doi:10.3969/j.issn.1000-436x.2014.09.008

考虑虚拟机间性能互扰的虚拟资源分配方法

杨雷[1]，邢禾续[2]，代钰[2]，徐赛[1]，王昊[1]，张斌[1]
（1. 东北大学 信息科学与工程学院，辽宁 沈阳 110819；2. 东北大学 软件学院，辽宁 沈阳 110819）

摘 要：提出了一种考虑虚拟机间性能互扰的虚拟资源分配方法。构建了虚拟机性能互扰度量模型，并给出了基于虚拟机负载模式的性能互扰度预测方法。在此基础上将对虚拟机间性能互扰的预测结果应用于资源分配方案性能互扰评价中，以作为虚拟资源分配的依据，能够有效保证虚拟资源分配方案的有效性。实验结果表明，和已有方法相比，提出的虚拟资源分配方法能够保证虚拟机上所部署应用的执行性能。

关键词：虚拟机性能互扰；虚拟资源分配；负载模式；性能互扰度预测
中图分类号：TP393 **文献标识码**：A **文章编号**：1000-436X(2014)09-0079-12

Virtual resource allocation method with the consideration of performance interference among virtual machines

YANG Lei[1], XING He-xu[2], DAI Yu[2], XU Sai[1], WANG Hao[1], ZHANG Bin[1]
(1. College of Information Science and Engineering, Northeastern University, Shenyang 110819, China;
2. College of Software, Northeastern University, Shenyang 110819, China)

Abstract: A method for allocating the virtual resource with the consideration of the performance interference among virtual machines was proposed. The model for evaluating the performance interference among virtual machines was formed and the method for predicting the performance interference was presented based on the workload pattern of the virtual machine. The predicted result of the performance interference among virtual machines was put into the evaluation of the resource allocation plan to be used as the basis for resource allocation. By doing this, the effectiveness of the resource allocation plan could be insured. The experiments show that the proposed virtual resource allocation method can insure the execution performance of the application deployed on the virtual machines.

Key words: performance interference among virtual machines; virtual resource allocation; load pattern; prediction of performance interference

1 引言

随着 Web 技术的发展，近年来云计算技术已经得到了工业界和学术界的广泛关注[1]。虚拟化技术的出现，使云数据中心能够动态组织多种计算资源，隔离具体的硬件体系结构和软件系统之间的紧密依赖关系，实现透明化的可伸缩计算系统架构[2]，从而灵活构建满足多种应用需求的计算环境，提高计算资源的使用效率，发挥计算资源的聚合效能，满足日益多样的计算需求。

云计算中需要根据应用的资源需求，为应用分配合适的虚拟资源以保证其执行性能。这就需要分析应用执行性能与虚拟资源需求之间的关系，并在该关系分析的基础上，以保证应用执行性能为目标，为任务分配合适的虚拟资源。然而，应用执行性能与虚拟资源需求之间的关系并不是确定的而是动态变化的，这主要是因为：一方面应用执行性能受应用工作负载大小的影响，当应用工作负载变

收稿日期：2013-09-08；**修回日期**：2014-03-29
基金项目：国家自然科学基金资助项目(60903008, 61073062)；中央高校基本科研业务费专项基金资助项目(N110404008, N110417003, N110804002)
Foundation Items: The National Natural Science Foundation of China (60903008, 61073062); The Fundamental Research Funds for the Central Universities (N110404008, N110417003, N110804002)

化时，在同一资源配置下，应用执行性能将发生变化；另一方面，应用执行性能受虚拟资源服务能力变化影响，虚拟资源部署在物理机上，目前的虚拟化技术能够为虚拟机间提供一些有效的性能隔离机制，但是当监控器将系统资源分片并分配给部署在同一物理机上的不同虚拟机时，将使虚拟机间由于争夺同一物理资源，而导致虚拟资源服务能力变化（通常将虚拟机间所存在的这样一种由于物理资源争夺所导致的服务能力变化称为虚拟机性能互扰），从而一个虚拟机在应用执行过程中对同一物理资源的抢占将影响另一个虚拟机的服务能力，进而影响其上所部署应用的执行性能。这就需要在虚拟资源分配问题求解中，考虑任务工作负载以及虚拟机间性能互扰对虚拟资源分配结果的影响。目前的研究工作中，在考虑任务工作负载动态变化的虚拟资源分配方面，已经开展了大量的研究工作，而考虑虚拟机间性能互扰的资源分配方面的研究工作还较少。为了保证云计算中应用的执行性能，需要在考虑虚拟机间性能互扰的基础上研究资源分配问题。

为此，本文提出了考虑虚拟机间性能互扰的虚拟资源分配方法，并给出了面向资源分配的虚拟机性能互扰度量和预测算法。针对虚拟机间性能互扰度量问题，构建了虚拟机互扰度模型，给出了基于多元线性回归的模型参数求解方法；针对虚拟机性能互扰预测问题，给出了基于层次聚类的虚拟机负载模式挖掘算法和虚拟机负载模式相似性匹配方法，并提出了基于虚拟机负载模式的性能互扰度预测算法和面向资源分配的资源分配方案互扰度评价方法，以通过对资源分配方案的互扰度评价得到虚拟机间性能互扰较小的资源分配方案。最后通过一系列实验验证了所提出的虚拟机性能互扰和预测算法以及考虑虚拟机间性能互扰的资源分配方法的有效性。

2 考虑虚拟机间性能互扰的资源分配方法

云计算中虚拟资源的广泛使用使资源分配问题求解中需要考虑虚拟资源的特点，保证资源分配结果的有效性。目前的虚拟化技术虽然能够为虚拟机提供一些有效的隔离机制[3]，如安全隔离可以防止其他虚拟机上的恶意程序攻击或截取数据，错误隔离能够防止一个程序的错误行为导致整个系统的崩溃，环境隔离使多个具有不同客操作系统的虚拟机能够运行在同一主机上，然而目前的虚拟化技术很难为同一主机上的虚拟机提供有效的性能隔离，即虚拟机行为能够对其他虚拟机造成影响并体现在性能变化上。因此，本文将在分析虚拟资源性能互扰特点的基础上，提出云计算中考虑虚拟机间性能互扰的虚拟资源分配方法。

为了验证虚拟机性能互扰现象的存在，本文将通过实验进行验证，实验中物理机配有 Intel Core i3 2120 3.3 G 的处理器、4 G 内存以及 250 G 硬盘，OS 版本为 Ubuntu 12.04。虚拟化系统使用目前流行的 Xen，Xen 虚拟机配置为 4VCPU、1 G 内存及 8 G 硬盘，客操作系统为 Ubuntu 12.04。由于页面限制，本文仅给出如表 1 所示的实验中所研究的部分应用。

表 1 部分测试应用程序

应用名称	应用类型
Super PI	CPU 密集型
cat	Disk I/O 密集型
Bzip2	混合型
Cp	Disk I/O 密集型
Gzip	混合型

表 1 中，应用 Super PI 是利用 CPU 的浮点运算能力来计算出 π，具有较高的 CPU 资源需求；cat 可以产生大量的磁盘读写操作，具有较高的 I/O 资源需求；Cp 可以产生磁盘写操作，具有较高的 I/O 资源需求；Bizp2 和 Gzip 是压缩工具，同时具有较高的 CPU 和 I/O 资源需求。实验同时运行 2 个 Xen 虚拟机，每个 Xen 虚拟机上运行表 1 中的各测试应用，以标准执行时间（标准执行时间为应用部署到物理机后的实际执行时间与应用在独占物理机情况下的执行时间之比）作为衡量虚拟机性能的评价指标，通过更换 Xen 虚拟机上的应用，得到标准执行时间结果如表 2 所示。

表 2 实验结果

	Super PI	cat	Bzip2
Super PI	1.018	1.009	1.021
cat	1.465	2.108	1.114
Bzip2	1.066	1.760	1.113

由表 2 可知，虚拟机上所部署应用的性能受到其他虚拟机所运行应用的影响，例如，CPU 密集型

的 Super PI 应用受到其他虚拟机的影响较小，而当磁盘操作较多的 cat 与磁盘操作较多的应用一起运行时，性能下降比较明显。根据表 2 可以看出，虚拟机间存在性能互扰，且当这些虚拟机所部署应用不同时，虚拟机间性能互扰程度不同，导致应用性能下降程度也不同。这就需要在虚拟资源分配问题的解决中考虑虚拟资源间的性能互扰对分配结果的影响，即对虚拟资源间性能互扰程度进行度量和预测，并根据预测结果对资源分配方案进行评估，以作为资源分配方案选取和调整的依据，保证资源分配结果的有效性。

因此，本文提出了一种考虑虚拟机间性能互扰的虚拟资源分配方法，图 1 给出了该方法的基本框架，其中，虚拟资源预分配器用于根据可用物理资源以及用户的资源需求，找到满足用户资源需求的物理机，作为可能的资源分配方案；虚拟机性能互扰历史数据库用于保存虚拟机部署的详细信息，包括各虚拟机性能及资源使用模式，是虚拟机性能互扰度量与预测的基础；虚拟机性能互扰度模型生成器是虚拟机间性能互扰度度量的核心，用于根据虚拟机性能互扰历史数据，建立虚拟机间性能互扰度模型；虚拟机性能互扰度量模型库用于保存已构建的虚拟机性能互扰度模型；虚拟机负载模式挖掘器用于根据虚拟机性能互扰度量模型库，对已构建的虚拟机间性能互扰度模型进行聚类；虚拟机性能互扰度预测器用于预测待部署虚拟机被放置在某一物理机后其与该物理机上其他虚拟机间的性能互扰程度；虚拟资源分配器是在可能的资源分配方案，根据虚拟机性能互扰度预测结果，选择具有最小虚拟机性能互扰度的资源分配方案作为虚拟资源分配结果以完成对虚拟资源的分配。

该方法中，虚拟机间性能互扰度量和预测是其中的关键问题，限于篇幅原因，本文将重点解决这 2 个问题。

3　虚拟机间性能互扰度量及预测算法

在本文所提出的考虑虚拟机间性能互扰的虚拟资源分配方法中，由于虚拟机间存在性能互扰，为了保证资源分配结果的有效性，本文将通过对共享同一物理机的虚拟机间性能互扰度进行度量和预测，以确定待分配虚拟机将被放置在哪个物理机上。接下来，将给出虚拟机间性能互扰度量及预测算法。

3.1　基于多元线性回归的虚拟机间性能互扰度量模型

对于一个虚拟机，其负载可以记为 W=<*cpuutil*, *memutil*, *rps*, *wps*, *await*, *svctm*>，其中，*cpuutil* 为背景负载平均 CPU 利用率；*memutil* 为背景负载平均内存利用率；*rps* 为每秒平均 I/O 读请求数；*wps* 为每秒平均 I/O 写请求数；*await* 为 I/O 请求的平均等待时间，指 I/O 请求从发出到执行完毕的时间；*svctm* 为 I/O 请求实际平均服务时间，指发送到设备的 I/O 请求的实际平均执行时间。

为了描述方便，将待分配虚拟机称为目标虚拟机，其负载记为 *FW*。如果待分配虚拟机将放置在物理机 *F* 上，记物理机 *F* 上已部署虚拟机集合为 $VMS=\{vm_1, vm_2, \cdots, vm_n\}$，对于其中任意一个虚拟机 vm_i，其负载记为 bw_i。在所提出的考虑虚拟机间性能互扰的虚拟资源分配方法中，需要度量目标虚拟机与物理机 *F* 上已部署虚拟机间的性能互扰度，由于物理机 *F* 上可能已经部署了多个虚拟机，这就需要度量目标虚拟机放置在 *F* 上之后，其他已部署虚拟与目标虚拟机间的性能影响程度，从而在虚拟资源分配问题求解中，选择与目标虚拟机性能影响程度最小的物理机作为目标虚拟机的放置对象。这也就是说，单独考察 *VMS* 中某一个虚拟机与目标虚拟机的性能互扰程度不能有效反映目标虚拟机放置在物理机之后的性能影响，而是需要度量 *VMS* 中所有虚拟机与目标虚拟机间的性能互扰程度。目

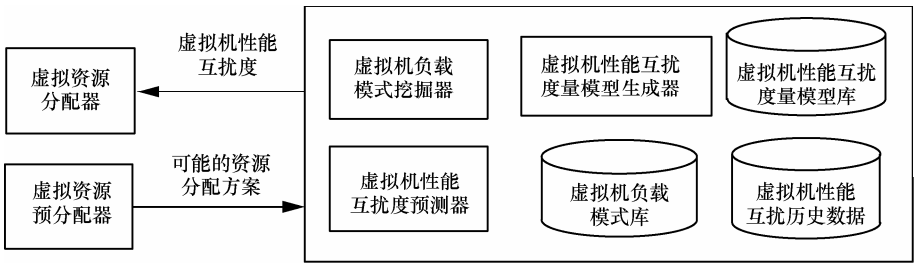


图 1　考虑虚拟机间性能互扰的虚拟资源分配方法框架

标虚拟机放置在物理机 F 上之后，其性能将受物理机 F 上其他虚拟机的影响，为了描述方便，将物理机 F 上已部署虚拟机集合 *VMS* 称为目标虚拟机的背景虚拟机，并称 *VMS* 中所有虚拟机的负载之和为背景负载，记为 $BW=<cpuutil_{BW}, memutil_{BW}, rps_{BW}, wps_{BW}, await_{BW}, svctm_{BW}>$，其中，$cpuutil_{BW}=\sum bw_i.cpuutil$，$memutil_{BW}=\sum bw_i.memutil$，$rps_{BW}=\sum bw_i.rps$，$wps_{BW}=\sum bw_i.wps$，$await_{BW}=\sum bw_i.await$，$svctm_{BW}=\sum bw_i.svctm$。下面将给出虚拟机性能互扰度的定义。

定义 1 虚拟机性能互扰度（PID, performance-interference degree）。虚拟机性能互扰度反映了如果将待分配虚拟机放置在某一物理机上，其性能受该物理机上其他虚拟机性能影响的程度，记为 *PID*(*FW*@*BW*)。其中，*FW* 为目标虚拟机负载；*BW* 为背景虚拟机负载。

由虚拟机性能互扰度的定义可知，虚拟机性能互扰度是相对于虚拟机独享物理机资源时的性能，表示在与其他虚拟机共享同一物理机时虚拟机性能下降的程度，如式(1)所示。

$$PID(FW@BW)=\frac{Perf(FW@BW)-Perf(FW@Idle)}{Perf(FW@Idle)} \quad (1)$$

其中，*Perf*(*FW*@*BW*)表示待分配虚拟机在背景负载为 *BW* 时的性能，*Perf*(*FW*@*Idle*)表示待分配虚拟机在独享物理机资源时的性能。

虚拟机性能与同一物理机上其他虚拟机的资源使用模式，即负载有关。对于同一虚拟机，不同的背景负载对其性能的影响程度是不同的，因此虚拟机性能互扰度模型构建的目标是建立虚拟机性能互扰度与其背景负载的函数关系，以反映背景负载变化对虚拟机性能互扰的影响程度。

本文对选取的如表 1 所示的测试应用进行应用间性能互扰度测量，即利用检测工具收集无背景负载情况下该应用所在虚拟机的资源负载参数及运行时间，结果如表 3 所示，其中，*cpuutil*、*memutil*、*rps*、*wps*、*await*、*svctm* 为虚拟机负载参数，*cpuutil* 为 CPU 利用率；*memutil* 为内存利用率；*rps* 为每秒的 I/O 读请求数；*wps* 为每秒的 I/O 写请求数；*await* 为 I/O 请求的平均等待时间，指 I/O 请求从发出到执行完毕的时间；*svctm* 为 I/O 请求实际平均服务时间，指发送到设备的 I/O 请求的实际执行时间。

改变 VM_1 的背景负载，即在虚拟机 VM_2 上运行不同的程序，得到 VM_1 上应用运行时间，截取部分结果如表 4 所示。在表 3 和表 4 的基础上，以背景负载参数为自变量，以虚拟机性能互扰度为因变量，利用 matlab 绘制应用背景负载参数 *rps*、*wps*、*await* 以及 *svctm* 与虚拟机性能互扰度 *PID* 的散点图。图 2 给出了 Bzip2 应用的性能互扰度与其背景

表 3 无背景负载时的应用性能表现

App	*cpuutil*	*memutil*	*rps*	*wps*	*await*/s	*svctm*	*Runtime*/s
Bzip2	0.91	0.007	278.099	0.999	0.775	0.345	84.265
cat	0.001	0.044	335.158	433.111	137.643	1.58	27.801
Super PI	0.98	0.001	0.245	1.547	12.222	9.39	99.107
Iozone	0.264	0.036	370.25	392.95	151.99	9.93	108.724
Ccrypt	0.762	0.421	172.77	168.68	5.162	2.13	216.611
Gzip	0.912	0.053	295.72	11.83	8.44	1.18	218.95

表 4 应用在不同背景负载下的运行时间（单位：s）

	Bzip2	cat	Super PI	Iozone	Ccrypt	Gzip
Bzip2	93.887	148.28	89.787	146.988	142.82	144.168
cat	30.996	58.892	40.762	40.762	45.104	50.026
Super PI	101.141	99.981	100.892	121.99	105.21	103.54
Iozone	175.33	205.27	109.756	198.73	110.26	113.64
Ccrypt	245.03	296.55	233.50	311.73	257.91	254.03
Gzip	230.75	393.99	227.09	403.74	245.10	240.02

负载参数的散点图。由图 2 可知，Bzip2 性能互扰度与 *wps*、*rps*、*await* 及 *svctm* 呈现出比较明显的线性关系，因此可以利用多元线性回归分析建立自变量与因变量之间的关系，即建立虚拟机性能互扰度与其背景负载参数间的函数关系。

基于上述分析，可以建立如式(2)所示的虚拟机性能互扰度模型。

$$PID(FW@BW)=a_0+a_1\times cpuutil_{BW}+a_2\times memutil_{BW}+a_3\times rps_{BW}+a_4\times wps_{BW}+a_5\times await_{BW}+a_6\times svctm_{BW} \tag{2}$$

其中，a_0、a_1、a_2、a_3、a_4、a_5、a_6、a_7 为系数，$cpuutil_{BW}$、$memutil_{BW}$、rps_{BW}、wps_{BW}、$await_{BW}$、$svctm_{BW}$ 为目标虚拟机的背景负载参数，$cpuutil_{BW}$ 为背景负载 CPU 利用率；$memutil_{BW}$ 为背景负载内存利用率；rps_{BW} 为每秒的 I/O 读请求数；wps_{BW} 为每秒的 I/O 写请求数；$await_{BW}$ 为 I/O 请求的平均等待时间，指 I/O 请求从发出到执行完毕的时间；$svctm_{BW}$ 为 I/O 请求实际平均服务时间，指发送到设备的 I/O 请求的实际执行时间。

如果能够得到系数 a_0、a_1、a_2、a_3、a_4、a_5、a_6，就可以根据式(2)，在已知背景负载的情况下，得到目标虚拟机与某一物理机上背景虚拟机间的互扰度。下面将给出基于多元线性回归的参数求解方法。该方法的基本思想是：建立多元线性回归方程[4]（如式(3)所示），找到 a_0、a_1、a_2、a_3、a_4、a_5、a_6 的估计值 a_0'、a_1'、a_2'、a_3'、a_4'、a_5'、a_6'，以使观测值 $PID(FW@BW)$ 与回归值 $PID\left(\widehat{FW@BW}\right)$ 的残差平方和最小。

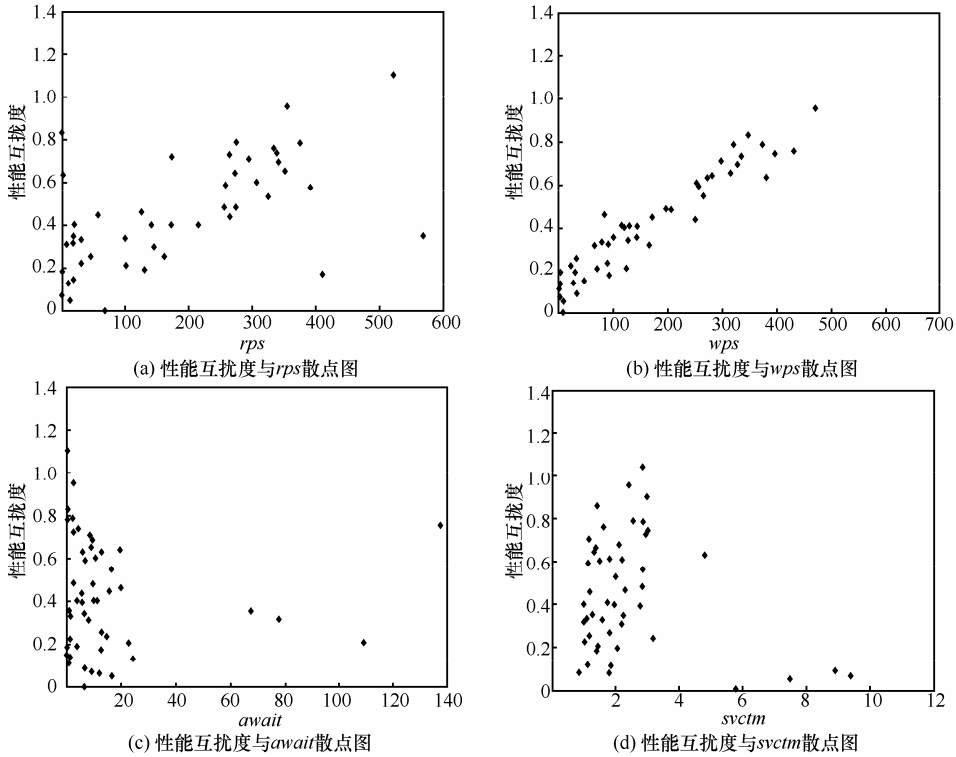
$$PID\left(\widehat{FW@BW}\right)=a_0'+a_1'\times cpuutil_{BW}+a_2'\times memutil_{BW}+a_3'\times rps_{BW}+a_4'\times wps_{BW}+a_5'\times await_{BW}+a_6'\times svctm_{BW} \tag{3}$$

这里，残差平方和为

$$Q=\sum_i\left(PID(FW@BW)_i-PID\left(PID\left(\widehat{FW@BW}\right)\right)_i\right)^2 \tag{4}$$

可求得多元线性回归方程的系数。从而，在已知背景虚拟机负载的情况下，可以根据式(3)得到目标虚拟机的性能互扰程度。限于篇幅的原因，本文将不再给出具体的求解回归方程系数的算法。

基于上述分析，如果可以获得虚拟机间性能互扰历史数据，即背景虚拟机负载以及目标虚拟机和背景虚拟机性能互扰度观测值的历史数据，可以根据式(3)得到虚拟机间性能互扰程度。然而，在虚拟资源分配问题中，由于目标虚拟机还未部署到物理机，目标虚拟机与背景虚拟机间的上述性能互扰历史数据是无法在虚拟资源分配前



(a) 性能互扰度与*rps*散点图　(b) 性能互扰度与*wps*散点图

(c) 性能互扰度与*await*散点图　(d) 性能互扰度与*svctm*散点图

图 2　Bzip2 性能互扰度与其背景负载参数的散点图

得到的，这就使根据历史性能互扰数据构建虚拟机间性能互扰度模型难以实现。为此，本文将通过对已有虚拟机间性能互扰度模型的复用，对虚拟资源分配问题中，目标虚拟机与背景虚拟机间性能互扰程度进行预测。以下，本文将给出虚拟机间性能互扰度预测算法。

3.2 虚拟机间性能互扰度预测算法

虚拟机间性能互扰度模型构建的基础是有较多的可以用于训练并得到回归方程系数的历史数据。但是，在虚拟资源分配问题中，由于虚拟资源分配方案中待分配虚拟机并未真正部署到物理机上，从而并没有可以用于分析和度量这些虚拟机间性能互扰的历史性能数据。为此，本文将研究一个虚拟机间性能互扰度预测算法，以通过对虚拟机间性能互扰度量历史经验的复用，对待分配虚拟机与背景虚拟机间的性能互扰度进行预测，以作为考虑虚拟机间性能互扰的资源分配问题求解依据。

虚拟机间性能互扰度与目标虚拟机和背景虚拟机负载有关。如果 2 个应用的资源使用模式相似，即负载相似，可以利用其中一个的虚拟机性能互扰度模型得到另一个虚拟机在背景负载已知时的性能互扰度。本文将利用层次聚类技术[5]，对已有的虚拟机间性能互扰度模型进行聚类得到虚拟机性能负载模式，从而通过虚拟机性能负载模式匹配获得可以用于虚拟机间性能互扰预测的性能互扰度度量模型，以作为虚拟机间性能互扰预测度预测的基础，实现对虚拟机间性能互扰度的预测。

记已有的虚拟机性能互扰度模型构成集合 $H=\{PID(FW_1@), PID(FW_2@), \cdots, PID(FWn@)\}$，这里，由于在虚拟机性能互扰度模型中背景负载为自变量，因此在模型中只记录目标虚拟机的负载。对于集合 H，根据目标虚拟机负载对历史虚拟机性能互扰度模型进行聚类。为了描述方便，将目标虚拟机负载称为负载模式。

本文采用层次聚类[5]来组织集合 H 中的虚拟机间性能互扰度模型。这里，层次聚类分析的基本思想是：计算虚拟机负载模式之间的相似度，将相似度最高的模式聚集到一簇。这样，在进行层次聚类分析前需要计算虚拟机负载模式之间的相似度。

对于同一应用，虚拟机负载参数具有不同的取值范围，例如，CPU 利用率的取值范围是 0~1，而 I/O 请求的平均等待时间的取值范围是大于 0 ms。由于不同的资源负载参数将具有不同的取值范围，为了保证层次聚类的准确性，需要对这些负载参数进行规范化处理。本文将使用 Z-Score 方法对虚拟机负载参数进行规范化处理。记 $WP=\{wp_1,wp_2,\cdots,wp_n\}$ 为虚拟机负载模式集合，对于其中任意的 wp_i，$wp_i=[x_1, x_2,\cdots, x_m]$，$x_i$ 为虚拟机负载参数。从而，集合 WP 可以转换为式(5)所示的矩阵。

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \tag{5}$$

矩阵的均值可以计算为

$$\overline{x}_j=\frac{1}{n}\sum_i x_{ij} \tag{6}$$

矩阵的标准方差可以计算为

$$s_j=\sqrt{\frac{1}{n-1}\sum_i\left(x_{ij}-\overline{x}_j\right)^2} \tag{7}$$

矩阵的极差可以计算为

$$R_j=\max_i\left\{x_{ij}\right\}-\min_i\left\{x_{ij}\right\} \tag{8}$$

这样，通过 Z-Score 标准变换，对于矩阵 $\boldsymbol{X}$ 中的每个元素 x_{ij} 可以根据式(9)变换为 0~1 之间的一个数。

$$x_{ij}^*=\begin{cases}\dfrac{x_{ij}-\overline{x}_j}{s_j}, & s_j=0\\ 0, \text{ 其他}\end{cases} \tag{9}$$

本文将利用欧氏距离计算 2 个负载模式间的相似度。对于负载模式 wp_1 和 wp_2，根据欧氏距离，它们之间的相似度可以使用式(10)计算。

$$d\left(wp_i,wp_j\right)=\frac{1}{\sqrt{\left(cpuuil_{vm_i}-cpuutil_{vm_j}\right)^2}}+\frac{1}{\sqrt{\left(memuil_{vm_i}-memutil_{vm_j}\right)^2}}+\frac{1}{\sqrt{\left(wps_{vm_i}-wps_{vm_j}\right)^2}}+\frac{1}{\sqrt{\left(await_{vm_i}-await_{vm_j}\right)^2}}+\frac{1}{\sqrt{\left(svctm_{vm_i}-svctm_{vm_j}\right)^2}} \tag{10}$$

其中，wp_i = <$cpuuil_{vmi}$, $memutil_{vmi}$, rps_{vmi}, wps_{vmi}, $await_{vmi}$, $svctm_{vmi}$>；wp_j = <$cpuuil_{vmj}$, $memutil_{vmj}$, rps_{vmj}, wps_{vmj}, $await_{vmj}$, $svctm_{vmj}$>；对于 wp_i 和 wp_j 中的每个元素，其取值为经过 Z-Score 变换后的值。

根据式(10)，$d(wp_i, wp_j)$的取值范围是 0~1。$d(wp_i, wp_j)$越大，表示 wp_i 和 wp_j 之间越相似。

对于负载模式集合 WP，对于其中的任意元素 wp_i，可以计算得到其与其他负载模式间的相似度，从而得到如下的相似度矩阵 $\boldsymbol{D}$。

$$\boldsymbol{D}=\begin{bmatrix} d_{11} & \cdots & d_{1m} \\ \cdots & \cdots & \cdots \\ d_{n1} & \cdots & d_{nm} \end{bmatrix} \tag{11}$$

其中，d_{ij} 表示负载模式 wp_i 和 wp_j 之间的相似度。

根据矩阵 $\boldsymbol{D}$，层次聚类过程可以描述如下。

step1　在负载模式集 WP 中，选择距离最小（即相似度最高）的 2 个负载模式 wp_i 和 wp_j，它们聚集为一簇 wp_k = <$cpuutil_k$, mem_k, rps_k, wps_k, $await_k$, $svctm_k$>，该簇的负载模式计算如式(12)~式(17)所示。

$$cpuutil_k=\frac{\left(cpuutil_i+cpuutil_j\right)}{2} \tag{12}$$

$$memutil_k=\frac{\left(memutil_i+memutil_j\right)}{2} \tag{13}$$

$$rps_k=\frac{\left(rps_i+rps_j\right)}{2} \tag{14}$$

$$wps_k=\frac{\left(wps_i+wps_j\right)}{2} \tag{15}$$

$$await_k=\frac{\left(await_i+await_j\right)}{2} \tag{16}$$

$$svctm_k=\frac{\left(svctm_i+svctm_j\right)}{2} \tag{17}$$

step2　将 wp_k 放入集合 WP 中并通过删除 wp_i 和 wp_j 与其他负载模式的相似度以及增加 wp_k 与其他负载模式的相似度更新矩阵 $\boldsymbol{D}$。

step3　转到 step1 直到在 WP 中只有簇类型的负载模式。

层次聚类结束后，在集合 WP 中将有 $2n-1$ 个负载模式记录，层次聚类的结果可以表示为树状图，如图 3 所示。

图 3 中实线矩形框代表为原始虚拟机模式，在图中称为实节点，虚线矩形框表示簇类型的虚拟机负载模式，称为虚节点。原始虚拟机负载模式与簇类型虚拟机负载模式之间的区别在与前者有与之相应的虚拟机性能互扰度模型，而后者没有。

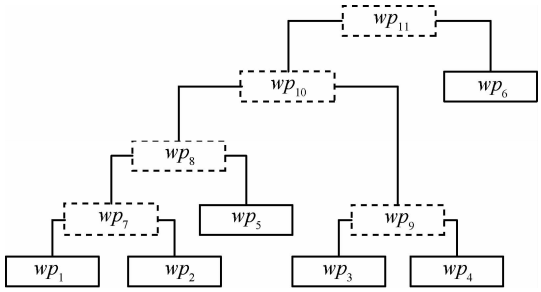


图 3　虚拟机负载模式层次聚类

基于上述思想，给出虚拟机负载模式层次聚类算法。

算法 1　虚拟机负载模式层次聚类算法

输入：负载模式集合 WP_n

输出：负载模式集合 WP_{2n-1}

算法：

1) begin

2) t=1,k=n+1;

3) 生成相似度矩阵 $\boldsymbol{D}$;

4) While t<n　do

5)　　begin

6)　　t = t+1;

7)　　从距离矩阵 $\boldsymbol{D}$ 中选择最小的值 $d(wp_i, wp_j)$且 $i \neq j$ 且(i<=n or j<=n);

8)　　根据式(12)~式(17)生成簇类型负载模式 wp_k 加入集合 WP;

9)　　从集合 D 中删除 wp_i 和 wp_j 与其他虚拟机负载模式的距离，同时添加 wp_k 与其他虚拟机负载模式的距离;

10)　　k = k +1;

11)　　end

12) end

记负载模式集合的规模为 n，生成相似度矩阵 $\boldsymbol{D}$ 的时间复杂度为 $O(n^2)$，生成簇类型负载模式的时间复杂度为 $O(n^3)$，从而算法 1 的时间复杂度为 $O(n^3)$。

在虚拟资源分配问题中，很难在虚拟机部署到物理机之前得到待部署虚拟机与物理机上其他虚拟机间性能互扰历史数据，从而无法直接构建虚拟机间性能互扰度模型并预测虚拟机间的性能互扰

度。本文将讨论如何根据虚拟机间性能互扰度量的历史经验，预测待分配虚拟机与背景虚拟机间的性能互扰度。

记待部署虚拟机 *vm* 的负载模式为 *wp*，虚拟机性能互扰度度量模型构成集合为 $H=\{PID(FW_1@), PID(FW_2@),\cdots,PID(FWn@)\}$，该集合中目标虚拟机负载模式构成集合为 $WP=\{wp_1, wp_2,\cdots, wp_n\}$。本文将通过在 *WP* 中找到与 *wp* 具有最小距离的负载模式，并将该负载模式所对应的虚拟机性能互扰度模型作为 *wp* 的虚拟机性能互扰度模型，从而根据该模型在背景虚拟机负载已知的情况下得到待部署虚拟机在物理机上的性能互扰度。由于在该过程中，需要遍历整个 *WP* 集合以计算 *wp* 与 *WP* 中所有元素的相似度，为了提高遍历效率，本文将根据上节所建立的层次聚类树状图进行负载模式遍历，遍历过程如下。

step1 将聚类树的根节点作为当前节点。

step2 计算 *vm* 与当前节点间的距离 *d*。如果当前节点不是叶子节点且距离 *d* 大于阈值 t_1，将该节点插入到队列 *Q*；否则，如果当前节点是叶子节点且距离 *d* 大于阈值 t_2 且距离 *d* 不等于 1，将该节点插入队列 *Q* 并将该节点放入集合 *R*；否则，如果当前节点是叶子节点且距离 *d* 等于 1，删除 *R* 中所有元素，将该节点放入集合 *R* 并返回结果；否则如果 *d* 是叶子节点并且距离 *d* 不大于阈值 t_2，将该节点插入队列 *Q*；否则，转到 step3。

step3 从 *Q* 中取一个节点 *n*，如果节点 *n* 还有未被计算与 *vm* 相似度的兄弟节点，则取一个这样的兄弟节点为当前节点并转到 step2；否则，转到 step4。

step4 如果节点 *n* 还有未被计算与 *vm* 相似度的孩子节点，则取一个这样的孩子节点为当前节点并转到 step2；否则转到 step5。

step5 从 *Q* 中取一个节点并转到 step2，直到 *Q* 为空。

根据上述过程，可以得到集合 *R*，这里 *R* 中的元素具有和 *vm* 最近的距离，即 *R* 中的元素与 *vm* 最相似。如果在 *R* 中只存在一个元素 *e*，为了对 *vm* 与其他虚拟机间的性能互扰度进行预测，可以直接使用元素 *e* 的虚拟机性能互扰度度量模型。如果在集合 *R* 中有多个元素，将使用式(18)对 *vm* 与其他虚拟机间的性能互扰度进行预测。

$$PID(FW@BW)=\sum_i\left(\frac{d_i}{sumd}\right)\times PID(FW_i@BW) \quad (18)$$

其中，*FW* 是虚拟机 *vm* 的负载模式；FW_i 是集合 *R* 中元素 e_i 的负载模式；d_i 是虚拟机 *vm* 的负载模式与元素 e_i 的负载模式间的距离；$sumd=\sum_{j=1}^{n}d_j$ 为距离之和。

基于上述的基本思想，算法 2 给出了在背景虚拟机负载模式已知的情况下，虚拟机 *vm* 与同一物理机上其他虚拟机间性能互扰度的预测算法。

算法 2 虚拟机间性能互扰度预测算法

输入：待部署虚拟机 *vm* 的负载模式 *FW*，背景虚拟机负载模式 *BW*，虚拟机负载模式层次树 *HierarchicalTree*

输出：互扰度 *pid*

算法：

1) begin
2) 遍历 *HierarchicalTree* 得到集合 *R*;
3) If 如果在集合 *R* 中仅存在一个元素 *e* then
4) *pid*= *PID*(*e*@*BW*);
5) else if 在集合 *R* 中没有元素 then
6) *pid*=−1;
7) else for each element e_i in *R* do
8) begin
9) $pid_i=PID(e_i@BW)$;
10) $pid=pid+d_i\times pid_i/sumd$;
11) end
12) return *pid*;
13) end

记 *HierarchicalTree* 的高度为 *k*，生成集合 *R* 的时间复杂度为 *O*(*k*)；记集合 *R* 的规模为 *m*，计算性能互扰度的时间复杂度为 *O*(*m*)，从而算法 2 的时间复杂度为 *O*(*k*+*m*)。

3.3 考虑虚拟机间性能互扰的虚拟资源分配算法

考虑虚拟机间性能互扰的虚拟资源分配是要在虚拟资源分配时，降低共享同一物理机的虚拟机间的性能互扰程度，进而保证虚拟机性能。

记用户的虚拟资源需求为 *vm*，可用物理机集合为 $PM=\{pm_1, pm_2, \cdots, pm_n\}$，虚拟机资源分配问题的目标（式(19)给出了该问题的目标函数）为：在可用物理机集合 *PM* 中，为 *vm* 找到一个可放置的物理机 PM_i，使该物理机的可用资源满足 *vm* 的需求并且在该虚拟机放置到物理机 PM_i 后，该虚拟机

与物理机上其他虚拟机间的性能互扰度最小。

$$\min_j PID\left(vm.FW @ PM_j.BW\right)VM, PM_j$$

满足

$$\begin{aligned} &PM_j.CPUavail \geqslant vm.CPU \\ &PM_j.MEMavail \geqslant vm.MEM \\ &PM_j.DISKavail \geqslant vm.DISK \end{aligned} \tag{19}$$

其中，*vm.CPU* 为待部署虚拟机的 CPU 资源需求；*vm.MEM* 为待部署虚拟机的内存资源需求；*vm.DISK* 为待部署虚拟机的磁盘资源需求；$PM_j.CPUavail$ 为物理机 PM_j 的可用 CPU 资源；$PM_j.MEMavail$ 为物理机 PM_j 的可用内存资源；$PM_j.DISKavail$ 物理机 PM_j 的可用磁盘资源；*vm.FW* 为 *vm* 的平均负载；$PM_j.BW$ 为物理机 PM_j 上 *vm* 的背景负载。

对式(19)的问题求解可以通过比较虚拟机在不同物理机上的性能互扰度得到性能互扰度最小的物理机作为虚拟机放置对象。限于篇幅原因在此不进行详细描述。

4　实验

为了验证本文所提出的虚拟机性能互扰和预测算法以及考虑虚拟机间性能互扰的资源分配方法的有效性，开展了一系列的实验。

实验中所使用的物理机配有 Intel Core i3 2120 3.3 G 处理器、4 G DDR3 内存以及 250 G 7 500 rpm 的硬盘，其 OS 版本为 ubuntu 12.04。实验利用 Xen 虚拟系统提供虚拟化支持，Xen 虚拟机配置为 4 个 VCPU、1 G 内存以及 8 G 硬盘，操作系统为 ubuntu 12.04。

实验 1　虚拟机性能互扰度模型的有效性验证

在本文设计的实验中，物理机上承载 2 个虚拟机，即位于 Domain1 域的 VM_1 与位于 Domain2 的 VM_2，实验中每个虚拟机可运行一个应用。实验以虚拟机 VM_1 作为目标虚拟机研究其性能互扰度，虚拟机 VM_2 上运行其他各类应用作为 VM_1 的背景负载，并从常用工具和标准测试套件中选取各种应用。表 5 给出了 Bzip2 的性能互扰度预测结果与真实运行监测到的性能互扰度。

由表 5 可知，Bzip2 的性能互扰度预测平均误差为 10%，最小误差为 0.7%，最大误差为 29.8%，当负载参数 *rps* 和 *wps* 值较大时，性能互扰度也较大，其预测结果误差也较大。这说明 *rps*（或者 *wps*）与性能互扰度之间的关系不是严格的线性关系。在下一步研究工作中，将采用非线性回归方法分析它们与性能互扰度之间的关系，以提高虚拟机间性能互扰度预测的准确性。

将实验环境中的应用 Bzip2、cat、Iozone、Super PI、Ccrpyt 和 Gzip 作为目标虚拟机应用，训练所构建的性能互扰度模型。改变背景虚拟机负载，记录根据构建的性能互扰度模型所预测的性能互扰度预测值和实际观测值，在不同背景负载下，比较性能互扰度预测平均值和观测平均值。实验结果如图 4 所示。

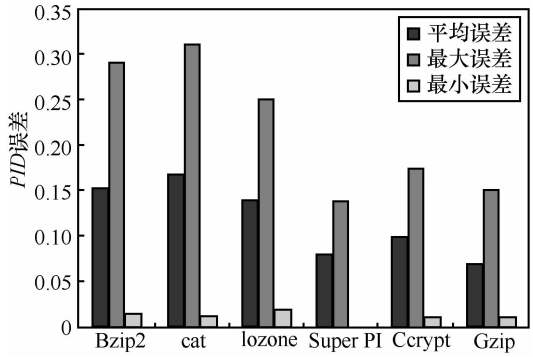


图 4　虚拟机间性能互扰度量误差

表 5　　**Bzip2 的性能互扰度预测值与真实结果**

rps	*wps*	*await*	*svctm*	predicted *PID*	Observed *PID*	*error*
141.238	121.428	5.765	1.973	0.398 6	0.401 4	0.7%
175.498	31.716	3.634	2.033	0.189 1	0.198	4.5%
264.861	251.134	5.751	5.452	0.439 2	0.385 7	13.9%
0.762	0.468	9.172	5.874	0.077 4	0.080 2	3.5%
7.156	65.821	7.844	2.165	0.314 5	0.301 7	4.2%
264.894	335.418	38.551	2.968	0.731 9	1.042 9	29.8%
568.563	142.745	67.854	1.254	0.351 4	0.453 9	22.6%
10.495	3.154	24.264	3.746	0.138 8	0.137 2	1.2%

由图 4 可以看出，I/O 密集型应用（如 cat 和 Iozone）和混合型应用（如 Bzip2 和 Gzip）的度量误差均高于 CPU 密集型的应用（如 Super PI）。虽然在某些情况下预测误差较大，但从平均误差来看，各应用性能互扰度预测平均误差介于 7%~18%，具有较好的预测准确性。

实验 2　虚拟机性能互扰度预测算法的准确性

对表 1 中的应用使用层次聚类算法进行聚类，结果如图 5 所示。

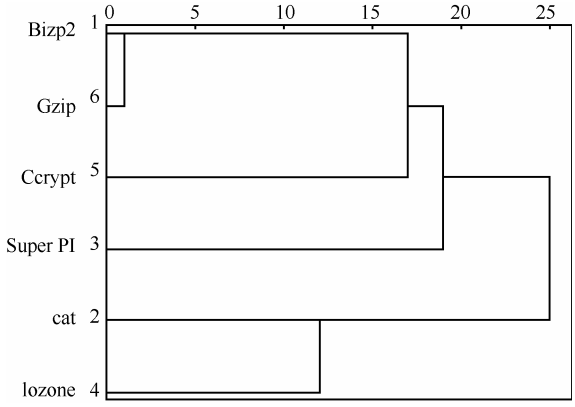


图 5　聚类结果

从图 5 可看出，Iozone 和 cat 等 disk I/O 密集型应用聚为一簇，Bzip2 和 Ccrypt 等混合类型应用聚为一簇。聚类结果与事先已知的各个应用的资源使用特点相符。

为了验证基于虚拟机间性能互扰度历史经验复用的性能互扰度预测算法的有效性，本文使用应用 make 作为目标虚拟机，该应用为混合类型应用，其负载模式为<0.98, 0.03, 10.11, 35.73, 2.19, 1.56>，表 6 给出了该应用与图 5 中各个应用之间的距离。

表 6　make 与其他应用负载模式间的距离

Bzip2	cat	Super PI	Iozone	Ccrypt	Gzip
0.549	0.229	0.515	0.216	0.337	0.526

实验中相似距离阈值设置为 0.5，应用 Bzip2、Gzip 和 Super PI 的性能互扰度模型将用于预测 make 的性能互扰度，即 make 的性能互扰度为

$$\begin{aligned} &PID(\text{make}@BW) \\ &= 0.345\times PID(\text{Bzip2}@BW)+ \\ &0.324\times PID(\text{SuperPI}@BW)+ \\ &0.331\times PID(\text{Gzip}@BW) \end{aligned} \tag{20}$$

改变背景虚拟机负载，并利用式(20)计算应用 make 与背景虚拟机的性能互扰度，比较性能互扰度预测值与实际值间的误差。实验结果如图 6 所示。

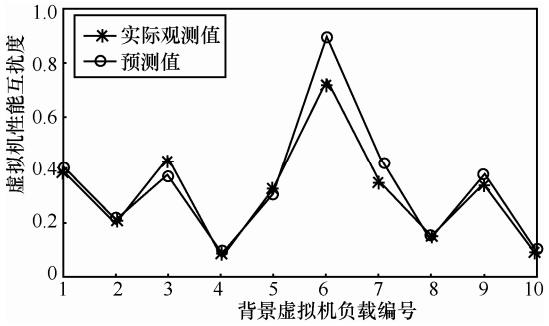


图 6　make 性能互扰度预测值与实际值

由图 6 可知，make 应用的性能互扰度预测值基本上接近观测值。

另外，分别将应用 Cp，Dd 和 add_double 作为目标虚拟机所承载的应用，在没有这些虚拟机与背景虚拟机性能互扰历史数据的情况下，利用本文所提出的虚拟机性能互扰度预测算法计算性能互扰度并与实际观测值进行比较。实验结果如图 7 所示。

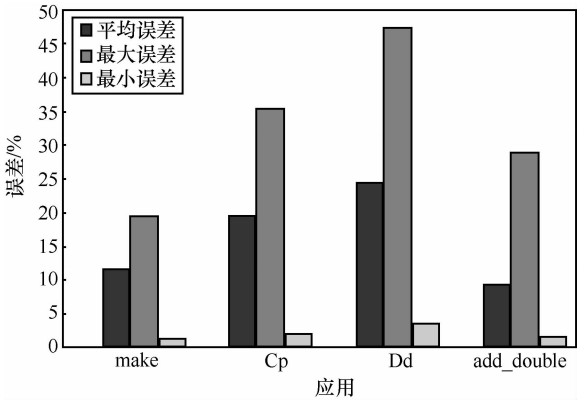


图 7　应用性能互扰度预测误差

由图 7 可知，应用 make、Cp、Dd 和 add_double 性能互扰度预测的平均误差范围在 9.25%~25.8%之间。

实验 3　考虑虚拟机间性能互扰的虚拟资源分配方法的有效性

为了验证虚拟机互扰度量和预测方法在资源分配中的有效性，将采取 3 种不同的资源分配方案，其中，方案 1 是随机选择可用虚拟资源满足虚拟机资源需求的物理机作为虚拟机部署对象；方案 2 是采取文献[6]的方法对虚拟机性能间性能互扰进行预测并以此作为虚拟资源分配依据的方法；方案 3

是采取本文所提出的虚拟机间性能互扰度预测方法对满足虚拟机资源需求的物理机进行评价并选择性能互扰度最低的物理机作为虚拟机部署对象。实验使用 12 台主机，机器配置均相同，采用 Intel Core i3 2120 3.3 G 处理器、4 G DDR3 内存以及 250 G 7 500 rpm 的硬盘，其 OS 版本为 ubuntu 12.04。实验利用 Xen 虚拟系统提供虚拟化支持，Xen 虚拟机配置为 4 个 VCPU、1G 内存以及 8 G 硬盘。采用方案 1、方案 2 与方案 3 针对不同类型的应用部署需求，分别进行虚拟资源分配，比较平均标准执行时间（标准执行时间为应用部署到物理机后的实际执行时间与应用在独占物理机情况下的执行时间之比）进行对比实验，其结果如图 8 所示。

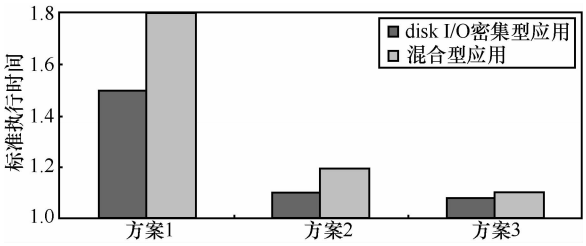


图 8　3 种方案下的实验结果对比

由图 8 可知，方案 3 的标准执行时间接近于 1，比方案 1 和方案 2 的标准执行时间小，表示方案 3 的执行时间更接近于虚拟机在无性能互扰情况下的执行时间。而方案 1 在一般情况下具有较大的标准执行时间。这主要是因为方案 1 在虚拟资源分配中并未考虑虚拟机间性能互扰的影响，进而容易导致所部署应用性能下降。当应用类型为 I/O 密集型应用时，方案 2 通常具有较好的标准执行时间，这主要是因为方案 2 所依据的评价虚拟机间性能互扰度模型主要是面向 I/O 密集型应用的，而对于混合型应用，虚拟机性能互扰度具有较大的误差。

上述实验说明本文所提出的虚拟机间性能互扰度量及预测方法的有效性，同时也表明考虑虚拟机间性能互扰的资源分配方法的有效性。

5　相关工作

在虚拟机资源分配以及虚拟机间性能互扰方面，目前已经开展了大量的研究工作。

文献[7]提出了一个负载相关的部署策略，该策略基于负载预测的结果将负载峰值错开的虚拟机部署到同一节点，以提高系统资源利用率。文献[8]提出了一种基于典型相关性分析(CCA)的应用分析技术，建立反映目标应用的性能和所有虚拟机资源使用模式间关系的应用性能预测模型，并在虚拟机部署之前进行预测以降低虚拟机间的影响，文献[9]提出了一种基于整数规划度量性能互扰的虚拟机放置方法，该方法用于降低由于 network I/O 造成的虚拟机互扰。文献[6]研究了同一主机上一对虚拟机间的性能互扰，并利用回归分析模型建立应用性能与系统级参数之间的关系，以对应用性能进行预测。文献[10]在前者的基础上提出了一个任务和资源分配控制框架（TRACON, task and resource allocation control framework），该框架通过 3 个组件（性能预测模型、基于性能度量的调度器和资源监控器）降低数据密集型应用之间的互扰并提高系统的整体性能。文献[11]使用约束满足问题对虚拟机部署进行建模，约束条件为用户的服务等级协议，目的是最大化节省能耗，其实现思想是最大化空闲主机数，通过关闭空闲主机来节省能源。文献[12]提出了一个 2 层的控制系统来解决任务和虚拟机之间以及虚拟机和服务器间之间的映射，该系统目标是在提高系统资源利用率的同时，降低能耗和热耗散。

上述的方法在虚拟资源分配方面，文献[11]和文献[12]并未考虑虚拟机间性能互扰对资源分配结果有效性的影响，容易导致虚拟机间由于性能互扰而导致性能下降的问题。文献[7]考虑到了虚拟机间的性能互扰并采用错峰资源分配的方法降低所部属虚拟机对物理机上背景虚拟机的性能互扰，但是仅考虑了 CPU 利用率这一负载参数，由于虚拟机间性能互扰受多个负载参数的影响，该方法仅适用于 CPU 密集型应用间的资源分配。文献[6,9,10]建立了虚拟机性能互扰度量模型，但是所构建的这些模型都是基于历史虚拟机间性能互扰度量经验建立的，并针对于特定类型的应用，例如，文献[6]建立了一个针对于 disk I/O 密集型应用的性能互扰度量模型，并通过该模型对 disk I/O 密集型应用的性能进行预测。但是，在虚拟资源分配问题中，无法事先获得待部署应用与物理机间的性能互扰历史数据，难以直接构建性能互扰度量模型进行性能互扰度的预测；同时待部署应用可能是混合类型的应用，而不同混合类型的应用资源使用模式是不同的，无法构建面向混合类型应用的通用的虚拟机性能互扰度量模型。这就需要针对虚拟资源分配中，虚拟机性能互扰度量问题的这一特点提出面向虚

拟资源分配的虚拟机间性能互扰度量及预测方法。

6 结束语

本文提出了一种考虑虚拟机间性能互扰的虚拟资源分配方法。该方法以虚拟机间性能互扰度量和预测为基础，将对虚拟机间性能互扰的预测结果应用于资源分配方案性能互扰评价中，以作为资源分配方案选取和调整的依据，保证了资源分配方案的有效性。针对于虚拟机间性能互扰度量问题，建立了一种虚拟机性能互扰度度量模型，以反映在不同负载情况下虚拟机间性能互扰程度，并利用多元线性回归实现对该模型的参数化求解。针对于虚拟机间性能互扰度预测问题，提出一种面向虚拟资源分配的虚拟机间性能互扰度预测算法，以通过对虚拟机间性能互扰度量历史经验的复用，对资源分配方案中待部署虚拟机与物理机上背景虚拟机间的性能互扰度进行预测。本文并未考虑虚拟机间对网络 I/O 资源的抢夺问题，下一步将对该问题进行研究，以提高方法的适用性。同时，下一步将利用非线性回归分析方法建立虚拟机负载与性能互扰度间的关系，以提高虚拟机性能互扰度度量的准确性。

参考文献：

[1] 罗军舟, 金嘉晖, 宋爱波. 云计算—体系架构与关键技术[J]. 通信学报, 2011,32(7):3-21.
LUO J Z, JIN J H, SONG A B. Cloud computing: architecture and key technologies[J]. Journal on Communications, 2011, 32(7):3-21.

[2] CHEN M，ZHANG H, SU Y Y. Effective VM sizing in virtualized data centers[A]. 12th IFIP/IEEE International Symposium on Integrated Network Management (IM'2011)[C]. Dublin, Ireland, 2011.594-601.

[3] LEITE D, PEIXOTO M, SANTANA M. Performance evaluation of virtual machine monitors for cloud computing[A]. Symposium on Computer Systems (WSCAD-SSC'2012)[C]. De outubro, Brasil, 2012. 65-71.

[4] HIROTOGU A. A new look at the statistical model identification[J]. IEEE Transactions on Automatic Control, 1974, 19(6):716-723.

[5] ULRIKE L. A tutorial on spectral clustering[J]. Statistics and Computing, 2007,17 (4):395-476.

[6] KOH Y, KNAUERHASE R, BRETT P. An analysis of performance interference effects in virtual environments[A]. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS' 2007)[C]. San Jose, California, USA, 2007. 200-209.

[7] 潘飞, 蒋从锋, 徐向华. 负载相关的虚拟机放置策略[J]. 小型微型计算机系统, 2013, 34(3): 520-524.
PAN F, JIANG C F, XU X H. Placement strategy of virtual machines based on workload characteristics[J]. Journal of Chinese Computer Systems, 2013,34(3): 520-524.

[8] ANH V D, JUNLIANG C, CHEN W. Profiling applications for virtual machine placement in clouds[A]. 4th IEEE 2011 International Conference on Cloud Computing (CLOUD 2011)[C]. Honolulu, Hawaii, USA, 2011.660-667.

[9] LIN J W, CHEN C H. Interference-aware virtual machine placement in cloud computing system[A]. International Conference on Computer & Information Science (ICCIS'2012)[C]. Kuala Lumpur, Malaysia, 2012.598-603.

[10] CHIANG R C, HUANG H H. TRACON: interference-aware scheduling for data-intensive applications in virtualized environments[A]. International Conference for High Performance Computing, Networking, Storage and Analysis(SC'2011)[C]. New York, NY, USA, 2011. 1 - 12.

[11] VAN H N, TRAN F D, MENAUD J M. Performance and power management for cloud infrastructures[A]. IEEE International Conference on Cloud Computing (CLOUD'2010)[C]. Miami, USA, 2010. 329-336.

[12] XU J, FORTES J A B. Multi-objective virtual machine placement in virtualized data center environments[A]. IEEE/ACM Int'l Conference on Green Computing and Communications (GreenCom'2010) & Int'l Conference on Cyber, Physical and Social Computing (CPSCom'2010)[C]. Hangzhou, China, 2010. 179-188.

作者简介：

杨雷（1974-），男，辽宁营口人，博士，东北大学副教授，主要研究方向为云计算、服务计算、服务性能管理、海量数据处理及分析。

邢禾续（1990-），男，黑龙江大庆人，东北大学硕士生，主要研究方向为云计算、海量数据处理及分析。

代钰[通信作者]（1980-），女，辽宁沈阳人，博士，东北大学副教授，主要研究方向为云计算、服务计算、服务性能管理。E-mail:daiyu@mail.neu.edu.cn。

徐赛（1989-），男，河南商丘人，东北大学硕士生，主要研究方向为云计算、服务性能管理。

王昊（1992-），男，山东济宁人，东北大学硕士生，主要研究方向为云计算、服务性能管理。

张斌（1964-），男，辽宁本溪人，东北大学教授、博士生导师，主要研究方向为云计算、服务性能管理。