基于代价感知的云任务处理策略①

况亚萍, 王 雷, 刘小龙, 黄承真

(中国科学技术大学 自动化系网络传播系统与控制安徽省重点实验室, 合肥 230039)

摘 要: 绿色云计算旨在保证服务质量的前提下, 最大限度地降低云服务成本. 针对这一问题, 提出一种兼顾云数据中心高性能和低成本的策略, 通过虚拟机动态迁移等技术进行云任务聚合, 关闭空闲的节点以最小化能量消耗, 同时考虑虚拟机迁移性能, 以此保证任务的服务质量. 实验结果表明, 代价感知的策略可以减少打开节点的个数, 兼顾迁移能耗和迁移延迟, 在节能的前提下使服务质量得到保证.

关键词: 绿色云计算; 虚拟机; 动态迁移; 任务聚合

Cost-Aware Workload Process Strategy in Cloud Environment

KUANG Ya-Ping, WANG Lei, LIU Xiao-Long, HUANG Cheng-Zhen

(Department of Automation, USTC Key laboratory of network communication system and control, Hefei 230027, China)

Abstract: Green cloud computing aims at reducing the cost of cloud services in the premise of guaranteeing the quality of service(QoS). To solve this problem, we make a trade-off between migration performance and energy consumption. The basic idea of this paper is to concentrate workloads into a minimum set of nodes and close the idle ones to save energy. At the same time, we consider the performance of Virtual Machine migration to guarantee the QoS. The experiment result shows that this strategy can reduce the number of open node and give consideration to migration energy consumption and migration delay. So it guarantees the QoS in the premise of saving energy.

Key words: green cloud computing; virtual machine; live migration; workload concentration

近年来, 随着计算密集型和数据密集型应用的不断增多, 云计算数据中心的规模不断增大, 消耗的成本也随之不断增加. 数据显示[1], 数据中心在其生命周期中的维护与消耗的成本已经大大超过硬件购置成本. 其中, 数据中心运行时的能量消耗是最主要的维护成本. 例如, 2006 年美国的数据中心能耗约为 45 亿 kWh, 占美国电力消耗总量的 1.5%左右, 并且按每年 18%的趋势持续增加[2]. 除此之外, 大量的 CO_2 排放加剧了温室效应. 因此, 绿色云计算逐渐成为当今最受关注的问题之一. 常见的减少数据中心能耗的方法是使用虚拟机动态迁移技术. 这种技术可以使得虚拟机在保持运行的情况下, 从一台主机无缝切换到另一台主机, 因此可以把虚拟机聚集到更少的主机上, 并关闭空闲的节点, 从而减少了能耗. 近年来出现的云计算模式多为通过互联网提供的一种 pay-as-you-go 的服务方式[3]. 因此, 保证服务质量也是云计算的一个主要目标. 云提供商有责任为用户提供可靠的服务质量. 服务质量常用服务等级协议(SLA)的形式表示, 包括吞吐量, 响应时间等方面. 因此, 在任务聚合的同时, 要尽量任务的执行性能不受影响, 这就需要一个任务性能和能耗的折中.

为解决以上问题, 提出一种云环境下代价感知的任务聚合与迁移策略. 主要包括以下内容:

- 用虚拟机封装任务, 并使用虚拟机动态迁移技术实现任务的迁移和聚合.
- 提出了一种启发式策略, 对任务的一些动态事件做出响应, 得到最优的任务放置策略. 任务动态事件包括任务到达, 任务结束, 任务调整大小等.
- 在策略中加入代价感知模块, 在保证任务最大程度聚合的同时, 使任务迁移的代价最小, 即迁移性能最优, 从而保证任务的服务质量.

① 基金项目:中央高校基本科研业务费专项基金(WK2100100012)
收稿时间:2013-04-22;收到修改稿时间:2013-05-15

近年来, 有很多研究证明了虚拟机的迁移可以节省大量的能耗, 于此同时, 也增加了额外的运行时间[4]. 现存的大部分算法仅仅考虑了任务聚合带来的能量节约, 而忽略了任务迁移造成的能量消耗和迁移代价问题[5-7]. Bo Li 等人提出了一种云环境下任务动态放置策略[7], 把任务放置问题抽象为装箱问题, 并进行改进, 加入了对任务动态事件的考虑. 通过聚合任务来节约能耗. 但是忽略了迁移造成的能耗和性能损失. Anton Beloglazov 等人提出了一种动态阈值的虚拟机聚合策略[8], 以及几种虚拟机选择策略. 通过改进 Best-Fit 算法, 加入动态阈值的计算, 以适应任务的多样性.

在虚拟机的迁移性能计算方面, William 等人通过实验验证了虚拟机迁移对 Web2.0 应用造成的性能退化, 并提出了一种性能退化模型[9]. 这种方法的主要目的是注重服务质量而不是迁移性能. Bing Wei 等人提出了一种虚拟机迁移代价模型[10], 通过迁移代价来衡量迁移性能, 并且提出了 Co-migration 的现象和解决方案. 但是 Co-migration 的数量有很大限制, 要求不大于 3. Haikun Liu 等人也提出了一种虚拟机动态迁移的性能模型[11], 并且对不同因素造成的性能损失进行了研究. 但是, 很少有研究把迁移能耗和迁移性能综合考虑, 在任务聚合的同时, 保证任务的服务质量. 本文剩余部分的结构如下所示: 第 2 节对问题进行了进一步陈述, 并提出了系统模式架构. 第 3 节详细地介绍了算法细节. 第 4 节中, 对本文策略进行了实验, 并对实验结果进行分析. 最后, 在第 5 节进行了总结.

1 问题描述和系统模型

1.1 问题描述

云计算环境中的节点分为计算节点和存储节点. 本文假设所有的计算节点是同构的. 节点之间使用高速局域网(LAN)互连, 因此网络传输延迟可以忽略. 云任务被提交和分配后, 被封转在虚拟机(Virtual Machine, 以下简称为 VM)中进行处理. 任务处理完毕后, 释放 VM 资源. 每个计算节点上可同时运行多个 VM, 由虚拟机监视器(Virtual Machine Monitor, 以下简称为 VMM)统一管理. 当节点上无任务运行时, 定义该节点为“关闭”状态, 否则为“打开”状态.

在云环境下, 任务被提交时是动态分配的, 并不确定被分配到哪一个节点上执行. 当任务完成后, 需要重新建立 VM 和节点之间的对应关系. 为了节约能耗, 我们主要考虑如何最大化每个节点的资源利用率 , 最小化打开节点的个数.

另一方面, 云任务在其生存周期中具有动态性, VM 迁移不可避免. 然而, VM 迁移会带来一定的能耗, 并且影响任务的执行性能. 如何保证云服务质量是不可忽略的问题. 本文中引入迁移代价的概念, 量化了 VM 迁移中的能耗和性能损耗.

综上所述, 本文的目标是使用最少的节点来聚合云任务, 同时尽可能使任务迁移的代价最小. 这里的代价包括两个方面: 迁移的能量消耗和迁移的性能消耗.

因此, 抽象出一个最优化模型:

$$Min(n, m\_\cos t) \quad (1)$$

其中, n 表示“打开”节点的个数, $m\_\cos t$ 表示迁移代价. 该模型说明, 本文的主要目标是在最小化打开节点数目的同时, 使迁移代价最小.

1.2 系统模型

由以上的假设和分析, 系统设计为三个组成部分: 事件触发器(EI), 集中管理器(CM)和资源池(RP), 如图 1 所示. 事件触发器的主要作用是响应各种系统事件. 把系统事件分为为三种: 任务到来, 任务结束和任务量调整. 事件触发器接收事件并通知集中管理器对事件进行响应. 集中管理器是系统中最重要的模块, 它运行能量感知的启发式算法, 对不同的事件进行处理, 得到一个任务放置序列模式, 并派发给每个节点. 每个节点中运行 0 个或者多个 VM, 它们由一个虚拟机监视器(VMM)管理. VMM 根据新的放置策略对本节点上的 VM 进行管理. 此外, VMM 还负责监控本主机上的资源使用情况, 把 VM 提出的任务调整大小请求发送给事件触发器进行处理.

(1) VM 动态迁移

VM 动态迁移技术使得 VM 可以在源主机和目标主机之间“无缝”地迁移. VM 在源主机内运行的同时, 内存页被不断地拷贝到目标主机, 如图 2 所示. 在这个过程不可避免地产生内存脏页. 脏页以迭代的方式拷贝到目标主机来保证内存页的一致性. 迭代结束的条件有多种: (1)未传输的内存页小于一定的阈值; (2)迭代次数小于一定阈值; (3)脏页率超过网络传输速率; (4)网络传输数据量过大, 超过原始内存的许多倍. 迭代结束后, VM 会经历一个短暂的 stop-and-copy 过程来传输剩余脏页, 并在目标主机上启动 VM, 从源主机删除 VM.

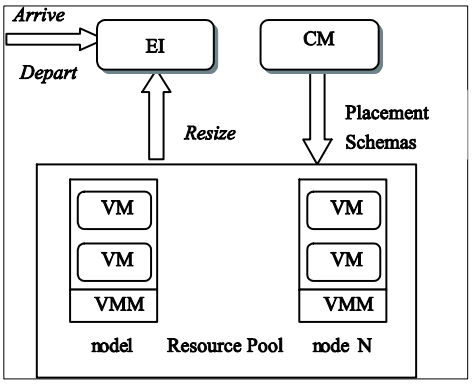

图 1　系统模型

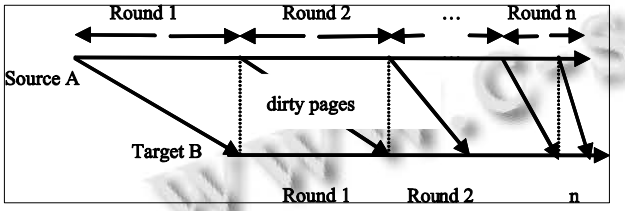

图 2　虚拟机动态迁移 pre-copy 过程

可以看到，VM 动态迁移技术本质上是一个 pre-copy 过程，它主要包括三个阶段:

● Select:确定目标主机和待迁移的 VM

● Pre-copy: 迭代地把前一轮产生的脏页拷贝到目标主机

● Stop-and-copy: 传输剩余脏页和 VM 状态，在目标机器上启动 VM，从源主机上删除 VM

在这三个步骤中，pre-copy 过程消耗代价最大. 而 stop-and-copy 过程中，由于剩余脏页很少，而且节点直接的传输带宽可视为无限，所以这一阶段的停顿时间可以忽略不计. 因此，本文主要考虑 pre-copy 过程.

因此，在本文中 VM 的模型表示为以下形式: $VM=(V_0,d,T_{th})$，其中 V_0 为 VM 的占用的初始内存大小，它代表了在 VM 中运行的任务的资源请求量. d 为脏页率，大小与任务特性有关. T_{th} 为此 VM 可以忍受的迁移延迟上限，它反映了在这个 VM 中运行的任务的服务等级，T_{th} 小的 VM 的服务等级较高. 若 T_{th} 接近无穷大时，表示无服务等级限制. 这时，迁移过程主要由 d 和 V_0 决定.

(2) 迁移代价模型

本文考虑的代价不仅仅是虚拟机迁移的能量消耗，而且包括迁移的性能. 迁移性能通过量化迁移延迟表示.

由上面的分析知，在迁移过程中，主要影响迁移延迟的因素有: VM 的原始内存页大小 V_0、内存脏页率 d、网络传输速率 R，迭代次数阈值为 n，取 R=100. 并使用如下公式(2)(3)(4)来量化传输的内存总大小和迁移延迟[7]:

$$V=V_0\cdot\frac{1-\lambda^{n+1}}{1-\lambda} \tag{2}$$

$$T=\frac{V}{R}=\frac{V_0}{R}\cdot\frac{1-\lambda^{n+1}}{1-\lambda} \tag{3}$$

$$\lambda=\frac{d}{R} \tag{4}$$

研究表明，迁移能耗与传输的内存大小近似成线性相关[12]. 为了保证服务质量，假设迁移性能与迁移延迟/延迟阈值成指数关系. 因此，使用以下的代价模型:

$$m\_\cos t(VM)=\alpha V\exp(\frac{\beta}{1-\frac{T}{T_{th}}})+\gamma \tag{5}$$

其中，*m_cost* 表示 VM 迁移的代价. V 表示在迁移过程中共传输的内存大小. T 表示迁移延迟，T_{th} 表示迁移延迟的上限. α,β,γ 为训练参数.

(3) 节点模型

本文把资源池表示为节点的集合形式 $S=(S_1,S_2,...,S_N)$，资源池共有 N 个节点. 每一个节点 S_i 上运行一个或者多个 VM. 每个节点有三种状态: 当有 VM 运行时为“打开”状态，无可用资源时为“满载”状态，无 VM 运行时为“关闭”状态. 每一个节点表示为 $S_i=(VM_1,VM_2,...,VM_{vi}),\ i\in[1,N]$.

2　代价感知的云任务放置策略

在之前的研究工作中，VM 的放置问题多被抽象为一个装箱问题. 经典的装箱问题的解法有 First-Fit, Best-Fit, Next-Fit, First-Fit-Decreasing 等. 但是这些传统的方法在处理过程中会在节点上产生大量的资源差额，造成资源浪费. 主要原因是，在装箱问题中，物体一旦被装入箱子就不会再移动，而在 VM 放置问题中，VM 会频繁而且随机地迁移或者结束离开. 为了解决这一问题，对传统的装箱算法进行了改进，应对不同的动态事件的发生. 同时，尽量保证迁移代价的最小化.

2.1 资源预处理

为了防止过度迁移，把每个节点的资源大小(0,1]分为 2M-2 个区间[13]，每一个区间代表一个资源等级.

$$L_0 = ((M-1)/M, 1]$$
$$L_1 = ((M-2)/(M-1), (M-1)/M]$$
...
$$L_{M-1} = (1/3, 1/2]$$
...
$$L_{2M-4} = (1/M, 1/(M-1)]$$
$$L_{2M-3} = (0, 1/M]$$

取 M=4, 每个节点可划分为以下六个子区间:

$$(0,\frac{1}{4}],(\frac{1}{4},\frac{1}{3}],(\frac{1}{3},\frac{1}{2}],(\frac{1}{2},\frac{2}{3}],(\frac{2}{3},\frac{3}{4}],(\frac{3}{4},1]$$

定义 L_{k+1} 比 L_k 等级高, 并且认为, 较小的 VM 比较大的 VM 更容易放置. 因此 VM 只能被比它等级高的 VM 替换.

2.2 算法细节

用 *S_NF* 表示未满载的节点列表, 由以上的分析可知, 系统动态事件主要分为三类: Arrive, Depart, Resize. 下面介绍针对这三种事件的算法.

(1) *Arrive* 事件

当一个新任务到达时, 我们的原则是尽量把它插入到已经处于“打开”状态的节点上, 避免打开一个新的节点. 本文采用的是一种启发式算法. 我们认为资源较小的任务更容易被插入, 所以不同于 BF 等普通的装箱算法, 我们首先尝试把新到来的任务用任务量更小的任务来替换. 若插入后, 节点满载或超载, 则使用下面的 Cost-aware Select 算法(以下简称 CA 选择算法)选择需要迁移的虚拟机, 来保证迁移的代价最小, 然后把这些待迁移的虚拟机当作新到来的任务重新插入到主机列表中.

表 1 arrive 算法

Procedure：Arrive
Input: new arriving VM *x*
Output: a placement schema
1. if *level(x)*=0 or *level(x)*=5
2. insert *x* into server *s* with Best-Fit
3. returns
4. foreach server *n* in *S_NF*
5. filter out all VMs *v* which *level(v)<level(x)*
6. insert *x* into server *s* with First-Fit
7. if *s* is over-load
8. select VMs to migrate using Cost-guided Select
9. insert migrating VMs as new arriving VMs

(2) *Depart* 事件

当有任务完成时, VM 释放在节点上占用的资源. 若一个节点上所有的 VM 都完成释放, 则此节点的状态由“打开”转换为“关闭”. VMM 监督节点上 VM 的运行状态, 在必要时进行更新.

表 2 Depart 算法

Procedure：Depart
Input: VM *v* to depart
Output: a placement schema
1. get server *s* of *v*
2. remove *v* from the VM list of *s*
3. update *s*
4. if *s* is empty
5. closes

(3) *Resize* 事件

在云环境下, 任务的 Resize 事件频繁而且随机地发生. 由于其频繁性, 原则上应尽量减少由 Resize 事件造成的迁移. 按照这一原则, 把大小发生变化的云任务当做一个新到来的任务进行处理, 从源节点删除, 插入到新的节点中, 这在本质上是 VM 的迁移.

表 3 Resize 算法

Procedure：Resize
Input: old size *x* of VM v,new size *y*
Output: a migration schema
1. delete v with size x using Depart procedure
2. insert new sized v using Arrive procedure

(4) *Cost-aware Select* 算法

当节点能力超载时, 需要进行任务迁移. VM 迁移会产生额外的能量消耗并影响任务的执行性能. 本文把 VM 迁移过程中的能耗和迁移性能量化为 m_cost, 由公式(5)得到, 用迁移代价来决定待迁移的 VM.

如表 4 所示, 当需要迁移时, 首先计算此节点中所有 VM 的迁移代价 m_cost, 并按照非递增顺序排列. 从队列中删除第一个 VM 并更新节点资源, 直到节点不再超载为止.

表 4 Cost-aware Select 算法

Procedure：Cost-aware Select
Input: a full-loaded server *s* ,the VM list *list1* on s
Output: VM migrating list *vm_mig*
1. foreach VM v in *list1*
2. compute *m_cost(v)*
3. reorder list1 in ascending order by *m_cost(v)*
4. add the first VM v1 into *vm_mig*
5. remove v1 from *list1*
6. end if *s* is under-load
7. else go to step 4

3 实验

3.1 实验设置

为了验证任务聚合和迁移策略的有效性，本文基于 Cloudsim 3.0 模拟了真实的云计算环境，并进行了一系列的仿真实验.

实验设定了 20、40、100 三种任务规模，在实验过程中随机产生任务，并以一定的速率提交给资源池. 每一个任务的大小是(0,1]之间的一个随机值. 根据任务特点的不同，每个任务有不同的内存脏页率 d 和迁移延迟上限 T_{th}.

3.2 实验结果

实验组 1: 本组实验的目的是验证 Cost-aware Select(CA)算法是否能够达到很好的任务聚合效果. 实验中选择如下任务规模: 10、50、100、200、500、1000，并将实验结果与传统的 Best-Fit 算法在打开节点数目和资源利用率方面进行了比较，如图 3 所示.

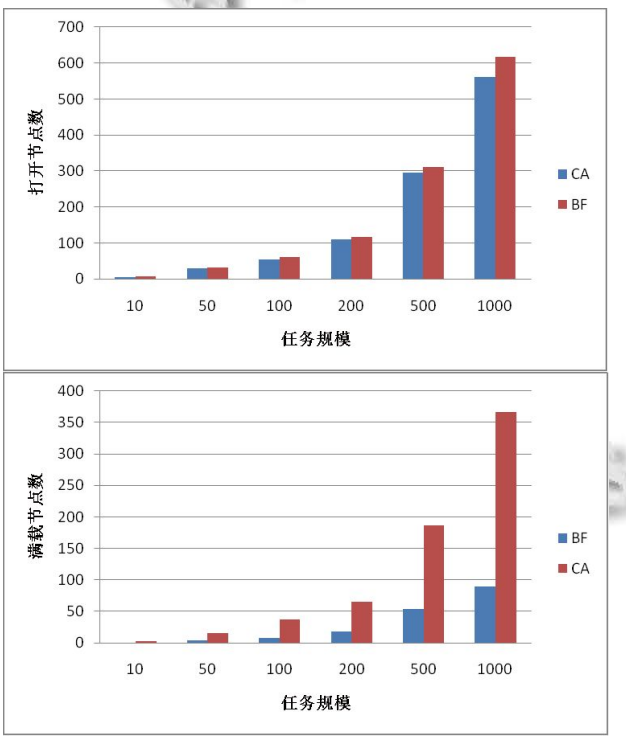

图 3　在不同的任务规模下, CA 算法与 Best-Fit 算法在打开节点个数和满载节点个数上的对比

从图 3(a)可以看到, CA 算法相比 Best-Fit 算法，打开的节点更少. 而且效果随着任务量规模的增大有更加明显的趋势. 不仅如此，从图 3(b)可以看出，CA 算法的满载节点数远远大于 Best-Fit 算法. 这说明了资源池的资源利用率得到了很大的提高. 这种现象的原因在于，CA 算法充分利用了虚拟机迁移来聚合任务，随时使任务处于一种紧密聚合的状态. 从而提高了物理节点的资源利用率，即提高了能效.

实验组 2: 本组实验的目的是验证 CA 算法中迁移选择策略的有效性(即迁移代价是否更低)，并与随机选择策略进行比较. 随机选择策略的含义是，当有节点处于“超载”状态时，随机选择一个虚拟机进行迁移. 在任务规模为 10、50、100、500、1000 这 5 种情况下，分别记录了两种方法的迁移代价. 实验结果如图 4 所示.

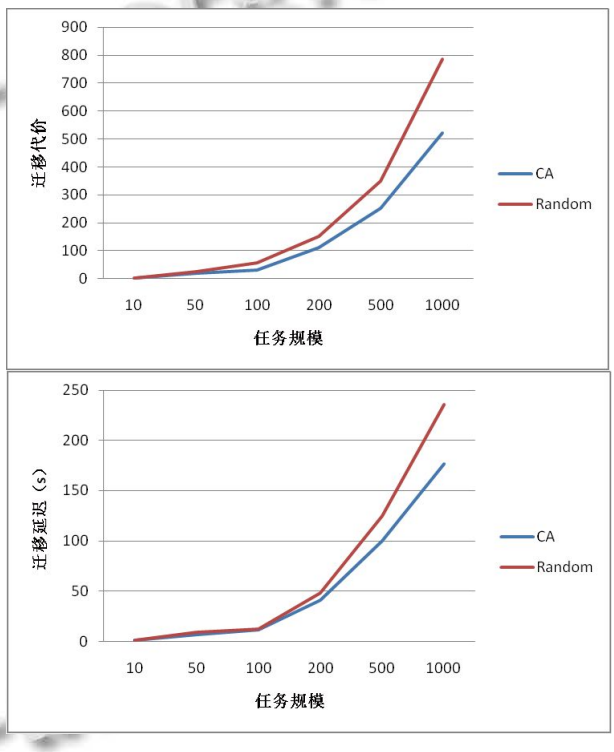

图 4　在不同的任务规模下, CA 选择策略与 Random 选择策略在迁移代价和迁移延迟上的对比

从图 4 可以看出，本文算法不论是在迁移延迟或者迁移代价方面，都优于随机选择策略. 这种现象的原因在于，CA 算法中虚拟机选择策略是代价感知的. 每次需要迁移时，优先选择代价小的任务. 实验发现，在迁移时，往往选择资源需求较大的以及迁移延迟上限较大的任务，读任务与写任务相比，读任务迁移的概率更大. 这种现象与本文假设相符，证明了算法的有效性.

综上所述，实验证明 CA 算法能够在一定程度上优化任务聚合，提高能量效率，同时减少迁移代价.

而且，从数据趋势可以看出，任务规模越大，算法的效果越好.

4 结语

为了解决云计算环境下的节能问题，提出了一种代价感知的任务聚合和迁移策略. 首先通过改进装箱算法来最大化地聚合任务，减少打开节点的个数，节约能耗. 其次加入了一种代价感知的虚拟机选择策略，同时考虑迁移能耗和迁移性能. 最后通过实验验证了提出的策略不仅可以减少了打开节点的个数，而且能够兼顾迁移能耗和迁移延迟，在节能的前提下使服务质量得到保证.

本文的算法还有一些不足. 首先，在任务资源请求方面，为了简化算法，仅设定为内存需求量一种，而在实际环境中，资源请求量是一个多维向量，可以使用欧式距离表示资源差异. 其次，本文假设所有的节点是同构的，而云数据中心可能是由类型和能力不同的服务器甚至普通 PC 机组成，对任务的分配会产生影响. 除此之外，还应对任务的多样性加以考虑，不同任务不同处理方式，这些都将是未来的工作.

参考文献

1 Koomey J. Worldwide electricity used in data centers. Environmental Research Letters 3, IOP Publishing Ltd., 2008.

2 Report to congress on server and data center energy efficiency, U.S. Environmental Protection Agency ENERGY STAR Program. August 2, 2007.

3 Buyya R, Yeo CS. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. Proc. of PCC'08. IEEE. 2008.

4 Balifa J, Ayre RWA, Hinton K. Green cloud computing: Balancing energy in processing, storage, and transport. Proc. of IEEE'11. 2011. 149–167.

5 Chen Y, Wo T, Li JX. An efficient resource management system for online virtual cluster provision. IEEE International Conference on Cloud Computing. 2009: 72–79.

6 Ye KJ, Huang DW, Jiang XH, Chen HJ, Wu S. Virtual machine based energy-efficient data center architecture for cloud computing: A performance perspective. Proc. of IEEE/ACM'10. 2010. 171-–178.

7 Li B, Li JX, Huai JP, Wo TY, Li Q, Zhong L. EnaCloud: An energy-saving application live placement approach for cloud computing environments. Proc. of IEEE International Conference on Cloud Computing. 2009. 17–24.

8 Beloglazov A, Buyya R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in Ccloud data centers. Proc. of MGC'10. Bangalore, India, 2010.

9 Voorsluys W, Broberg J, Venugopal S, Buyya R. Cost of virtual machine live migration in clouds: A performance evaluation. Proc. of the 1st International Conference on Cloud Computing, Lecture Notes In Computer Science. Beijing, China. December, 2009.254–265.

10 Wei B, Lin C, Kong XZ. Energy optimized modeling for live migration in virtual data center. 2011 International Conference on Computer Science and Network Technology. 2011. 24–26.

11 Liu HK, Xu CZ, Jin H, Gong JY, Liao XF. Performance and energy modeling for live migration of virtual machines. HPDC'11. San Jose. California, USA. 2011. 171–181.

12 Wang XL, Liu ZH. An energy-aware VMs placement algorithm in cloud computing environment. 2012 International Conference on Intelligent Systems Design and Engineering Application. 2012. 627–630.

13 Lee CC, Lee DT. A simple online binpacking algorithm. Journal of the ACM, 1985, 32(3): 562–572.