云计算中基于能耗比例模型的虚拟机调度算法

肖 鹏,刘洞波,屈喜龙
(湖南工程学院计算机与通信系,湖南湘潭 411104)

摘 要: 针对资源虚拟化环境中的混合型负载调度问题,提出一种基于能耗比例模型的虚拟机调度算法.该算法利用处理器的"性能计数器"机制来评估各个虚拟机的近期能耗状态,并采用"最近最小能耗比例优先"的策略进行调度.理论分析给出了该算法的有效性证明和相关特性.实验结果显示,当系统面对混合型负载时,基于能耗比例模型的调度算法在"调度偏差"和"相对能效"两方面明显优于现有的虚拟机调度算法.

关键词: 云计算;虚拟机;能耗;调度算法
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2015)02-0305-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.02.016

An Virtual Machine Scheduling Algorithm Based on Energy-Consumption Ratio Model in Cloud Computing

XIAO Peng, LIU Dong-bo, QU Xi-long
(*Department of Computer and Communication, Hunan Institute of Engineering, Xiangtan, Hunan* 411104, *China*)

Abstract: To address the issue of scheduling mixed-workloads in resource virtualization environments, a novel virtual machine scheduling algorithm based on energy-consumption ratio model is proposed. In the proposed algorithm, the recent energy consumption status of individual virtual machines are evaluated by using the processor's 'performance monitor counters' mechanism, and the scheduling policy is 'most recent minimal energy-consumption ratio first'. Theoretical analysis presents the validation and characteristics of the proposed algorithm. Extensive experiments show that when the system is in presence of intensive mixed-workloads, the proposed algorithm significantly outperforms existing scheduling approaches in terms of scheduling deviation and normalized energy-efficiency.

Key words: cloud computing; virtual machine; energy consumption; scheduling algorithm

1 引言

近期报告显示[1,2]:全球数据中心在 2011 年的总耗电量已达 3 兆千瓦,且以每年 11.8%的速度增长,而其资源利用率却大多小于 50%[3,4].面对高能耗与低利用率并存的现象,如何有效控制数据中心的能耗成为一个亟待解决的课题.云计算[5]以虚拟化技术为基础,其低成本和安全性的特点极大地拓展了高性能计算的应用范围[6,7].宏观而言,虚拟化技术为云系统的能效优化提供了有效的支持[8,9];微观而言,虚拟化技术所引入的额外开销也对能效优化提出了若干挑战性问题,主要体现在:

(1)虚拟化技术通过"分时复用"机制在不同应用程序之间共享物理资源,从而导致虚拟机的功耗/能耗状态难以量化评估[10,11].

(2)在面对"混合型负载"时,虚拟化平台的执行效率将大打折扣,从而导致资源层的无效能耗开销增加[12~14].

(3)在面对数据密集型任务时,虚拟化存储技术在对海量"中间数据"进行临时存储时容易出现性能瓶颈[15~17].

本文研究主要集中在资源虚拟层的执行效率和能效表现这一方面.虚拟机(Virtual Machine,VM)的调度算法一般采用"比例共享"策略[18].在面对计算密集型任务时,"比例共享"策略能够有效地落实"比例公平"原则,但面对"混合型负载"时,现有调度算法往往不能保证"比例公平"原则.更为重要的是,"比例共享"策略是以静态方式来确定虚拟机的资源分配额度,当存在不同

收稿日期:2013-11-11;修回日期:2014-02-07;责任编辑:马兰英
基金项目:国家自然科学基金(No.61402163);湖南省教育厅科学研究项目(No.13B015);湖南省自然科学基金(No.13JJ9022,No.14JJ7071);湖南省科技计划项目(No.2012GK3075)

类型的虚拟机时,虚拟机之间存在复杂的竞争关系,调度算法很难同时兼顾效率和公平[19,20].对此,本文提出一种具有能耗感知能力的虚拟机调度算法,其核心思想是以虚拟机的近期能耗状态来动态确定其调度优先级,目标是在保证"比例公平"原则的同时提高虚拟系统的能效.

2 相关研究

里昂大学的研究小组在其"绿色开放云"系统中详细测评了虚拟机在启动、执行和迁移过程中的各种能耗开销,其实验测评结果显示[21]:当 I/O 操作很密集时,频繁地"上下文切换"会导致额外的能耗开销和性能损失;设计能效感知的任务调度策略时,必须充分考虑任务自身特性.此外,Lindberg 等人[22]对多种面向能效优化的启发式调度算法进行了详尽的横向比较,其实验测评结果显示:各种启发式调度算法的能效优化结果与负载特性之间存在较强的相关性.

以上测评结果显示:密集的 I/O 访问通常会导致虚拟层执行效率降低,从而降低系统总体能效.对此,研究者从不同的方向提出了若干解决方案.例如,Auburn 大学的研究小组针对大规模并行 I/O 的高能耗问题,设计了一个名为 PRE-BUD 的系统[23],其主要特色在于:通过一个独立的中间层将并行 I/O 访问的能耗和性能瓶颈集中化,然后采用各种技术来优化该中间层的能效和性能.Kang 等人[14]提出一种"I/O 组调度策略",即调度器会等到所有组内虚拟机都发出 I/O 请求后统一进行调度.Martinez 等人[24]提出了一种针对虚拟数据服务器的能效评估模型.针对 I/O 能耗的局部性特征,李建敦等人[25]提出了一种基于虚拟磁盘布局策略的节能调度算法,其核心思想是将存储阵列划分为"工作区"和"就绪区",然后依据 I/O 请求的变化情况来动态调整两个区的尺寸.

综上,研究虚拟层能效优化的关键在于:降低调度 I/O 密集型负载的额外能耗开销,同时确保低层物理设备的有效利用率.本文提出的虚拟机调度算法,正是针对"混合型负载"中 I/O 密集型虚拟机所造成的额外能耗开销,从虚拟机调度算法层面进行相应地优化.

3 问题描述

在非虚拟化环境中,服务器功耗一般采用以下线性模型来描述

$$P = P_{\text{static}} + \sum_{j\in J} k_j U_j \tag{1}$$

其中,P_{static}为静态功耗,集合 $J=\{\text{CPU},\text{Ram},\text{Disk},\text{I/O}\}$ 表示功耗组件,U_j为组件 j 的利用率,k_j为经验系数.采用虚拟化技术后,功耗模型则通常表示为

$$P = P_{\text{static}} + \sum_{i=1}^{M} P_i^{\text{vm}} \tag{2}$$

其中,P_i^{vm}为虚拟机 V_i 的功耗,M 为活跃虚拟机数目.结合(1)和(2),虚拟机功耗模型可以描述为

$$P_i^{\text{vm}} = \frac{P_{\text{static}}}{M} + W_i \sum_{j\in J} k_j U_j \tag{3}$$

其中,W_i 表示 V_i 的 CPU"比例权重".式(3)的主要缺点是:经验系数 k_j通常采用统计方法获得,因此模型整体误差较大.此外,为落实"比例共享"原则,调度器需要满足:

$$\forall\, i,j,\ \left|\frac{U_i(t_1,t_2)}{W_i}-\frac{U_j(t_1,t_2)}{W_j}\right|=0 \tag{4}$$

其中,$U_i(t_1,t_2)$表示虚拟机 V_i 在$[t_1,t_2]$时间段内的实际 CPU 利用率.式(4)的含义是:调度器应该保证所有虚拟机的预设权重 W_i 与实际 CPU 利用率 $U_i(t_1,t_2)$保持比例公平.在实际环境中,现有调度器很难完全满足以上条件,原因在于:不同虚拟机具有不同的执行特征,调度器很难依据其的执行特征来动态地调整调度原则.因此,式(4)的一般化描述应该为

$$\forall\, i,j,\ \left|\frac{U_i(t_1,t_2)}{W_i}-\frac{U_j(t_1,t_2)}{W_j}\right|=\psi \tag{5}$$

其中,ψ 可以看作实际调度结果与预设调度原则之间的偏差(下文简称调度偏差).

4 能耗感知的虚拟机调度策略

4.1 能耗比例模型

基于"资源利用率"的虚拟机功耗模型需要引入经验参数.事实上,在调度虚拟机时,只需要了解各个虚拟机的"相对指标"就可以确定其调度优先级.目前的难点在于:在不测量虚拟机能耗值的前提下,如何确定其能耗在总能耗中的比例.对此,本文提出利用"性能计数器"[26](Performance Monitor Counters,PMC)来构建虚拟机的能耗比例模型.虽然 PMC 机制的设计初衷并非针对能耗测量,但近期研究显示[26,27]:PMC 机制能够以极低地开销来精确测量各类设备的能耗.

一般而言,与具体设备相关的 PMC 事件集合可以分别表示为 $Q^{\text{cpu}},Q^{\text{mem}},Q^{\text{disk}},Q^{\text{io}}$.从硬件层面而言,PMC 事件代表着系统某些设备处于能耗活跃状态.因此,通过监控虚拟机触发的 PMC 事件以及其数量在系统总体 PMC 事件中的比例,我们就能够获得各个虚拟机的能耗在总能耗中的比例.以下公式(6)给出了基于 PMC 机制的虚拟机能耗比例模型.

$$\pi_j = \sum_{\substack{i=\{\text{cpu},\text{mem},\\ \text{disk},\text{io}\}}} \frac{\Delta Q_j^i(t_1,t_2)}{\Delta Q^i(t_1,t_2)} \tag{6}$$

其中,$\Delta Q_j^i(t_1,t_2)$表示在$[t_1,t_2]$时间段内 V_j所触发与设备 i 相关的 PMC 事件,$\Delta Q^i(t_1,t_2)$表示与设备 i 相关的

PMC 事件.因此,基于 PMC 机制的能耗比例模型是以虚拟机所触发的 PMC 事件占总 PMC 事件的比例来描述其功耗情况.以上能耗比例模型不引入经验系数,其相关信息可以通过 PMC 编程接口获取.

4.2 调度算法设计与分析

基于能耗比例模型,本文设计了一种具有能耗感知能力的虚拟机调度算法(VM Scheduling by Energy-consumption Ratio Model,VSERM),具体算法实现如下.

输入:

$(V_1, V_2, \cdots, V_n)$:虚拟机集合;

B_i:虚拟机 V_i 的 CPU 调度额度;

W_i:虚拟机 V_i 的调度权重;

π_i:虚拟机 V_i 的能耗比例;

Begin

// 初始化调度参数

1. for $i = 1$ to n do

2. $B_i := W_i$;

3. $\pi_i := 1$;

4. end for

//调度循环

5. while the processor is idling do

6. for each $V_i \in (V_1, V_2, \cdots, V_n)$do

7. $rank_i := (\pi_i - B_i)/W_i$;

8. end for

9. Sort$(V_1, V_2, \cdots, V_n)$as$\{V_{k1}, V_{k2}, \cdots, V_{kn}\}$ in ascendant order of their $rank_i$;

10. Schedule V_{k1} onto the CPU and set its CPU shares as π_{k1} B_{k1};

//更新虚拟机的调度参数

11. for $n = 2$ to n do

12. $B_{k_i} := B_{k_i} + rank_{k_1} \times W_{k_i}$;

13 $\pi_{k_i} := \pi_{k_i} + \sum_{j \in J - \{cpu\}} \frac{\Delta Q_j^{k_i}}{\Delta Q^{k_i}}$;

14. end for

15. $B_{k_1} := 0$;

16 $\pi_{k_1} := \sum_{j \in \{cpu\}} \frac{\Delta Q_j^{k_1}}{\Delta Q^{k_1}}$;

17. end while

End

以下定理给出了 VSERM 算法的特性与相关分析.

定理 1 采用 VSERM 算法时,实际调度结果与预设调度原则之间的偏差 ψ 存在明确上界.

证明 设在$[t_1, t_2]$时间段内算法执行 k 次,其调度序列为 $< V_{s1}, V_{s2}, \cdots, V_{sk} >$.虚拟机 V_i在每轮调度后的 CPU 调度额度和能耗比例分别记为 $B_i(n)$和 $\pi_i(n)$.依据 VSERM 算法中的步骤 11 ~ 步骤 16,对于任意虚拟机 V_i,在第 k 轮调度时其 $B_i(k)$满足

$$B_i(k) - B_i(1) = \sum_{n=1}^{k-1} \left(\frac{\pi_{s_n}(n) - B_{s_n}(n)}{W_{s_n}} W_i \right) - U_i(t_1, t_2) \tag{7}$$

其中,s_n是第 n 次调度时获得 CPU 的虚拟机下标.依据式(7)可以得到

$$\frac{U_i(t_1, t_2)}{W_i} = \sum_{n=1}^{k-1} \frac{\pi_{s_n}(n) - B_{s_n}(n)}{W_{s_n}} + \frac{B_i(1) - B_i(k)}{W_i} \tag{8}$$

因此,$\forall\, i, j\ (i \neq j)$可得

$$\begin{aligned} &\left| \frac{U_i(t_1, t_2)}{W_i} - \frac{U_j(t_1, t_2)}{W_j} \right| \\ &= \left| \frac{B_i(1) - B_i(k)}{W_i} - \frac{B_j(1) - B_j(k)}{W_j} \right| \\ &\leqslant \left| \frac{B_i(1) - B_i(k)}{W_i} \right| + \left| \frac{B_j(1) - B_j(k)}{W_j} \right| \\ &\leqslant \frac{\max\limits_{n \in \{1 \cdots k\}} \{B_i(n)\}}{W_i} + \frac{\max\limits_{n \in \{1 \cdots k\}} \{B_j(n)\}}{W_j} \\ &\leqslant \frac{\max\limits_{n \in \{1 \cdots k\}} \{B_i(n), B_j(n)\}}{\min\{W_i, W_j\}} \end{aligned} \tag{9}$$

结合式(5)可知,调度偏差 ψ 存在明确上界.

证毕

定理 2 若所有虚拟机的调度权重相同,VSERM 算法能够获得最小偏差 ψ.

证明 由于 $W_1 + W_2 + \cdots + W_n = 1$ 且 $\forall\, i\ W_i > 0$,因此当 $W_1 = W_2 = \cdots = W_n$ 时,对于任意 i 和 j 的组合($i \neq j$),表达式 $\min\{W_i, W_j\}$都是最大化的.结合式(9)的结论可知,偏差 ψ 的上界与 $\min\{W_i, W_j\}$成反比.因此,$W_1 = W_2 = \cdots = W_n$ 是最小化 ψ 的充分条件.

证毕

在现有调度算法中,调度偏差 ψ 只能通过事后统计的方式来获得,因此具有较大的不确定性.定理 1 的结论显示,VSERM 算法的偏差 ψ 具有明确的上界,因此便于在执行之前对算法性能进行定量的分析和预测.

定理 3 在 VSERM 算法执行过程中,每次被调度的虚拟机 V_i 都满足

$$\forall\, i \in \{1, \cdots, M\}, \min\left\{ \frac{1}{W_i} \cdot \frac{\Delta Q_J^i(t_1, t_2)}{\Delta Q_J(t_1, t_2)} \right\}$$

其中,M 为虚拟机总数,集合 $J = \{\text{CPU}, \text{Ram}, \text{Disk}, \text{IO}\}$表示能耗组件,$[t_1, t_2]$是 V_i 相继两次获得调度之间的时间间隔.

证明 设在$[t_1, t_2]$时间段内算法共执行 k 次.由定理条件可知,在时间段$[t_1, t_2]$内 V_i 都处于未被调度的状态.依据 VSERM 算法的步骤 13,可得

$$\pi_i(k) = \pi_i(k-1) + \frac{\Delta Q_{J - \{cpu\}}^i(k-1)}{\Delta Q_{J - \{cpu\}}(k-1)} \tag{10}$$

展开式(10)的递归表达式,可得

$$\pi_i(k) = \pi_i(1) + \sum_{n=1}^{k-1} \frac{\Delta Q_{J - \{cpu\}}^i(n)}{\Delta Q_{J - \{cpu\}}(n)} \tag{11}$$

其中,$\pi_i(1)$为 t_1 时刻 V_i 的能耗比例.由于 V_i 在 t_1 时刻

之前处于调度状态,依据步骤 16 可得

$$\pi_i(1)=\frac{\Delta Q^i_{\{cpu\}}(1)}{\Delta Q_{\{cpu\}}(1)} \tag{12}$$

将式(12)代入式(11)可得

$$\pi_i(k)=\frac{\Delta Q^i_{\{cpu\}}(1)}{\Delta Q_{\{cpu\}}(1)}+\sum_{n=1}^{k-1}\frac{\Delta Q^i_{J-\{cpu\}}(n)}{\Delta Q_{J-\{cpu\}}(n)}=\sum_{n=1}^{k-1}\frac{\Delta Q^i_J(n)}{\Delta Q_J(n)} \tag{13}$$

由于$[t_1,t_2]$时间段内 V_i处于未调度状态,因此 $U_i(t_1,t_2)=0$且 $B_i(1)=0$.结合式(7)和式(13)可得

$$\pi_i(k)-B_i(k)=\sum_{n=1}^{k-1}\frac{\Delta Q^i_J(n)}{\Delta Q_J(n)}-\sum_{n=1}^{k-1}\frac{\pi_{s_n}(n)-B_{s_n}(n)}{W_{s_n}}W_i \tag{14}$$

其中,$s_1,\cdots,s_{k-1}$是$[t_1,t_2]$时间段内获得调度的虚拟机下标序列.式(14)左右除以 W_i可得

$$\frac{\pi_i(k)-B_i(k)}{W_i}=\frac{1}{W_i}\sum_{n=1}^{k-1}\frac{\Delta Q^i_J(n)}{\Delta Q_J(n)}-\sum_{n=1}^{k-1}\frac{\pi_{s_n}(n)-B_{s_n}(n)}{W_{s_n}} \tag{15}$$

由 VSERM 的步骤 9 可知,满足以下条件的虚拟机将获得调度

$$\min\left\{\frac{1}{W_i}\sum_{n=1}^{k-1}\frac{\Delta Q^i_J(n)}{\Delta Q_J(n)}-\sum_{n=1}^{k-1}\frac{\pi_{s_n}(n)-B_{s_n}(n)}{W_{s_n}}\right\} \tag{16}$$

显然,式(16)中的 $\sum_{n=1}^{k-1}\frac{\pi_{s_n}(n)-B_{s_n}(n)}{W_{s_n}}$ 只与$[t_1,t_2]$时间段内的调度序列相关,对所有虚拟机而言,其值是相同的.因此,式(16)的调度策略等价于

$$\min\left\{\frac{1}{W_i}\sum_{n=1}^{k-1}\frac{\Delta Q^i_J(n)}{\Delta Q_J(n)}\right\} \tag{17}$$

证毕

定理 4 若虚拟机的调度权重相同,VSERM 算法采用"最近最小能耗比例优先"策略进行调度.

证明 由定理 3 的结论可知,当 $W_1=W_2=\cdots=W_n$ 时,VSERM 的调度策略等价于

$$\min\left\{\sum_{n=1}^{k-1}\frac{\Delta Q^i_J(t_1,t_2)}{\Delta Q_J(t_1,t_2)}\right\} \tag{18}$$

即 $\min\{\pi_i(t_1,t_2)\}$.而$[t_1,t_2]$是 V_i相继两次获得调度之间的时间间隔.因此,VSERM 算法采用"最近最小能耗比例优先"的策略进行调度.

证毕

定理 3 和定理 4 的结论显示,VSERM 算法是依据能耗比例 π_i与调度权重 W_i的比值来确定调度优先级的.即近期能耗比例因子越小,调度权重越大的虚拟机能够获得较高的调度优先级.这种调度策略能够同时兼顾系统中的所有虚拟机:若虚拟机等待调度的时间越长,其能耗比例 π_i的值则会越小,直至其获得调度;当各个虚拟机的 π_i比较接近时,调度权重 W_i的取值将其主导作用,即权重越大则调度优先级越高.

5 实验与性能分析

5.1 实验参数与配置

为了分析 VSERM 算法的性能表现,实验选择三种现有的虚拟机调度算法作为对比对象:CS[18],vSlicer[28]和 GIOS[14].为了模拟"混合型负载",实验选择了四个基准测试程序,包括 bzip2,mcf,TPC-W 和 IOZone 作为基本负载.其中,bzip2 和 mcf 来自 SPECcpu2006;TPC-W 则是用于分析 Web 服务器性能的测试程序;IOZone 在运行过程会产生大量的文件系统操作.通过用不同的方式来组合这些基准测试程序,我们构建了四组测试负载,记为 WL#1,WL#2,WL#3,WL#4.

5.2 调度偏差对比与分析

本次实验主要评估在不同特性的测试负载下,各种调度算法的调度偏差.实验分四组进行,每组采用不同的测试负载(WL#1 ~ WL#4),实验结果如图 1 所示.

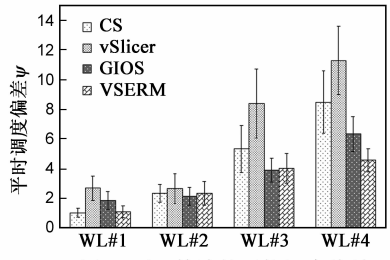


图1 不同算法的平均调度偏差

实验结果显示,当负载为纯 CPU 密集型任务时(WL#1),CS 和 VSERM 的调度偏差明显优于 vSlicer 和 GIOS,vSlicer 是四种调度算法中表现最差的.vSlicer 的最大特点是采用了"动态切片"技术来确定执行时间.通过观察实验过程,我们发现当 CPU 利用率不高时,vSlicer 倾向于采用更细粒度的切片技术.实验所用的测试负载 WL#1 虽然属于计算密集性,但其中的 bzip2 程序对 CPU 的要求明显高于 mcf,后者则更多地集中在内存访问方面.因此,在 vSlicer 算法执行过程中,运行 bzip2 的虚拟机获得了更多的处理器份额,从而增加了调度偏差.CS 和 VSERM 在调度偏差方面的表现差别不大,但两者产生调度偏差的原因却不同.如 VSERM 算法步骤 12 所示,其调度偏差处决于任务的等待时间,即等待时间较长的任务可能获得更多的额外 CPU 时间.

当测试负载为纯 I/O 密集型任务时(WL#2),实验结果显示四种算法的调度偏差比较接近.原因在于,I/O 密集型任务对 CPU 的要求很低,因此处理器在调度过程经常处于空闲状态,而各种调度算法则能够比较容易地依据调度权重来分配 CPU.在本次实验中,IO-

Zone对 CPU 的要求明显低于 TPC-W,这就是产生图 1 所示的调度偏差的主要原因.

当测试程序为“混合型负载”(WL # 3 和 WL # 4)时,四种算法的性能表现出现了明显的差别.总体而言,GIOS 和 VSERM 算法的表现明显优于 CS 和 vSlicer,而 vSlicer 的调度偏差是最大的.vSlicer 算法产生很大调度偏差的原因在于负载中的计算密集型任务(bzip2 和 mcf)获得了绝大部分 CPU 时间,而 I/O 密集型任务则只获得很少的 CPU 时间.从提高 CPU 利用率的角度而言,vSlicer 算法的策略非常有效,但也显著降低了 I/O 密集型任务的执行性能.当测试负载为 WL # 3 时,GIOS 和 VSERM 的调度偏差几乎相同,这一结果表明,I/O 批调度方式能够显著降低调度偏差.GIOS 采用的 I/O 批调度方式将多个具有 I/O 请求的任务合为单一的调度单元,这不仅提高了并发 I/O 任务的响应时间,也保证了 I/O 密集型任务在与 CPU 密集型任务竞争 CPU 时的优先权,从而降低了调度偏差.当测试负载为 WL # 4 时,系统所面对的混合负载强度显著增加,此时 GIOS 的调度偏差出现了约 40% 的增加,而 VSERM 算法则几乎保持不变.如前所述,VSERM 算法产生调度偏差的主要原因是等待中的任务可能获得更多的额外 CPU 时间.但是,当所有的任务等待时间都成比例地增加时,这种额外的 CPU 时间被均匀地分到了所有虚拟机实例上,从而减少了系统最终的调度偏差.

5.3　能效对比与分析

近期,研究者大多采用“相对能效”来描述虚拟化系统的能效,即在非虚拟化和虚拟化平台上运行相同的任务,然后将两种情况下的能耗开销比值定义为虚拟系统的“相对能效”(Normalized Energy-Efficiency, NEE).此外,已有研究显示[15,20]:系统的虚拟化程度(Virtualization Degree, VD)是影响虚拟化系统能效的重要因素,该指标一般定义为“任务数量与虚拟机数量的比值”.因此,本次实验将分析不同的 VD 参数对系统 NEE 指标的影响.实验结果如图 2 所示.

实验结果显示,系统负载特征对 NEE 指标具有显著影响.具体而言,当负载为计算密集型任务时,NEE 指标明显较高;而当负载为纯 I/O 密集型任务时,NEE 指标普遍偏低.产生这种现象的原因是:与虚拟化平台相关的机制都将引起系统额外的能耗开销,并延长任务的 I/O 处理和响应时间,最终造成 NEE 指标的下降.如图 2(*b*)所示,在所有实验中,只有 GIOS 算法在 VD = 5.0 和 6.0 时,系统的 NEE 指标才大于 1.GIOS 采用了 I/O 组调度方式,这种调度机制对纯 I/O 密集型负载具有较好的能效优化效果.实验结果还显示,NEE 指标首先随 VD 参数的增加而提升,当达到一个较高值后则呈缓慢降低趋势.在本次实验中,我们采用增加测试程序实例的方式来调整 VD 值.因此,在 VD 值较小时,虚拟化平台自身的开销在系统总能耗中的占比较大;当 VD 值较大时,这种占比开始明显降低.当系统负载强度达到一定程度后,系统的 NEE 开始大于 1,这表明虚拟化平台的能效开始高于非虚拟化平台.以图 2(*a*)为例,在 VD = 3.0 时,除 GIOS 算法外,其它算法对应的 NEE 都大于 1.因此,一般虚拟化系统都依据 NEE 是否大于 1 来判断系统的最小 VD 取值.

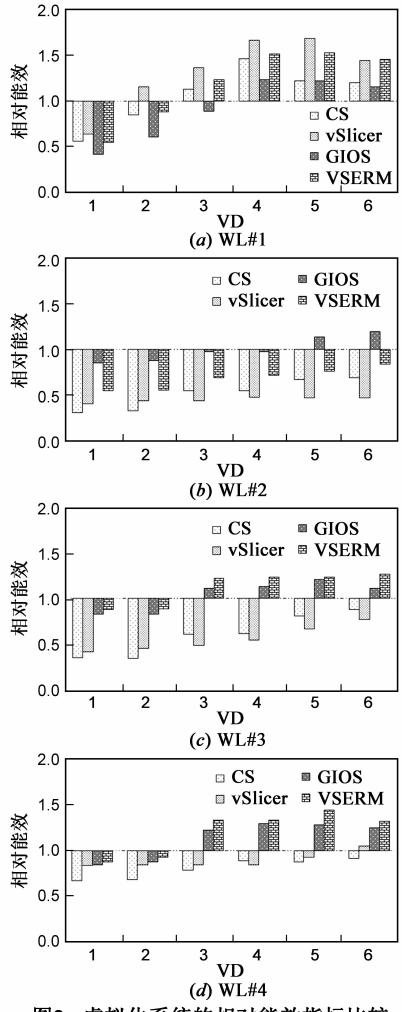


图2　虚拟化系统的相对能效指标比较

当测试负载为 WL # 1 时,实验结果显示,vSlicer 算法对应的 NEE 指标最优,其原因是 vSlicer 算法能够显著提高 CPU 的有效利用率,而 CPU 能耗在系统总能耗中的占比又是最高的.但当测试负载为 WL # 2 时,vSlicer 算法的 NEE 指标则是所有算法中最差的.因此,vSlicer 算法只能适用于纯计算密集型任务的调度.当系统采用 CS 算法时,只有在测试负载为纯计算密集型且 VD ≥3.0 时,虚拟化系统的能效优化效果才能显现.这也表明,当能效优化为系统主要目标时,CS 算法比较适用于计算密集型负载.

当测试负载为 WL#3 和 WL#4 时,GIOS 和 VSERM 算法的 NEE 指标明显优于 CS 和 vSlicer 算法.以 CS 算法为例,由于其采用了 Working-conserving 机制[18],这就导致测试负载中的 bzip2 和 mcf 程序得到了过多的处理器时间,从而提前结束;其它 I/O 密集型任务即使在完成其 I/O 操作后仍需等待一定的时间才能获得处理器,从而延长了其整体执行时间.在面对"混合型负载"时,这种调度过程中的偏向性导致 CS 算法的 NEE 指标偏低.以此相反,GIOS 算法则偏向于 I/O 密集型任务,因此即使在面对"混合型负载"时,GIOS 的 NEE 指标也较高.VSERM 在调度过程中不存在明显的偏向性,其调度策略是"最近最小能耗比例优先".因此,不论任务属于计算密集型还是 I/O 密集型,都能依据其最近的能耗比例情况来获得相应的调度优先级.这种特性在实验过程中的表现是:在执行 WL#3 和 WL#4 时,VSERM 算法的总执行时间要低于其他算法,并没有出现调度偏向性的现象.因此,VSERM 算法非常适用于"混合型负载"的调度.

6 结论

在开放的云计算环境中,混合型负载是虚拟化数据中心经常需要面对的情况.本文提出的 VSERM 算法采用了"能耗比例"作为调度的基本指标,其优点主要体现在两个方面:降低调度偏差和提高虚拟化系统的能效.降低调度偏差有利于提高调度算法的可预测性和可分析性,而提高虚拟化系统的能效则能够降低云服务提供者的运营成本.实验结果显示,VSERM 算法适用于经常出现"混合型负载"的虚拟化系统.在今后的工作中,我们将关注如何将 VSERM 算法的优点与其它调度算法的优点互相结合.

参考文献

[1] Goiri Í, Berral J L, Fitó J O. Energy-efficient and multifaceted resource management for profit-driven virtualized data centers [J]. Future Generation Computer Systems, 2012, 28(5): 718 – 731.

[2] Wang J, Feng L. A survey on energy-efficient data management [J]. ACM SIGMOD Record, 2011, 40(2): 17 – 23.

[3] G. Goth. Data center operators face energy irony[J]. IEEE Internet Computing, 2010, 14(2): 7 – 10.

[4] Kant K. Data center evolution: a tutorial on state of the art, issues, and challenges[J]. Computer Networks, 2009, 53(17): 2939 – 2965.

[5] Mateescu G, Gentzsch W, Ribbens C J. Hybrid computing-where HPC meets grid and cloud computing[J]. Future Generation Computer Systems, 2011, 27(5): 440 – 453.

[6] 俞能海,郝卓,徐甲甲,张卫明,张驰.云安全研究进展综述[J].电子学报,2013,41(2):371 – 381.
Yu Nenghai, Hao Zhuo, Xu Jiajia, Zhang Weiming, Zhang Chi. Review of cloud computing security[J]. Acta Electronica Sinica, 2013, 41(2): 371 – 381. (in Chinese)

[7] Chapman C, Emmerich W, Marquez, F G, Clayman S, Galis A. Software architecture definition for on-demand cloud provisioning[J]. Cluster Computing, 2012, 15(2): 79 – 100.

[8] Liu H, Jin H, Xu C Z, Liao X F. Performance and energy modeling for live migration of virtual machines[J]. Cluster Computing, 2013, 16(2): 249 – 264.

[9] Lovasz G, Niedermeier F, De-Meer H. Performance tradeoffs of energy-aware virtual machine consolidation[J]. Cluster Computing, 2013, 16(3): 481 – 496.

[10] Kansal A, Zhao F, Liu J, Kothari N, Bhattacharya A A. Virtual machine power metering and provisioning[A]. Proceedings of ACM Symposium on Cloud Computing[C]. New York, USA: ACM Press, 2010. 39 – 50.

[11] Chen H, Li Y, Shi W. Fine-grained power management using process-level[J]. Sustainable Computing: Informatics and Systems, 2012, 2(1): 33 – 42.

[12] Abd-El-Malek M, Wachs M, Cipar J, Sanghi K, Ganger G R, Gibson G A, Reiter M K. File system virtual appliances: portable file system omplementations[J]. ACM Transactions on Storage, 2012, 8(3): 1 – 36.

[13] Ibrahim K Z, Hofmeyr S, Iancu C. Characterizing the performance of parallel applications on multi-socket virtual machines [A]. Proceedings of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing[C]. Washington, USA: IEEE Press, 2011. 1 – 12.

[14] Kang H, Chen Y, Wong J L, Sion R, Wu J. Enhancement of xen's scheduler for mapReduce workloads[A]. Proceedings of International Symposium on High Performance Distributed Computing[C]. New York, USA: ACM Press, 2011. 251 – 262.

[15] Gao F, Abd-Almageed F, Hefeeda M. Distributed approximate spectral clustering for large-scale datasets[A]. Proceedings of International Symposium on High Performance Distributed Computing[C]. New York, USA: ACM Press, 2012. 223 – 234.

[16] Zhang X, Xu Y, Jiang S. You Choose: choosing your storage device as a performance interface to consolidated I/O service [J]. ACM Transactions on Storage, 2011, 7(3): 1 – 41.

[17] 宫学庆,金澈清,等.数据密集型科学与工程:需求和挑战[J].计算机学报,2012,35(8):1563 – 1578.
Gong Xueqing, Jin Cheqing, Wang Xiaoling, Zhang Rong, Zhou Aoying. Data-intensive science and engineering: requirements and challenges [J]. Chinese Journal of Computers, 2012, 35(8): 1563 – 1578. (in Chinese)

[18] Cherkasova L, Gupta D, Vahdat A. Comparison of the three

cpu schedulers in xen[J]. ACM SIGMETRICS Performance Evaluation Review, 2007, 35(2): 42 – 51.

[19] Kim H, Lim H, Jeong J, Jo H, Lee J. Task-aware virtual machine scheduling for I/O performance[A]. Proceeding ACM SIGPLAN/SIGOPS International Conference on Virtual execution Environments[C]. New York, USA: ACM Press, 2009. 101 – 110.

[20] Dhiman G, Marchetti G, Rosing T. vGreen: a system for energy-efficient management of virtual machines[J]. ACM Transactions on Design Automation of Electronic Systems, 2010, 16(1): 1 – 27.

[21] Lefevre L, Orgerie A C. Designing and evaluating an energy efficient cloud[J]. Journal of Supercomputing, 2010, 51(3): 352 – 373.

[22] Lindberg P, Leingang J. Comparison and analysis of scheduling heuristics for optimization of energy consumption and makespan in large-scale distributed systems[J]. Journal of Supercomputing, 2012, 59(1): 323 – 360.

[23] Manzanares A, Qin X, Ruan X, Yin S. PRE-BUD: prefetching for energy-efficient parallel I/O systems with buffer disks[J]. ACM Transactions on Storage, 2011, 7(1): 1 – 45.

[24] Martinez M R, Valdivia H, Seguel J, Greer M V. Estimating power/energy consumption in database servers[J]. Procedia Computer Science, 2011, 6(2): 112 – 117.

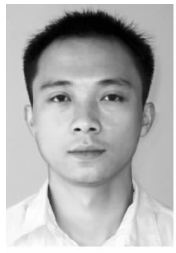
[25] 李建敦,彭俊杰,张武.云存储中一种基于布局的虚拟磁盘节能调度方法[J].电子学报, 2012, 40(11): 2247 – 2254.
Li Jiandun, Peng Junjie, Zhang Wu. A layout-based energy-aware approach for virtual disk scheduling in cloud storage [J]. Acta Electronica Sinica, 2012, 40(11): 2247 – 2254. (in Chinese)

[26] Bircher W L, John L K. Complete system power estimation using processor performance events[J]. IEEE Transactions on Computers, 2012, 61(4): 563 – 577.

[27] Bertran R, Becerra Y, Carrera D. Energy accounting for shared virtualized environments under DVFS using PMC-based power models[J]. Future Generation Computer Systems, 2012, 28(2): 457 – 468.

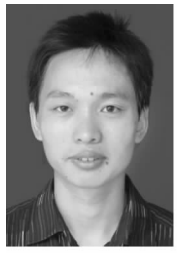
[28] Xu C, Gamage S, Rao P N, Kangarlou A, Kompella R R, Xu D. vSlicer: latency-aware virtual machine scheduling via differentiated-frequency CPU slicing[A]. Proceedings of International Symposium on High-Performance Parallel and Distributed Computing[C]. New York, USA: ACM Press, 2012. 3 – 14.

作者简介



肖　鹏(通信作者)　男,1979 年出生,湖南湘潭人,博士,湖南工程学院计算机系讲师,主要研究方向为高性能计算,云计算,能效优化等.
E-mail: efn4623@126.com

刘洞波　男,1974 年出生,湖南宁乡人,博士,湖南工程学院副教授,主要研究方向为分布式计算,云计算,网络能效优化等.



屈喜龙　男,1978 年出生,湖南新邵人,博士后,湖南工程学院副教授,主要研究方向为云制造、网络系统集成等.