一种高效的虚拟机动态内存迁移方法

程虹锡[1] 谭 良[2]
(四川师范大学计算机科学学院 成都 610068)[1] (中国科学院计算技术研究所 北京 100190)[2]

摘 要 内存预拷贝是虚拟机动态迁移的主流策略,但是传统内存预拷贝算法是以脏页量为工作集,这对数据修改量小但在内存中分布较广的虚拟机环境来说,总体呈现迭代次数普遍较高、迁移总时间较长和迁移数据量较大的缺点。针对这一问题,提出了基于页内脏数据的预拷贝算法,该算法用以页为单位的脏数据作为工作集,并引入了新的数据结构记录页内脏数据。相比于以脏页为工作集粒度的传统内存预拷贝算法,该算法的工作集不仅能更准确地反映虚拟机的实际迁移环境,而且粒度明显细化,使得工作集小于预先设定的阈值的几率较大,从而可以减少迁移数据冗余,降低迭代次数,缩短迁移时间,降低迁移带宽。实验表明,该算法具有较高的虚拟机动态迁移效率。

关键词 内存迁移,页内脏偏移段,虚拟机动态迁移
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.4.022

Efficient Method of Live Migration on Virtual Machine Memory

CHENG Hong-xi[1] TAN Liang[2]
(College of Computer Science, Sichuan Normal University, Chengdu 610068, China)[1]
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Pre-copy is the main strategy of the virtual machine live migration, but the traditional pre-copy algorithm regards dirty page as unit work set, which makes modification data small. But for a virtual machine environment which is widely distributed in memory, the overall number of iterations is generally too high, and data migration time is too long. To solve the problems, this paper proposed pre-copy algorithm based on the dirty data in page as unit work set. The algorithm uses a new data structure to record dirty data in page. Compared to the dirty pages as unit work set, the algorithm's unit work set not only more accurately reflects the actual migration environment of a virtual machine, but also has significantly refined particle size, which makes the total work sets smaller than the preset threshold more probably, which can reduce the number of iterations, to shorten the migration time and reduce migration bandwidth. Experiments show that the algorithm has higher efficiency of virtual machine live migration.

Keywords Memory migration, Dirty offset segment within page, Live migration of virtual machines

1 引言

随着云计算的快速发展,虚拟化技术作为高效利用计算机资源的角色,已经成为云结构中必不可少的要素[1-3]。而随着众多商业巨头(Amazon,阿里云等)以面向用户的云服务作为盈利模式的普及,用户的体验上升到了第一位,针对负载均衡和热点等问题[4],要依赖虚拟机的动态迁移技术来保证在解决问题的同时不会影响到用户体验。

虚拟机动态迁移是指在源虚拟机达到迁移条件后触发迁移流程,通过一系列步骤,源虚拟机将自己所有资源如磁盘数据、CPU状态信息、内存等通过网络迁移到目的虚拟机,迁移过程中要保证用户体验,即要求源虚拟机上服务的中断时间足够短,用户感受不到迁移的发生。其中内存的信息量由于较大且频繁变化,在迁移流程中处于较复杂的环节。

目前,虚拟机动态迁移被广泛认可和运用的方法是预拷贝方法,但是该方法在高脏页率的环境下,表现出迭代次数太多、占有网络带宽太多和总迁移时间太长等缺点。本文通过对预拷贝算法进行改进,将迁移粒度缩小到基于页内脏数据,相比于以脏页为工作集粒度的传统预拷贝算法,该算法的工作集不仅更能准确反映虚拟机的实际迁移环境,而且粒度明显细化,使得工作集小于预先设定的阈值的几率较大,从而可以减少迭代次数,缩短迁移时间,降低迁移带宽。实验表明,该算法具有较高的虚拟机动态迁移效率。

2 相关工作

为了实现虚拟机的动态迁移,一个完整的内存迁移的过程分为以下3个阶段[5]:Push阶段、Stop-and-Copy阶段、Pull阶段。实际上,目前大部分的迁移策略只包含其中的1个或2个阶段。典型的策略有预拷贝和后拷贝两种方法[6,7]。本文主要针对预拷贝方案进行改进。

到稿日期:2015-03-22 返修日期:2015-06-08 本文受国家自然科学基金(61373162),四川省可视化计算与虚拟现实(KJ201402)资助。
程虹锡(1990-),男,硕士生,主要研究领域为虚拟化技术,E-mail:15196633255@163.com;谭 良(1972-),男,博士,教授,CCF高级会员,主要研究领域为可信计算、云安全、大数据处理,E-mail:tanliang2008cn@126.com。

预拷贝方法是由 Clark 等人[6]首先提出并实现的虚拟机动态迁移机制，目前 Xen、KVM 及 VMare 等主流的虚拟化平台上均提供了对该机制的实现。经典 Pre-copy 算法的核心思想是将内存复制分为 3 个阶段：首先将虚拟机的全部内存页面从源主机复制到目的主机，期间源虚拟机不间断运行；然后迭代复制在前一轮复制传送过程中被修改并且本轮到当前为止未被修改过的虚拟机内存页面，每一轮复制传送结束后都需要验证当前情况是否符合进行 stop-and-copy 步骤的条件，满足条件则开始 stop-and-copy，否则继续进行迭代复制传送；在 stop-and-copy 阶段中，源主机暂停虚拟机的运行并将最后剩余的虚拟机内存脏页面复制到目的主机。Pre-copy 迭代复制的终止条件是在经过多轮复制之后内存脏页面总量收敛到规定的最少剩余脏页数。算法中将修改频繁的页称作可写工作集(Writable Working Set，WWS)，简称工作集。这种迁移方式很好地平衡了宕机时间和总迁移时间之间的矛盾，但是对于内存修改比较频繁的虚拟机，即虚拟机上运行的应用是内存密集型应用，需要重复拷贝修改过的所有页面，不仅总迁移时间会延长，而且会占用大量网络带宽，系统的性能也随之降低。

针对传统预拷贝方法存在的问题，在基于预拷贝方法的基础上，有许多对性能进行优化的算法被提出，这些方法基本都是围绕迭代迁移内存这个环节来进行优化，主要有两类。一类是基于概率和统计相关知识来对迭代过程中的每轮工作集利用重传预测或者按网络链路质量合理分发等方法[12,14,15]，加快每轮迭代过程，提高迁移效率。例如文献[12]利用马尔科夫链模型来分析本轮脏页被重传的概率，每轮只传输被重传概率小于预期概率的脏页，减少每轮迭代的传输量来加快迭代。另一类是对工作集通过压缩等方法进行二次处理[10,11,13]，目的也是通过减少每轮迁移的传输量来提高迁移效率。这些算法对缩短总迁移时间和减少网络带宽有一定的作用，但是这些算法不是从底层根本解决问题，不能准确地反映实际脏数据的大小，当脏页数较多，实际的脏数据较少时，这些算法不能敏锐地抓住跳出迭代的机会，表现出较低的效率。

关于虚拟机动态迁移，还有一些其它算法被提出[7,17]。与这些算法相比，预拷贝算法在迁移过程中极少时间改变用户访问资源和访问环境，且在内存脏页数足够少时才停机拷贝，使得停机时间极短，相比于其他算法，其用户体验方面表现得更好，实际应用更广。

总之，尽管当前对预拷贝方法进行了大量的优化，但这些算法对于数据修改量小但在内存中分布较广的虚拟机环境来说均不适合，因为这类虚拟机尽管数据变化小，但在内存中分布广，从而导致脏页量较大。本文将针对这一情况进行优化改进。

3 一种高效的虚拟机动态内存迁移方法

本节基于 Xen 提出将脏页内被修改数据的实际偏移地址范围作为工作集的迁移算法，从而能获得粒度更精确的脏数据来与阈值进行比较，最终达到高的迁移效率。

3.1 传统迁移机制流程及改进

Xen 传统迁移机制流程可以分为 5 个阶段：迁移准备阶段、锁定目标资源阶段、预拷贝阶段、停机拷贝阶段和结束迁移阶段，如图 1 所示。

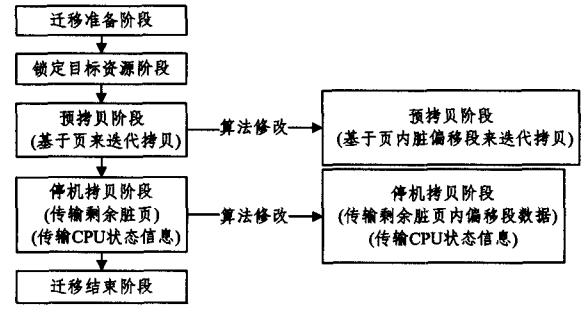

图 1 Xen 传统迁移机制流程及改进环节

各阶段具体功能如下：

(1)迁移准备阶段：源域在网络环境中搜索拥有足够资源实现迁移的目标域，当确定目标域后，迁移流程进入锁定目标资源阶段。

(2)锁定目标资源阶段：源域对目的资源进行申请并锁定，防止迁移过程中目标域上的资源丢失造成迁移失败，目标资源被锁定后发出迁移信号进入预拷贝阶段。

(3)预拷贝阶段：在迁移信号被发起后，源域的内存信息以页为粒度开始被收集和传输，通过迭代的方法传输两轮之间产生的新的脏页，第一轮将所有内存页传输到目的域，从第二轮开始通过位图确定本轮被更新过的脏页，并将其作为本轮的工作集传输到目标域。当满足某一轮传输的脏页数少于阈值或者达到最大迭代次数等临界条件时，源域被宕机，对外服务暂时被中断，预拷贝阶段结束，进入停机拷贝阶段。

(4)停机拷贝阶段：源域将剩下的脏页和当前的 CPU 状态信息传送到目的域，然后进入最后的结束迁移阶段。

(5)结束迁移阶段：网络被重定向到目标域，目标域被唤醒，源域资源被清除，最终迁移流程结束。

改进后的算法将对传统迁移机制流程中预拷贝阶段的工作集测定方法、预拷贝阶段和停机拷贝阶段传输脏页的数据结构进行改进，引入了记录页内脏数据的偏移地址范围来作为工作集，在每次迭代结束前以本轮记录的工作集来与原阈值进行比较，因为进程执行有访问局部性特征且新工作集的粒度足够的小，使得工作集小于阈值的机会较大，改进后的算法能更敏感地抓住迭代结束的机会；并且由于每轮网络传输的工作集相对于脏页粒度的工作集小许多，改进后的算法使每轮迭代的时间迅速减少，即迭代的敏捷度越来越高，从而每轮获取的工作集的大小迅速减小，能够更快地达到小于阈值条件。基于上面两个因素，改进后的算法能更快地跳出迭代，使得迁移总时间明显缩短，表现出高效的迁移性能。

3.2 改进迁移机制的算法

本小节介绍对迁移机制的改进方法，重点介绍改进方法中涉及的数据结构和算法流程。

3.2.1 数据结构

为了获取工作集，需要在源域发起迁移信号后开始对每轮被修改的页内偏移地址范围做记录。定义记录每个脏页内被修改偏移地址范围的结构体(如图 2 所示)为：页号 page_num，偏移起始地址 start_offset，偏移结束地址 end_offset，页内脏数据大小 data_size 和指向下一个结构体的指针 page_next。为了能够尽量减少分配内存操作，以提高收集工作集的效率，本文的算法采用 3 个链表来保存数据结构体：链表 next 收集下一轮的工作集，链表 current 保存本轮待处理的工作集，链表 free 保存以前分配的空闲结构体。记录本轮修改

内存操作的数据结构体都会插入到 next 链表，next 链表中需要添加新的结构体时，会先到 free 链表中找取，如果 free 中没有空闲结构体，才创建新的结构体到 next 中。当开始本轮迭代时，current 链表指向 next 获取本轮工作集，然后 next 指向空，来收集下一轮的工作集，current 中的结构体在被读取后会添加到 free 的表尾作为空闲结构体被下一轮迭代使用。

```
struct pageInfo {
int page_num;
int start_offset;
int end_offset;
int data_size;
struct pageInfo * next
    };
```

图 2　记录单个页内被修改偏移段的数据结构

3.2.2　算法设计

由于进程的运行有局部性访问特征，且为了不让数据结构占用太多的内存，保证每个页面只对应一个数据结构来做记录，本文在记录脏页内被修改数据的偏移地址范围时每轮只更新页内起始偏移地址和终止偏移地址。预拷贝阶段算法的流程如图 3 所示。

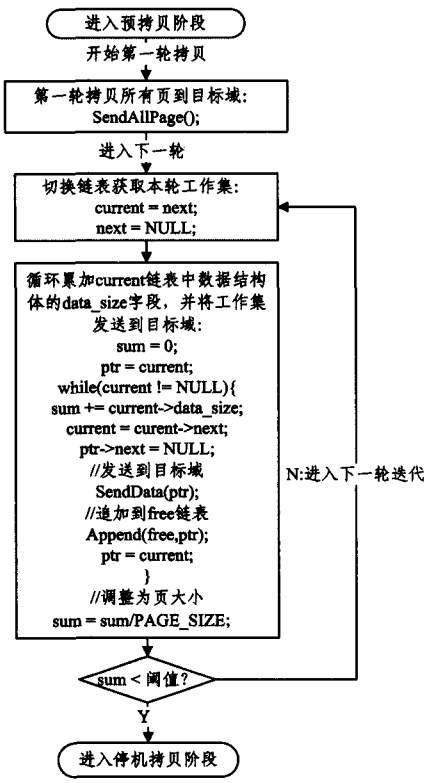

图 3　算法中工作集测定流程

具体的算法步骤如下：

步骤 1　在预拷贝阶段开始后，第一轮传输所有内存并开始收集下一轮的工作集(时间长，脏页积累多)，在内存页被修改前，遍历 next 确定该页数据结构是否已经存在。若没有找到匹配的 page_num，就用一个新的结构体记录并按 page_num 排序插入到 next 链表相应位置来提高下次匹配效率；否则，说明 next 链表中已经有该页结构体，只需要直接更新该页的数据结构体。若被修改的地址范围的起始偏移地址小于 start_offset，则用该值更新 start_offset；若被修改的地址范围的终止偏移地址大于 end_offset，用同样的方式更新 end_offset 字段，然后更新 data_size 等于 end_offset 减去 start_offset。

步骤 2　current 指向 next 链表，next 链表指向空。

步骤 3　用变量 sum 来累加 current 链表中每个数据结构的 data_size 字段，从而获取以字节为单位的脏数据大小，并且每读取一个数据结构体，将其和对应页内的脏数据一起发送到目的域，然后将其从 current 中删除，添加到 free 链表中，保证 next 链表能及时获取空闲结构体而不用重新分配影响效率。

步骤 4　调整 sum 为以页为单位的大小，然后与阈值比较，若小于阈值则进入停机拷贝阶段，否则转入步骤 2，进入下一轮迭代。

4　实验与分析

在局域网中通过两台 512MB 内存的 PC 节点来进行实验，虚拟平台为 Xen4.2，链路速度为 100Mb/s，虚拟机的操作系统为 Debian7.0，通过随机数访问内存来模拟高脏页率环境，当脏页数小于等于 50 时跳出迭代。具体实验结果如表 1 和表 2 所列。

表 1　每 100ms 修改一次内存的频率下随机两次实验的迭代过程

第 N 次实验	第 N 次迭代	实际脏页数	基于页内偏移段的脏页数	总迁移时间(s)	迭代次数
1	1	65692	16397	58.368	11
	2	24961	6153		
	3	10128	2565		
	4	4209	1054		
	5	1762	435		
	6	738	192		
	7	249	60		
	8	246	68		
	9	237	59		
	10	240	52		
	11	154	38(<50)		
2	1	66152	16361	58.152	8
	2	25992	6536		
	3	10395	2627		
	4	4333	1088		
	5	1339	324		
	6	379	96		
	7	315	81		
	8	96	29(<50)		

表 2　不同频率下修改内存来比较算法性能

访问内存时间间隔(ms)	算法	总迁移时间(s)	迭代次数
100	传统预拷贝算法	MAXTIME	MAXTIME
	改进后的算法	58.215	10
200	传统预拷贝算法	MAXTIME	MAXTIME
	改进后的算法	55.185	8
300	传统预拷贝算法	111.332	27
	改进后的算法	53.409	4

表 1 是在每 100ms 修改一次内存的频率下随机选出的两次实验数据，第 N 次迭代是每轮迭代的序号，每次实验中传输所有内存页后的下一轮标记为第一轮，实际脏页数是基于页为单位的工作集数目，基于页内偏移段的脏页数是基于页内脏数据为工作集，并将其转为以页为单位，转换公式为：基于页内偏移段的脏页数＝所有页内脏数据的和/单位页大小。从表中可以看到，在每一轮中基于页内偏移段的脏页数都要比实际脏页数少很多，改进后的算法每轮传输的工作集

大小是基于页内偏移段的脏页数，相比于传统预拷贝算法，每轮网络传输的工作集大量减少，加快了每轮的迭代，缩短了下一轮工作集累积时间，使得接下来每轮的工作集大小迅速收敛，直到低于阈值，缩短了总迁移时间且减少了迭代次数。另外，从表中还可以看到，两次实验中，每次跳出迭代时的实际脏页数都大于阈值，而基于页内偏移段的脏页数小于阈值，改进后的算法是以基于页内偏移段的脏页数来与阈值比较，这也体现了改进后的算法对跳出迭代拥有较强的敏感度，能够更早地跳出迭代。

表2对传统预拷贝算法和改进后的算法进行了对比，分别在100ms，200ms，300ms修改一次内存的频率下比较传统预拷贝算法和改进后算法的总迁移时间和迭代次数。在每100ms和200ms间隔随机访问一次内存时，因为内存访问频率太高，传统预拷贝算法不能通过比较阈值跳出迭代，只能通过设置超时时间等条件停止迭代，所以总迁移时间和迭代次数标记为MAXTIME，而改进后基于页内脏数据为工作集的算法在这两个频率下依然保持着高效的迁移效率，很快跳出迭代，完成了迁移。在300ms间隔下，传统预拷贝算法和改进后的算法都能够跳出迭代，但是从迭代次数和总迁移时间方面来进行对比，改进后算法的效率远高于传统预拷贝算法。

结束语 本文通过对传统基于预拷贝动态迁移算法的研究和分析，修改了传统预拷贝算法的数据结构，将基于页为工作集的粒度缩小到基于页内偏移段为工作集，既增加了迭代的敏捷度，使得工作集快速收敛，也提高了跳出迭代的敏感度，使得算法能及时跳出迭代，最终让动态迁移过程中的预拷贝环节的迭代次数大幅度减少，从而大幅度缩短了总迁移时间，表现出高效的迁移效率。

参 考 文 献

[1] Zhang Xin-ling, Zhang Dong, Cao Ling-ling, et al. Research of virtualization platform in Coud Computing[J]. Software Guide, 2013,12(11)(in Chinese)
张新玲，张东，曹玲玲，等. 云计算虚拟化平台性能研究[J]. 软件导刊，2013,12(11)

[2] Yang Jing. Key Technologies and Optimization for dynamic migration of Virtual Machines in Cloud Computing[D]. Changsha: Central South University, 2011(in Chinese)
杨旌. 面向云计算的虚拟机动态迁移关键技术及优化[D]. 长沙：中南大学，2011

[3] Liu Peng-cheng, Chen Rong. Cloud Computing-oriented Live Migration Framework for Virtual Machine[J]. Computer Engineering, 2010,36(5):37-39(in Chinese)
刘鹏程，陈榕. 面向云计算的虚拟机动态迁移框架[J]. 计算机工程，2010,36(5):37-39

[4] Zhang X, Huo Z, Ma J, et al. Exploiting data deduplication to accelerate live virtual machine migration[C]// Proceedings of the 2010 IEEE International Conference on Cluster Computing. 2010:88-96

[5] Clark C, Fraser K, Hand S, et al. Live Migration of Virtual Machines[C]// Proceedings of the 2nd Symposium on Networked Systems Design and Implementation. Boston, Massachusetts, USA, 2005:273-286

[6] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines[C]// Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI'05). Berkeley, USA: ACM Press, 2005:273-286

[7] 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

[8] Sohan R R, Moore A, Hopper A W A. Predicting the performance of virtual machine migration[C]// Proceedings of the Modeling, Analysis & Simulation of Computer and Telecommunication Systems(MASCOTS). Florida, USA, 2010:37-46

[9] Zhang Wei, Zhu Ming-fa, Gong Tao, et al. Performance degradation-aware virtual machine live migration in virtualized servers [C]// Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. Las Vegas, USA, 2012:429-435

[10] Jin H, Deng L, Wu S, et al. Live virtual machine migration with adaptive memory compression[C]// Proceedings of the IEEE International Conference on Cluster Computing(Cluster'09). New Orleans, Louisiana, USA, 2009:1-10

[11] Svard P, Hudzia B, Tordsson J. Evaluation of delta compression techniques for efficient live migration of large virtual machines [C]// Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'11). New York, 2011:111-120

[12] Sun Guo-fei, Gu Jian-hua, Hu Jin-hua, et al. Improvement of live memory migration mechanism for virtual machine based on pre-copy[J]. Computer Engineering, 2011, 37(13): 36-39(in Chinese)
孙国飞，谷建华，胡金华，等. 基于预拷贝的虚拟机动态内存迁移机制改进[J]. 计算机工程，2011,37(13):36-39

[13] Hu Liang, Zhao Jia, Xu Gao-chao, et al. HMDC: Live virtual machine migration based on hybrid memory copy and delta compression[J]. Applied Mathematics & Information Sciences, 2013,7(2L):639-646

[14] Liu Shi-hai, Sun Yu-qing, Liu Gu-yue. An Adaptive Bandwidth Allocation Algorithm for Virtual Machine Migration Based on Service Features[J]. Chinese Journal of Computers, 2013,36(9) (in Chinese)
刘诗海，孙宇清，刘古月. 面向业务特征的自适应虚拟机迁移带宽分配算法[J]. 计算机学报，2013,36(9)

[15] Cheng Ting-wei, Zhang Pu, Zhang Zhong-qing. Memory Optimization Algorithm of Migration Based on Xen Virtual Machine [J]. Computer Science, 2013,40(9)(in Chinese)
陈廷伟，张璞，张忠清. 基于Xen的虚拟机迁移时内存优化算法[J]. 计算机科学，2013,40(9)

[16] Zhang Jing-kun. Research and Application of Virtual Machine Live Migration Techniques Based on Xen[D]. Shenyang: Northeastern University, 2011(in Chinese)
张井昆. 基于Xen的虚拟机动态迁移技术研究及应用[D]. 沈阳：东北大学，2011

[17] Chen Yang, Huai Jin-peng, Hu Chun-ming. Live Migration of Virtual Machines Based on Hybrid 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

[18] Liu Hai-kun, Jin Hai, Liao Xiao-fei, et al. Live migration of virtual machine based on full system trace and replay[C]// Proc of the 18th ACM International Symposium on High Performance Distributed Computing. New York: ACM Press, 2009:101-110