudSim 平台模拟云计算环境.假设每台主机有一个 CPU,为体现计算能力的异构性,性能参数可以为 1 000 MIPS,1 500 MIPS 或 2 000 MIPS,相对应的最大能量消耗功率分别为 250W,300W 或 400W.开启一台主机和创建一个新的虚拟机所需时间分别为 90s 和 15s.

下面给出实验中所用的模拟参数:

(1) 参数 *taskCount* 表示一轮实验中任务的数量.

(2) 参数 *baseDeadline* 用于控制任务的截止期.任务的截止期根据公式(15)计算而得:

$$d_i=a_i+DeadlineTime \tag{15}$$

其中,*DeadlineTime* 服从均匀分布 $U(baseTime, a\times baseTime)$.在本文中,设定 a 的值为 4.

(3) 单位时间内到达的任务数服从泊松分布,参数 *intervaTime* 表示两个连续任务间的平均时间间隔.

实验参数设定见表 1.

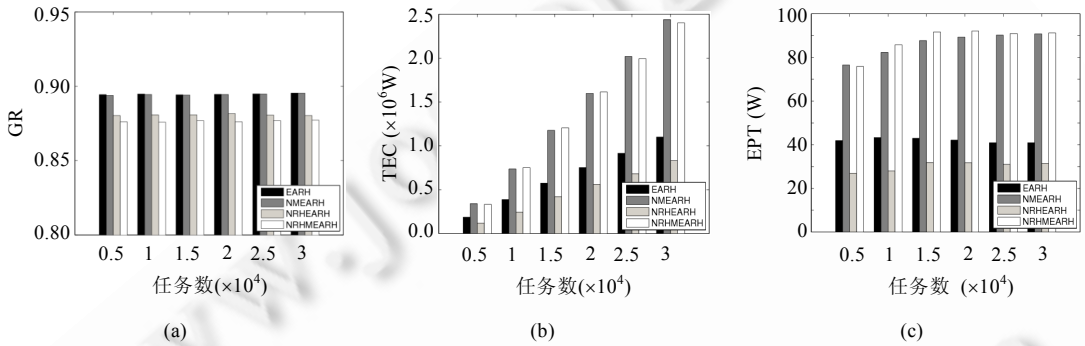
Table 1 Parameters for simulation studies

表 1 实验参数设定

参数	取值(固定值)-(变化值)
taskCount (10^4)	(1)-(0.5,1.0,1.5,2.0,2.5,3.0)
baseDeadline(s)	(250)-(150,200,250,300,350,400)
intervaTime(s)	(3)-(1,3,5,7,9,11)
Tasklength (MI)	(100000)

4.2 任务数对性能的影响

在本组实验中,为测试任务数量对调度算法性能的影响,将任务数从5 000变化到30 000.图2显示了EARH, NRHEARH,NMEARH 和 NRHMEARH 在完成率、总能量消耗和平均单任务能耗 3 个方面的性能对比.



(a) (b) (c)

Fig.2 Impact on performance caused by task count

图 2 任务数量对性能的影响

图 2(a)显示了所有策略的完成率保持稳定,几乎不随任务数的变化而变化.这是因为云可以无限量地按需提供计算资源,当任务数量增加时,将开启新的主机来满足新增任务的需求.然而,虽然有无限的资源可以按需使用,但并非所有任务都可以完成,这是因为开启主机或创建虚拟机都需要额外的时间消耗,使任务不能及时开始导致不能满足截止期要求.同时我们可以发现,EARH 和 NMEARH 的任务完成率要更高.这是由于采用了滚动窗口技术后,可以使截止期较短的任务能优先完成,有效地提高任务完成率.

从图 2(b)中可以发现:尽管 NRHEARH 可以提供最好的节能效果,但其任务完成率却较低(见图 2(a));相反,NMEARH 具有较高的能量消耗,但其任务完成率也较高.上述实验结果表明,这两种调度算法缺乏在任务完成率和能量消耗之间进行权衡的能力.此外,从图 2(b)可以看出,EARH 与 NRHEARH 与另外两种算法相比,能够明显降低总能量消耗,且这种优势随着任务数量的增加更加明显.这一结果表明,在采用虚拟机迁移技术后,一方面可以在任务数量增加时整合现有虚拟机,在已开启的主机上创建新的虚拟机,以避免新开主机增加能耗;另一方面,可以将负载较小的主机上的虚拟机迁移走,并关闭这些主机,进一步减少能耗.

图 2(c)的结果表明,NMEARH 与 NRHMEARH 的平均单任务能耗随着任务数量的增加稍有上涨,而另两种采用了虚拟机调度技术的算法基本不变.其原因在于,当任务数量增加时,要创建新的虚拟机以容纳这些任务,NMEARH 与 NRHMEARH 没有采用虚拟机迁移技术,在现有主机没有足够空间时,只能通过新开主机放置这些虚拟机,增加了单任务能耗;而 EARH 与 NRHEARH 采用了虚拟机迁移技术,可以充分利用每台开启主机的计算能力,尽量避免新开主机,使单任务能耗保持在一个稳定的值.

4.3 任务到达率对性能的影响

为了测试任务到达率对性能的影响,在本组实验中,将参数 *intervaTime* 的值从 1 变化到 11,步长为 2.图 3 显示了 EARH, NRHEARH,NMEARH 和 NRHMEARH 的性能.

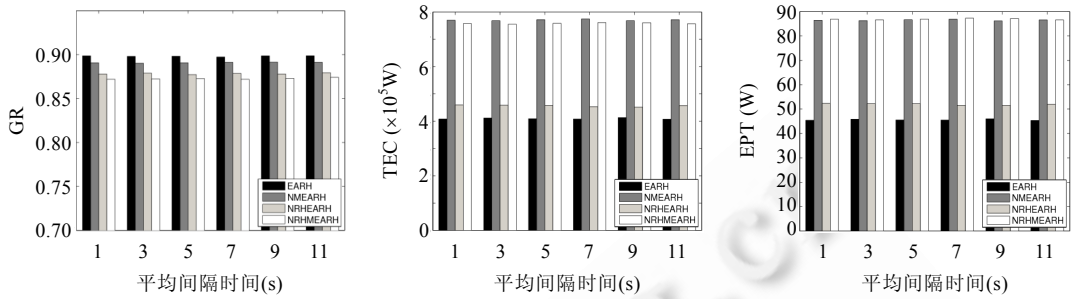


Fig.3 Impact on performance caused by task arrival rate

图 3 任务到达率对性能的影响

图 3(a)显示了当参数变化时,4 种算法的任务完成率几乎不变.这同样得益于虚拟云中无限的按需提供的计算资源,当参数 *intervaTime* 取值较小,有大量任务几乎同时到达时,系统可以通过创建新的虚拟机或开启主机来容纳这些任务.此外,采用滚动窗口技术的 EARH 与 NMEARH 的完成率要高于没有采用该技术的 NRHEARH 与 NRHMEARH,其结果与图 2(a)中所示情况的解释一致.

图 3(b)显示了 NMEARH 和 NRHMEARH 消耗的能量最多,EARH 和 NRHEARH 消耗的能量较少.其原因与图 2(b)中的解释一致.此外,从图 3(b)可以看出,当任务到达率变化时,4 种算法的能量消耗基本不变,表明任务到达率对算法的能量消耗影响不大.

图 3(c)中的结果表明:当参数变化时,4 种算法的平均单任务能耗基本不变,表明任务到达率对平均单任务能耗的影响不大.另外,EARH 和 NMEARH 的单任务能耗要明显小于 NMEARH 和 NRHMEARH,其原因与图 2(c)中的解释一致.

4.4 任务截止期对性能的影响

本组实验的目的是为了测试任务截止期对 EARH,NRHEARH,NMEARH 和 NRHMEARH 性能的影响.参数 *baseDeadline* 的变化从 150 到 400.图 4 显示了实验中 4 种方法的性能.

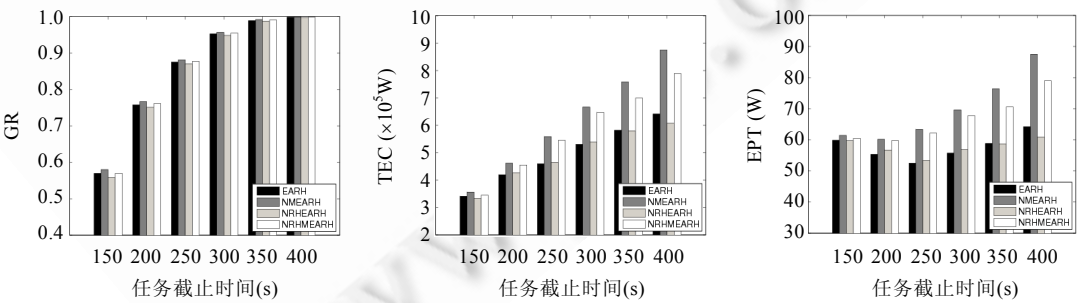


Fig.4 Impact on performance caused by task deadline

图 4 任务截止期对性能的影响

从图 4(a)中可以看出,随着 *baseDeadline* 参数取值的增大(即截止期变得宽松),EARH,NRHEARH,NMEARH 和 NRHMEARH 的完成率随之增加.这是因为任务的截止期限制变得宽松,使得任务可以较晚完成.此外,图 4(a)显示了 NMEARH 和 NRHEARH 分别具有最高和最低的完成率,NMEARH 的完成率最高.这是由于采用了滚动窗口技术后,截止期较为紧张的任务能够优先完成,相对宽松的任务能够向后推迟,提高了任务的完成率,因

此,NMEARH 与 EARH 比另外两种算法的完成率更高.同时,在截止期比较紧张时,NMEARH 由于没有采用虚拟机迁移技术,开启了更多的主机,这为后续到达的任务提供了更多的计算余量,使后续到达的截止期紧张的任务可以顺利完成,因此完成率比采用虚拟机迁移技术的 EARH 和 NRHEARH 要高.

从图 4(b)中可以看出,当 *baseDeadline* 的值增加时,4 种算法的能量消耗随之增加.这是由于当截止期变宽松时,系统能够完成更多的任务,因此需要消耗更多的能量.此外,EARH 和 NRHEARH 的能量消耗要小于另外两种算法,且这种优势随着 *baseDeadline* 参数取值的增大而不断扩大.这种现象的原因在于,EARH 和 NRHEARH 采用了虚拟机迁移的技术,能够充分利用已开启主机的计算能力,避免了为完成新增任务而开启新的主机;而 NMEARH 与 NRHMEARH 只能通过不断开启主机来适应由于截止期变宽松而新增的能够完成的任务,能耗快速增大.

图 4(c)显示了 NMEARH 与 NRHMEARH 的平均单任务能耗随着截止期变宽松不断变大,其原因在于新开启了大量的主机,更多主机上的计算能力不能得到充分利用而空闲,致使效率更低,单任务能耗不断变大.当 *baseDeadline* 取值小于 300 时,EARH 和 NRHEARH 的单任务能耗减小.这是因为当截止期变宽松后,在已开启的主机上有更多的任务可以完成;且由于采用了虚拟机迁移技术,不需要新开主机,因此对主机计算能力的利用效率变高,单任务能耗减小.而当 *baseDeadline* 取值大于 300 后,已开启的主机不能再支持由于截止期变宽松而增加的任务,必须再开启主机,这导致了部分计算能力的空闲,使单任务能耗稍有上升.

5 结 论

本文研究了虚拟云中心独立任务节能调度问题,其调度目标是在用户期望与系统节能之间进行权衡.为了达到这一目标,本文提出了一种基于滚动窗口的节能调度算法(EARH).该策略可有效提高系统灵活性,根据系统负载情况对新到达任务和滚动窗口中的任务进行资源伸缩.为了评估 EARH 算法的性能,本文通过大量模拟实验对其进行了测试.实验结果表明:在系统负载的较大变化范围内,EARH 算法与其他算法相比具有更好的性能,可以为云数据中心实时任务处理提供良好的适应性.

下一步的研究工作主要包括 3 个方面:第一,将 EARH 扩展到同时采用 DVFS 技术的节能调度策略;第二,改进现有调度策略以处理虚拟云中心中的并行任务;第三,进一步考虑调度中的通信开销.

References:

[1] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009,57(3):599−616. [doi: 10.1016/j.future.2008.12.001]

[2] Dastjerdi AV, Tabatabaei SGH, Buyya R. A dependency-aware ontology-based approach for deploying service level agreement monitoring services in cloud. Software-Practice and Experience, 2012,42(4):501−518. [doi: 10.1002/spe.1104]

[3] Younge AJ, Laszewski G, Wang L, Alarcon SL, Carithers W. Effcient resource management for cloud computing environments. In: Proc. of the IEEE Int'l Green Computing Conf. (IGCC 2010). Washington: IEEE, 2010. 357−364. [doi: 10.1109/GREENCOMP.2010.5598294]

[4] Zhu XM, He C, Li KL, Qin X. Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters. Journal of Parallel and Distributed Computing, 2012,72(6):751−763. [doi: 10.1016/j.jpdc.2012.03.005]

[5] http://do.chinabyte.com/389/12349389.shtml

[6] Zhu D, Melhem R, Childers BR. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2003,14(7):686−700. [doi: 10.1109/TPDS.2003.1214320]

[7] Rountree B, Lowenthal DK, Funk S, Freeh VW, de Supinski BR, Schulz M. Bounding energy consumption in large-scale MPI programs. In: Proc. of the ACM/IEEE Conf. on Supercomputing (SC 2007). New York: ACM Press, 2007. 1−9. [doi: 10.1145/1362622.1362688]

[8] Yu Y, Prasanna VK. Power-Aware resource allocation for independent tasks in heterogeneous real-time systems. In: Proc. of the 9th Int'l Conf. on Parallel and Distributed Systems (ICPADS 2002). Washington: IEEE, 2002. 341−348. [doi: 10.1109/ICPADS.2002.1183422]

[9] Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP. Managing energy and server resources inhosting centers. In: Proc. of the 18th ACM Symp. on Operating SystemsPrinciples. New York: ACM Press, 2001. 103−116. [doi: 10.1145/502034.502045]

[10] Zikos S, Karatza HD. Performance and energy aware cluser-level scheduling of compute-intensive jobs with unknown service times. Simulation Modelling Practice and Theory, 2011,19(1):239−250. [doi: 10.1016/j.simpat.2010.06.009]

[11] Ge R, Feng X, Cameron KW. Performance-Constrained distributed DVS scheduling for scientific applications on power-aware clusters. In: Proc. of the ACM/IEEE Conf. on Supercomputing (SC 2005). New York: ACM Press, 2005. 34−44. [doi: 10.1109/SC.2005.57]

[12] Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. Above the clouds: A Berkeley view of cloud computing. UCB/EECS-2009-28, Berkeley: EECS Department, University of California, Berkeley, 2009.

[13] Liu L, Wang H, Liu X, Jin X, He WB, Wang QB, Chen Y. GreenCloud: A new architecture for green data center. In: Proc. of the 6th Int'l Conf. on High Performance Distributed Computing (HPDC 2008). New York: ACM Press, 2008. 29−38. [doi: 10.1145/1555312.1555319]

[14] Petrucci V, Loques O, Mosse D. A dynamic configuration model for power-efficient virtualized server clusters. In: Proc. of the 11th Brazillian Workshop on Real-Time and Embedded Systems. 2009. 35−44.

[15] Bi J, Zhu ZL, Tian RX, Wang QB. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In: Proc. of the 2010 IEEE 3rd Int'l Conf. on Cloud Computing. Washington: IEEE, 2006. 15−24. [doi: 10.1109/CLOUD.2010.53]

[16] Verma A, Ahuja P, Neogi A. pMapper: Power and migration cost aware application placement in virtualized systems. In: Proc. of the 9th ACM/IFIP/USENIX Int'l Conf. on Middleawre. New York: ACM Press, 2008. 243−264. [doi: 10.1007/978-3-540-89856-6_13]

[17] Beloglazov A, Abawajy J, Buyya R. Energy-Aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 2012,28(5):755−768. [doi: 10.1016/j.future.2011.04.017]

[18] Goiri I, Berral JL, Fito JO, Julia F, Nou R, Guitart J, Gavalda R, Torre J. Energy-Efficient and multifaceted resource management for profit-driven virtualized data centers. Future Generation Computer Systems, 2012,28(5):718−731. [doi: 10.1016/j.future.2011.12.002]

[19] Wang XY, Du ZH, Chen YN. An adaptive model-free resource and power management approach for multi-tier cloud environments. The Journal of Systems and Software, 2012,85(5):1135−1146. [doi: 10.1016/j.jss.2011.12.043]

[20] Yan L, Luo J, Jha NK. Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2005,24(7):1030−1041. [doi: 10.1109/TCAD.2005.850895]

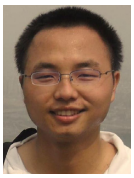
[21] Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience, 2012, 24(13):1397−1420. [doi: 10.1002/cpe.1867]

陈超(1990－),男,湖南岳阳人,硕士生,主要研究领域为云计算,移动云计算.

王吉(1990－),男,博士生,主要研究领域为云计算,实时系统.

朱晓敏(1979－),男,博士,副教授,CCF 高级会员,主要研究领域为系统优化与调度,实时系统.

纪浩然(1987－),男,博士生,主要研究领域为分布式调度.



陈黄科(1990－),男,博士生,主要研究领域为云计算资源组织与优化.

包卫东(1971－),男,博士,教授,博士生导师,主要研究领域为信息系统.