多虚拟机实时迁移中自适应的迁移算法选择框架

崔 勇[1,2] 林予松[2] 刘 炜[2] 高 山[1,2] 王宗敏[2]
(郑州大学信息工程学院 郑州 450001)[1] (郑州大学河南省信息网络重点开放实验室 郑州 450052)[2]

摘 要 IaaS云计算平台中主要通过实时迁移多台虚拟机来实现资源的动态调度、管理与优化。虽然Pre-copy和Post-copy是单虚拟机实时迁移的两种主流算法,且各有优缺点,但现有的多虚拟机实时迁移系统只是单一地使用其中一种迁移算法,无法根据各虚拟机的不同负载情况灵活选择最有效的迁移算法,降低了整体迁移效率。提出一种自适应的实时迁移算法选择框架,利用模糊聚类方法对待迁移的多虚拟机进行分类,按类别选择最适合的迁移算法。实验结果表明,所提出的迁移算法选择框架能够在多虚拟机实时迁移中发挥两个迁移算法的各自优势,有效提高整体的实时迁移性能。

关键词 实时迁移,虚拟机,迁移算法选择框架,Pre-copy,Post-copy
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.8.012

Adaptive Migration Algorithm Choosing Framework in Live Migration of Multiple Virtual Machines

CUI Yong[1,2] LIN Yu-song[2] LIU Wei[2] GAO Shan[1,2] WANG Zong-min[2]
(Institute of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)[1]
(Henan Provincial Key Lab on Information Networking, Zhengzhou University, Zhengzhou 450052, China)[2]

Abstract In IaaS cloud computing platform, live migration of multiple virtual machines plays a dominant role in the dynamic scheduling, optimization and management of IT resources. Although Pre-copy and Post-copy are the prevalent live migration algorithms for the single virtual machine, which both have the pros and cons, only one of them is monotonously adopted in the context of the gang of live migration. This scheme cannot choose the best migration algorithm for each virtual machine according to its overload, eventually degrading the whole migration performance. This paper proposed an adaptive live migration algorithm choosing framework, which uses fuzzy clustering method to classify the virtual machines to be migrated and migrates them with the chosen optimum migration algorithm. Experiment results show that the proposed framework can exert each advantage of the two basic migration algorithms and improve the whole live migration performance.

Keywords Live migration, Virtual machine, Migration algorithm choosing framework, Pre-copy, Post-copy

1 引言

IaaS(Infrastructure As A Service)云计算平台采用虚拟机实时迁移技术[1]来对以虚拟机为单元的数据中心资源进行动态调度、管理及优化。其可在保证虚拟机中负载连续运行的情况下将虚拟机的状态数据(如虚拟机内存镜像、vCPU寄存器)及持久数据(如虚拟磁盘)迁移到其他宿主机上,并切换到该宿主机上继续运行。由于现代数据中心普遍采用共享存储架构,虚拟机持久数据迁移相对简单,因此实时迁移的对象主要是虚拟机的状态数据,特别是内存镜像[2]。主流的实时迁移算法包括预拷贝(Pre-copy)[3]和后拷贝(Post-copy)[4]两种,其中Pre-copy最先被广泛应用于开源及商用的虚拟机管理器(如Xen、KVM、Vmware vSphere),而Post-copy是针对Pre-copy的不足而提出的另一种解决方案,可以说两者各有优劣,适用于不同的场景。Pre-copy在高脏页率情况下存在脏页重传问题[5],不适合迁移运行写密度型负载的虚拟机;Post-copy在频繁读取页面时会产生大量网络缺页中断[6],不适合迁移运行读密度型的虚拟机。

虚拟机实时迁移是IaaS云平台实现服务器在线维护、自动在线负载均衡、自动在线电源管理等高级功能的基础。这些应用一般都需要同时迁移多台虚拟机,而多虚拟机实时迁移对生产业务的影响及对计算、网络资源的占用是不容忽视的[7],因此多虚拟机实时迁移的整体性能关系到IaaS云平台的实际运行效率。虽然单虚拟机实时迁移有两个较为成熟的

到稿日期:2015-07-27 返修日期:2015-11-02 本文受教育部博士点专项科研基金(20114101110007),河南省科研重点项目(13A520562),河南省创新人才项目(2011HASTIT003),河南省高等学校重点科研项目(15A520028),河南省基础与前沿技术研究项目(152300410047)资助。
崔 勇(1983-),男,博士生,CCF学生会员,主要研究方向为虚拟化技术、云计算,E-mail:goodaliens@126.com;林予松(1973-),男,博士,副教授,CCF会员,主要研究方向为下一代互联网、互联网医疗;刘 炜(1981-),男,博士,讲师,主要研究方向为网络安全、网络性能研究;高 山(1977-),女,博士生,副教授,主要研究方向为个性化推荐;王宗敏(1964-),男,博士,教授,主要研究方向为下一代互联网(通信作者)。

迁移算法，但是目前多虚拟机实时迁移中却普遍只采用单一的迁移算法来迁移运行不同负载的虚拟机[7-10]，无法保证各个虚拟机的最优迁移方式，缺乏灵活性和运行效率，进而影响整体迁移性能。本文首次针对该问题提出一种自适应的迁移算法选择框架，以虚拟机的内存读、写速率为度量标准，基于模糊聚类方法将待迁移虚拟机分类，根据类别选择最有效的迁移算法，取长补短、因地制宜地发挥两个迁移算法的作用，以达到提高多虚拟机实时迁移整体性能的目的。

2 实时迁移算法介绍

虚拟机实时迁移的性能指标主要包括总迁移时间(Migration time)和停机时间(Downtime)。其中总迁移时间指从开始迁移虚拟机到迁移结束(即虚拟机能够在目标宿主机上正常运行)的时间；停机时间指虚拟机切换到目标宿主机时一个短暂的暂停时间，是总迁移时间的一部分。实时迁移算法的性能评价准则是：这两个时间越少，迁移性能越高，也就是对数据中心计算、网络资源的占用越少，同时对虚拟机内业务的影响越小。

实时迁移算法总体上可分为3个阶段。

• 拷贝阶段(Copy phase)：在虚拟机正常运行时，将内存页面数据传输到目标宿主机；

• 停机阶段(Stop phase)：暂停虚拟机，并将设备状态数据(如vCPU寄存器)同步到宿主机；

• 恢复阶段(Resume phase)：在宿主机上配置运行环境，将虚拟机运行状态完全切换过来。

其中拷贝阶段的传输方式可分为以下两类。

• 推拷贝(Push copy)：源宿主机主动向目标宿主机传输页面数据；

• 拉拷贝(Pull copy)：目标宿主机请求源宿主机向其传输页面数据。

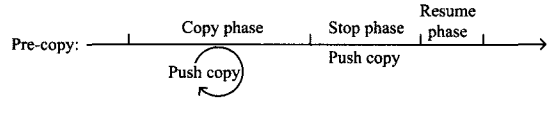

图1 Pre-copy算法流程

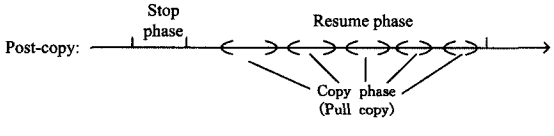

图2 Post-copy算法流程

图1、图2分别显示了主流实时迁移算法Pre-copy和Post-copy的流程，两者在阶段次序及传输方式上不同。Pre-copy首先进行内存页面数据传输，采用迭代Push copy的方式不断同步内存脏页，待剩余脏页足够小或达到固定迭代次数时进入停机阶段，并将剩余脏页及设备状态传输过去，最后在一个短暂的恢复阶段将虚拟机切换完毕。Post-copy在迁移开始就进入停机阶段，待设备状态数据传输过去后立即在宿主机恢复虚拟机运行，之后若访问不存在的页面，以网络缺页中断的Pull copy方式向源主机请求页面，获取页面后虚拟机继续运行。

两个算法都存在局限性：Pre-copy由于不断在两端同步脏页，当虚拟机脏页率较高时存在某些页面多次重复传输的问题，增加了总迁移时间，且若此时网络带宽较低，则在停机阶段还需要传输大量剩余脏页，增加了停机时间；Post-copy虽然使得内存页面最多传输一次，但当虚拟机访问页面频率较高时会产生大量网络缺页中断，增加总迁移时间，且若此时网络带宽较低，则页面获取时间较长，影响了虚拟机用户的使用体验。因此，Pre-copy不适合迁移写密度型的虚拟机，而Post-copy不适合迁移读密度型的虚拟机[4]。

3 基于模糊聚类的自适应迁移算法选择框架

在多虚拟机实时迁移中，为了根据虚拟机负载类型采用最优实时迁移算法，进而提高整体实时迁移性能，本文提出一个自适应的迁移算法选择框架，即首先采用改进的模糊聚类方法对虚拟机分类，然后根据虚拟机类型选择最适合的迁移算法。

3.1 多虚拟机模糊聚类

本文采用模糊C均值聚类方法(FCM)[11]，根据读、写内存页面的速率对待迁移的虚拟机进行分类，并通过改进FCM中初始聚类中心的方法来提高分类效率。

FCM是一种无监督学习的模糊分类方法，其根据样本数据对某一分类的隶属度进行样本集合的软划分，通过求解所有样本与其所属分类的聚类中心的欧氏距离之和最小这一优化问题，得到最优隶属度矩阵及聚类中心，从而确定样本分类。

在多虚拟机模糊聚类中，设 n 个虚拟机的数据样本集合为 $X=\{x_1,x_2,\cdots,x_n\}$，其中 $x_j=(R_j,W_j)^{\mathrm{T}}$ 为虚拟机 VM_j 的特征向量，记录其内存读、写速率。将虚拟机划分为读密度型、写密度型、读写复合密度型3类，则分类数 c 为3，第 i 个分类的聚类中心为 c_i。令 $U=(\mu_{ij})_{c\times n}$ 为样本隶属度矩阵，$\mu_{ij}\in[0,1]$ 为 VM_j 属于第 i 个分类的隶属度，则最优分类结果即最优隶属度矩阵和聚类中心可以通过求解以下优化问题得到：

$$\min J(U,c_1,c_2,c_3)=\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{m}d_{ij}^{2}$$

$$\text{s.t.}\sum_{i=1}^{c}\mu_{ij}=1,\forall j$$

其中，m 为模糊分类指数，即控制样本与聚类中心的分离程度，这里取值为2。$d_{ij}=\|c_i-x_j\|$ 为数据样本 j 与第 i 个聚类中心间的欧氏距离。采用拉格朗日乘子法对以上带约束的优化问题进行求解，可分别得到隶属度矩阵和聚类中心的计算公式[11]：

$$\mu_{ij}=\left[\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{2}{m-1}}\right]^{-1},\forall i,j \tag{1}$$

$$c_i=\frac{\sum_{k=1}^{n}(\mu_{ik})^{m}x_k}{\sum_{k=1}^{n}(\mu_{ik})^{m}},\forall i \tag{2}$$

FCM算法首先初始化隶属度矩阵和聚类中心，然后根据式(1)、式(2)对两者进行迭代修正，直至结果收敛。

标准的FCM算法采用随机初始化聚类中心的方法，存在易使迭代收敛到局部最小值的问题，导致聚类效果不佳[12]。对此，本文对初始聚类中心方法进行改进，根据拟划分的3种虚拟机类型及样本数据极值构造出聚类中心参考

点，通过计算样本点与聚类中心参考点的余弦相似度找到具有最佳分类特征的样本作为聚类中心。用余弦相似度方法即式(3)计算两个向量 $A(a_1,a_2)$、$B(b_1,b_2)$ 的相似程度，余弦值越接近 1，则两个向量越相似。

$$\cos\theta(A,B)=\frac{a_1a_2+b_1b_2}{\sqrt{a_1^2+b_1^2}+\sqrt{a_2^2+b_2^2}} \tag{3}$$

改进的初始聚类中心选择方法如下：

1)在所有样本中分别找到最大内存读、写速率值，记为 R_{max} 及 W_{max}。

2)定义对应读密度型、写密度型及读写复合密度型 3 个分类的聚类中心参考点分别为向量 $CR=(R_{max},0)^T$，$CW=(0,W_{max})^T$ 和 $CRW=(R_{max},W_{max})^T$。

3)计算所有样本与 CR 的余弦相似度，取余弦值最高的样本作为读密度类型的初始聚类中心。

4)除去上步找出的样本点，计算其他样本与 CW 的余弦相似度，取余弦值最高的样本作为写密度类型的初始聚类中心。同样，通过计算样本与 CRW 的余弦相似度找到读写复合密度类型的初始聚类中心。

3.2 自适应选择迁移算法

根据以上分类结果，待迁移的各虚拟机按所属类型自适应地确定迁移算法，如图 3 所示。其中读密度型采用 Pre-copy，写密度型采用 Post-copy，而读写复合密度型由于其读、写速率总体上相近，还需进一步考虑正式进行实时迁移时段的读、写速率情况。

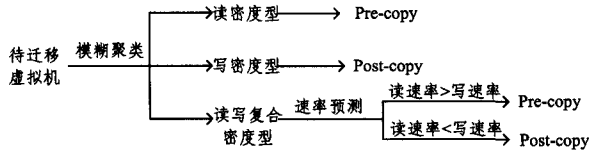

图 3 自适应选择迁移算法流程

在启动实时迁移任务后，首先对各虚拟机内存读、写速率进行统计与记录。如图 4 所示，对某个虚拟机，在一个 T 时间内以 P 为周期按时间顺序分别统计并记录下各个 P 时间内的读、写速率。上节模糊聚类方法中采用的读、写速率分别是记录的 T 时间内读、写速率的算术平均值。对于读写复合密度类型，基于读、写速率构成的时序记录，利用指数平滑法[13]预测正式进行迁移时段的读、写速率值，通过比较两个预测值来选择迁移算法。

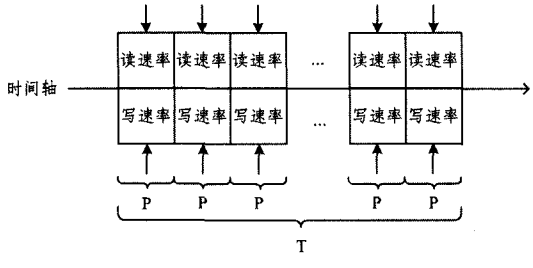

图 4 虚拟机内存读、写速率统计方法

设待迁移的某个虚拟机的内存读速率记录构成的时间序列为 $R=\{r_1,\cdots,r_n\}$，内存写速率记录构成的时间序列为 $W=\{w_1,\cdots,w_n\}$，则读、写速率在正式迁移时段的预测值 R_r、W_p 分别由以下递归公式得到：

$$R_p=ar_n+(1-\alpha)R_{n-1} \tag{4}$$

$$W_p=aw_n+(1-\alpha)W_{n-1} \tag{5}$$

其中，R_{n-1}，W_{n-1} 分别为 r_{n-1}，w_{n-1} 的平滑指数值，α 为平滑参数，这里取为 0.3。将式(4)展开可得

$$R_p=\sum_{i=1}^{n}\alpha(1-\alpha)^{n-i}d_i+(1-\alpha)^nR_0 \tag{6}$$

其中，R_0 为平滑初值，这里取序列 R 的算术平均值。

同样，将式(5)展开可得到 W_p。如图 3 所示，若 $R_p>W_p$，说明虚拟机在进入正式迁移阶段时主要进行内存读操作，选择 Pre-copy 算法；若相反，则说明虚拟机在进入正式迁移阶段时主要进行内存写操作，选择 Post-copy 算法。

3.3 框架实现

本文基于开源虚拟机管理器 QEMU-KVM 2.1.5[14] 实现了自适应的迁移算法选择框架，如图 5 所示，主要包括 3 个模块：虚拟机聚类模块、迁移算法选择模块和读、写速率跟踪模块。

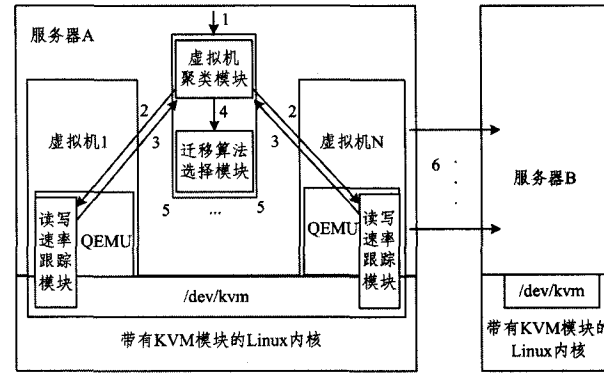

图 5 自适应的迁移算法选择框架实现架构图

QEMU-KVM 平台下每个虚拟机运行在一个独立的 QEMU 程序进程中，而框架的前两个模块的实现分别对应于 3.1 节、3.2 节的内容。由于需要与多虚拟机进行通信，因此采用 C 语言编码，以一个独立于各 QEMU 进程的新进程运行。其中与 QEMU 进程通信的功能基于 Libvirt API 实现，而由于 QEMU-KVM 2.1.5 及以上版本在支持原有的 Pre-copy 算法的基础上又可选用 Post-copy 算法进行实时迁移，因此在迁移算法选择模块中选择迁移算法时，只需直接调用 QEMU-KVM 提供的迁移命令即可。

读、写速率跟踪模块运行在每个 QEMU 进程中，负责监测虚拟机内存页面的读、写操作，统计次数，计算出读、写速率并对其进行记录。该模块的实现代码分布在 QEMU 用户态模块及 KVM 内核模块中：在 QEMU 模块中运行一个计时器，按图 4 所示的方法，在 $T=10s$ 时间内，以 $P=1s$ 为周期调用 KVM 内核模块函数进行内存页面读、写次数的统计，然后将统计值返回到 QEMU 用户态空间中以计算出读、写速率，最后分别将其记录在两个 double 类型的数组中。

针对在内核模块中捕获对内存页面的读写操作，考虑到运行效率及实现复杂度，利用硬件辅助的内存虚拟化技术，即 Intel 平台下的 EPT(Extended Page Table)[15] 或 AMD 平台下的 NPT(Nested Page Tables)[16]，在 QEMU-KVM 中对虚拟机内存页面读、写操作进行监测。EPT 或 NPT 在硬件层次对内存虚拟化进行支持：在 CPU 处于非根模式下增加一个

EPT 或 NPT 页表，将虚拟机的物理地址直接映射到宿主机的物理地址。虚拟机进行访存操作时，首先根据自己的页表完成进程逻辑地址到虚拟机物理地址的转换，然后由 EPT 或 NPT 页表完成虚拟机物理地址到宿主机物理地址的转换。这些地址翻译都是通过硬件完成的，避免了传统由软件实现内存虚拟化所导致的虚拟机频繁陷出及内存开销问题。由于 CPU 的每次访存操作都会修改页表项的 A(Accessed)位和 D(Dirty)位，因此根据 EPT 或 NPT 页表项的 A 位和 D 位就可判断内存页面的读、写操作，进而统计出读、写次数。在 P 周期内每隔 $S=50$ms 对 EPT 或 NPT 页表中对应虚拟机的所有页表项进行一次扫描。对于某个页表项，若 A 位为 1 且 D 位为 0，则判定该页面在上次扫描之后被读过，读次数加 1；若 A 位为 1 且 D 位为 1，则判定该页面在上次扫描之后被写过，写次数加 1。扫描之后将 A、D 位都重置为 0，并且调用 Linux 内核函数 *flush_tlb_page* 刷新对应的 TLB 表项，使得 CPU 下次访问该页面时可以进行正确设置。这里由于采用了硬件辅助的内存虚拟化技术，刷新 TLB 表项不会导致虚拟机陷出，保证了虚拟机的运行效率。P 周期结束后，将读、写次数返回给用户态模块进行处理。

据此，基于自适应的迁移算法选择框架的多虚拟机实时迁移运行流程如下(序号与图 5 对应)：

1. 启动一台物理机上的多虚拟机的实时迁移。

2. 虚拟机聚类模块对各个 QEMU 进程下发“获取页面读、写速率”指令。

3. 各 QEMU 进程的读、写速率跟踪模块收到指令后，对虚拟机内存页面读、写速率进行监测、统计和记录，完成后将记录结果返回给虚拟机聚类模块。

4. 虚拟机聚类模块根据各虚拟机的读、写速率记录对多虚拟机进行分类，完成后将分类结果传给迁移算法选择模块。

5. 迁移算法选择模块收到分类结果后为各虚拟机选择最佳迁移算法，并根据迁移算法同时启动各个 QEMU 进程的迁移命令。

6. 各 QEMU 进程开始正式执行虚拟机迁移。

4 仿真实验

本文分别基于单一迁移算法或自适应选择迁移算法的方式进行多虚拟机实时迁移实验，对比两者的实时迁移性能，从而对所提出的方案进行验证评估。

4.1 实验配置

实验使用 3 台配置相同的物理主机，其 CPU 为 Intel Core i7-4770 3.4GHz，内存 8GB，硬盘 500GB，操作系统为 Ubuntu Server 12.04，通过以太网交换机 1000Mbit/s 互联。两台主机安装 QEMU-KVM 2.1.5 作为虚拟化服务器，另一台安装 NFS 作为共享存储服务器。一台虚拟化服务器启动 4 台虚拟机同时运行，各虚拟机都配置为 1 个 1GHz 的 vCPU、512MB 虚拟机内存、10GB 虚拟磁盘，操作系统与宿主机相同；另一台虚拟化服务器空载运行，作为迁移的目的端。

虚拟机的运行负载通过 Linux 下的系统测试套件 Lmbench 中的内存测试程序 bw_mem 进行配置。各虚拟机负载配置情况如下。

1)虚拟机 VM1：采用带“rd”参数的 bw_mem 命令，不断对一个 8MB 的内存区域进行读操作；

2)虚拟机 VM2：采用带“wr”参数的 bw_mem 命令，不断对一个 8MB 的内存区域进行写操作；

3)虚拟机 VM3：通过一个 Shell 脚本不断交替执行 bw_mem 的 4 次“rd”和 4 次“wr”，循环对一个 2MB 的内存区域进行先读 4 次后写 4 次的操作；

4)虚拟机 VM4：通过一个 Shell 脚本循环执行 3 个 bw_mem 的“wr”和 1 个 bw_mem 的“rd”，不断对一个 2MB 的内存区域进行写 3 次后读 1 次的操作。

实验中，待各虚拟机稳定运行后，先后采用所有虚拟机都使用 Pre-copy、都使用 Post-copy 及自适应选择迁移算法(以下简称 Adaptive-copy)的迁移算法方案，将 4 个虚拟机以并行的方式(平均分配迁移带宽)实时迁移到目的端，以对比不同迁移算法选择方案下的实时迁移性能。以下实验结果均为 10 次实验数据的算术平均值。

4.2 实验结果及分析

在 Adaptive-copy 实时迁移的实验中对虚拟机聚类及迁移算法选择情况进行了跟踪，结果如表 1 所列。可以看到，虚拟机分类结果较为理想，其中 VM3 因为是读写复合密度型，其迁移算法还需要在迁移算法选择模块中根据每次实验的具体情况实时判定，因此实验结果为 10 次迁移中有 6 次为 Pre-copy，4 次为 Post-copy。

表 1 实验中虚拟机聚类及迁移算法选择结果

虚拟机\类型及算法	VM1	VM2	VM3	VM4
所属类型	读密度型	写密度型	读写复合密度型	写密度型
迁移算法	Pre-copy	Post-copy	Pre-copy (60%)	Post-copy

图 6 显示了 3 个迁移算法方案下各虚拟机的总迁移时间。可以看到，在单一采用 Pre-copy 的方案中，读密度型的 VM1 的性能最优。由于不存在脏页重传问题，VM1 的迁移时间基本等于拷贝整个内存镜像的时间。而对于其他 3 个带有内存写操作的虚拟机，由于要经过多次迭代拷贝进行脏页同步，其迁移时间都较 VM1 长，而且相比另 2 个方案中相同虚拟机的实验结果，性能也是最低的。特别是写密度型的 VM2，其采用 Pre-copy 后迁移性能非常低；在单一采用 Post-copy 的方案中，读密度型的 VM1 由于产生大量网络缺页中断，需要多次通过网络 Pull 页面，其迁移时间比其他 3 个带有在本地直接分配页面进行写操作的虚拟机都长，且性能更远不及在 Pre-copy 下的结果。该方案中，写密度型的 VM2 的迁移时间最短；在 Adaptive-copy 方案下，由于对 VM1、VM2、VM4 都直接选择了最适合的迁移算法，其迁移性能都对应于上面 2 个方案中最好的结果。对于 VM3，由于在多次实验中既有 Pre-copy 又有 Post-copy，图 6 中的迁移时间是这些迁移时间的平均值。可以看到，对于读写复合密度型的 VM3，由于 Adaptive-copy 根据迁移时段的读写侧重情况来选择迁移算法，因此其迁移性能比其它 2 个方案都好。

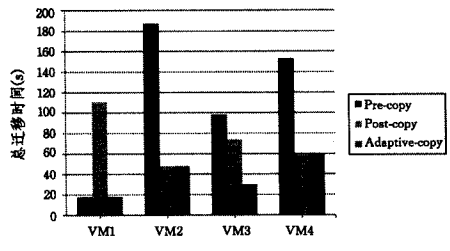

图 6　3 个迁移方案下各虚拟机的总迁移时间对比

图 7 中对比了 3 个迁移算法方案下各虚拟机的停机时间。首先可以看到 Post-copy 方案的结果突出了 Post-copy 在停机时间方面的优势，由于 Post-copy 在停机阶段主要进行设备状态的少量数据传输，因此各虚拟机的停机时间相等，且构成了所有方案的最小值。其次，在 Pre-copy 方案中，除 VM1 外的各虚拟机的停机时间都是 3 个方案中最长的，其中 VM2 最明显。原因是对于有写操作的虚拟机，Pre-copy 在停机阶段要传输剩余脏页，脏页率越高则剩余脏页越多，因而停机时间越长；而对于读密度型的 VM1，由于停机切换阶段基本没有脏页需要传输，其停机时间基本等于采用 Post-copy 的结果。最后，Adaptive-copy 方案下，同样因为除 VM3 外的 3 个虚拟机都直接采用了最优迁移算法，其停机时间基本都是所有方案中的最小值。对于 VM3，由于其停机时间是多次实验中 Pre-copy 和 Post-copy 结果的平均值，该值比 Post-copy 方案中的最小值要高，却远低于 Pre-copy 方案的数据。

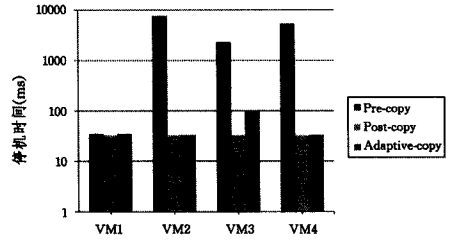

图 7　3 个迁移方案下各虚拟机的停机时间对比

根据以上各虚拟机的实时迁移性能结果，最后对多虚拟机实时迁移的整体性能进行评估。由于实验以并行方式迁移多虚拟机，以各虚拟机中最长的总迁移时间作为整体迁移时间，以各虚拟机停机时间之和为整体停机时间。另外，引入另一个性能指标“整体时间积”(见式(7))，以综合以上两个时间对整体迁移性能进行评估。

整体时间积＝整体迁移时间×整体停机时间　　(7)

表 2 对各迁移算法方案的 3 个整体性能指标进行了比较。可以看到，由于在实验所迁移的 4 个虚拟机中具有写操作的个数较多，Pre-copy 方案在 3 者中性能最差，特别在整体停机时间及整体时间积上与其他方案的差距十分明显，说明单一采用 Pre-copy 进行多虚拟机实时迁移时，若写密度型虚拟机数量较多，则整体迁移性能非常差。Post-copy 方案中由于受 VM1 迁移时间拖累，整体迁移时间较长，但因所有虚拟机都采用 Post-copy 算法，其整体停机时间是 3 个方案中最优的。Adaptive-copy 方案在整体迁移时间上优势明显，较前两方案分别减少了 68%和 47%。而由于 VM3 停机时间是采用 Pre-copy 和 Post-copy 算法的平均值，因此 Adaptive-copy 在整体停机时间上无法达到 Post-copy 方案的最小值。可以看到，Adaptive-copy 在整体迁移性能优化上权衡了整体迁移时间和整体停机时间，虽然无法让两者都达到最优，但是在整体时间积上是最优的，低于 Post-copy 方案 20%，说明 Adaptive-copy 在综合性能上是最好的。

表 2　3 个迁移算法方案的整体性能对比

迁移算法方案\性能指标	Pre-copy	Post-copy	Adaptive-copy
整体迁移时间(s)	187	111	58
整体停机时间(ms)	16175	132	200
整体时间积	3024725	14652	11600

5　相关工作

近年来，国内外对虚拟机实时迁移的研究工作主要侧重于迁移算法优化及多虚拟机迁移整体性能优化。

在迁移算法优化方面，文献[17]提出一个基于自适应内存页面压缩的实时迁移，根据内存数据的不同特点将内存页面分类，对不同类别的页面采用不同的压缩方法，以降低页面传输中的数据传输量；文献[5]在 Pre-copy 中利用重复数据删除技术降低数据传输率，基于计算内存页面的 hash 值找出虚拟机内存镜像中相同或相似的页面，在迁移中将冗余的页面数据只传输一次，提高了迁移效率；文献[18]基于页面预取(Pre-paging)和动态气球(Dynamic self-ballooning)技术对 Post-copy 进行优化。其采用 Pre-paging，在发生网络缺页异常时，根据缺页发生的空间局部性原理，在 pull 回所缺页面的同时还主动 push 回其邻居页面，以减少发生网络缺页异常的次数。另外，采用动态气球技术定期将虚拟机的空闲页面回收给虚拟机管理器，避免在迁移中传输空闲页面，以提高迁移速度；文献[6]从迁移算法的角度综合了 Pre-copy 和 Post-copy 特点，提出一种基于内存混合复制的动态迁移方法 HybMEC，分 3 个阶段进行页面拷贝：首先 push 全内存镜像，期间将产生的脏页映射到一个脏页位图中，然后进行类似 Post-copy 的停机切换，期间除了同步设备状态外还传输脏页位图，最后根据脏页位图 pull 回未同步的脏页。

在多虚拟机整体迁移性能优化方面，文献[8]基于重复数据删除技术优化多虚拟机并行实时迁移的性能。由于一个物理机上待迁移的多虚拟机在操作系统、运行库和应用程序上有很大相似性，通过为虚拟机内存页面或子页面建立哈希表，使得虚拟机之间的相同页面或页面中相似的部分只传输一次，减轻了多虚拟机同时迁移的带宽占用程度。文献[9]针对相互协作构建多层应用服务的多虚拟机的并行实时迁移问题，提出一个迁移调度系统 VMbuddies。其中包括一个“同步协议”用于保证具有协作关系的多虚拟机能同时迁移到目的端，避免因多层应用服务的组件分离造成整个应用系统性能下降；另外，还包括一个基于分布优化问题建模的带宽分配方案，用于给各迁移进程分配最优带宽，保证整体迁移代价最小。文献[10]考虑在 WAN 的高延时、低带宽条件下协作的多虚拟机迁移时的组件分离问题，提出一个基于关系分组的多虚拟机迁移方案 Clique Migration，其根据虚拟机间的通信量，分别采用 Min-Cut 和 Kmeans 原理提出两种对待迁移的虚拟机进行分组的算法，实现组间串行迁移、组内并行迁移，进而降低协作虚拟机因跨 WAN 迁移后分离所带来的性能开销。

与以上的研究工作相比，本文侧重于第二个方面，而且是从一个新的角度即自适应进行迁移算法选择来提高实时迁移整体性能。

结束语 IaaS云计算平台主要利用多虚拟机实时迁移对资源进行优化、调度与管理。目前的多虚拟机实时迁移采用固定、单一的迁移算法对各虚拟机进行迁移，没有考虑迁移算法对运行不同负载虚拟机的适用性和有效性，进而影响整体迁移性能。本文提出一种根据各虚拟机负载情况自适应选择迁移算法的有效框架，基于模糊聚类方法将虚拟机进行分类，为不同类型的虚拟机选择最适合的迁移算法。实验表明，所提出的方案能够发挥两个基本迁移算法的优点，有效提高多虚拟机实时迁移整体性能。下一步将考虑在实际的云平台（如Openstack）上部署该框架，在生产环境中进一步对该方案进行评估。

参 考 文 献

[1] Nelson M, Lim B, Hutchines G. Fast Transparent Migration for Virtual Machines[C]// Proceeding of the USENIX Annual Technical Conference. 2005:391-394

[2] Yuan Ye, Zhao Hai-yan, Cao Jian, et al. Research on Memory Migration of Virtual Machines[J]. Journal of Chinese Computer Systems, 2014, 35(2):412-418(in Chinese)
袁野，赵海燕，曹健，等. 虚拟机内存迁移技术研究[J]. 小型微型计算机系统，2014，35(2)：412-418

[3] Clark C, Fraser K, Hand S, et al. Live Migration of Virtual Machines[C]// Proceeding of The 2nd Symposium on Networked Systems Design and Implementation. 2005:273-286

[4] Hines M R, Deshpande U, Gopalan K. Post-copy Live Migration of Virtual Machines[J]. ACM SIGOPS Operating Systems Review, 2009, 43(3):14-26

[5] Zhang X, Huo Z, Ma J, et al. Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration[C]// Proceeding of the 2010 IEEE International Conference on Cluster Computing. 2010:88-96

[6] Chen Yang , Huai Jin-peng, Hu Chun-ming. Live Migration of Virtual Machines Based on Hybird Memory Copy Approach[J]. Chinese Journal of Computers, 2011, 34(12):2278-2291(in Chinese)
陈阳，怀进鹏，胡春明. 基于内存混合复制方式的虚拟机在线迁移机制[J]. 计算机学报，2011，34(12)：2278-2291

[7] Ye K, Jiang X, Ma R, et al. VC-Migration: Live Migration of Virtual Clusters in The Cloud[C]// Proceeding of the 13th ACM/IEEE International Conference on Grid Computing. 2012: 209-218

[8] Deshpande U, Wang X, Gopalan K. Live Gang Migration of Virtual Machines[C]// Proceeding of the 20th ACM International Symposium on High Performance Distributed Computing. 2011: 135-146

[9] Liu H, He B. VMbuddies: Coordinating Live Migration of Multi-tier Applications in Cloud Environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(4):1192-1205

[10] Lu T, Stuart M, Tang K, et al. Clique Migration: Affinity Grouping of Virtual Machines for Inter-cloud Live Migration[C]// Proceeding of the 9th IEEE International Conference on Networking, Architecture, and Storage(NAS). 2014:216-225

[11] Bezdek J C. Pattern Recognition with Fuzzy Objective Function Algorithms[M]. Plenum Press, New York, 1981

[12] Zhang Hui-zhe, Wang Jian. Improved Fuzzy C Means Clustering Algorithm Based on Selecting Initial Clustering Centers[J]. Computer Science, 2009, 36(6):206-209(in Chinese)
张慧哲，王坚. 基于初始聚类中心选取的改进FCM聚类算法[J]. 计算机科学，2009，36(6)：206-209

[13] 张忠平. 指数平滑法[M]. 北京：中国统计出版社，1996

[14] KVM. Kernel Based Virtual Machine[EB/OL]. (2015-07-20). http://www.linux-kvm.org/page/Main_Page

[15] Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture[M]. 2011: 35-104

[16] AMD Corporation. AMD64 Architecture Programmer's Manual Volume 2[M]// System Programming. 2006:1-124

[17] Jin H, Deng L, Wu S, et al. MECOM: Live Migration of Virtual Machines by Adaptively Compressing Memory Pages[J]. Future Generation Computer Systems, 2014, 38:23-35

[18] Hines M R, Gopalan K. Post-copy Based Live Virtual Machine Migration Using Adaptive Pre-paging and Dynamic Self-ballooning[C]// Proceeding of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. ACM, 2009:51-60

(上接第35页)

[5] Wang Fei, Si Guang-ya, Rong Ming, et al. Research on network-of-networks model of armament system-of-systems[J]. Systems Engineering and Electronics, 2015, 37(9): 2052-2060(in Chinese)
王飞，司光亚，荣明，等. 武器装备体系的异质超网络模型[J]. 系统工程与电子技术，2015，37(9)：2052-2060

[6] Zhang Jie-yong, Lan Yu-shi, Yi Kan, et al. Network－centric C4ISR system structure and super-Network model[J]. Command Information System and Technology, 2014, 5(5):1-12(in Chinese)
张杰勇，蓝羽石，易侃，等. 网络中心化C4ISR系统结构与超网络模型[J]. 指挥信息系统与技术，2014，5(5)：1-12

[7] Lempereur B, Merabti M, Qi Shi. Information Flow Monitoring: Model, Policy, and Analysis[C]// Developments in E-systems Engineering. 2011:227-232

[8] Wang Zhong-jie, Xu Xiao-fei. Service value dependency model based on layered hyper-graph[J]. Computer Integrated Manufacturing Systems, 2011, 17(8):1834-1843(in Chinese)
王忠杰，徐晓飞. 基于分层超图的服务价值依赖模型[J]. 计算机集成制造系统，2011，17(8)：1834-1843

[9] Zou Zhi-gang, Liu Fu-xian, Sun Shi-man, et al. Extended granular computing－based supernetwork model for auti－air operational architecture[J]. Complex Systems and Complexity Science, 2014, 11(2):24-35(in Chinese)
邹志刚，刘付显，孙施曼，等. 基于扩展粒度计算的防空体系结构超网络模型[J]. 复杂系统与复杂性科学，2014，11(2)：24-35