面向业务特征的自适应虚拟机迁移带宽分配算法

刘诗海[1),2),3)] 孙宇清[1)] 刘古月[4)]

[1)](山东大学计算机科学与技术学院 济南 250101)

[2)](中国人民解放军68067部队 兰州 733000)

[3)](中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

[4)](北京邮电大学国际学院 北京 102209)

摘 要 虚拟机动态迁移是支持绿色云计算环境的重要技术,迭代时间和宕机时间是迁移性能的衡量指标,而虚拟机迁移时使用的网络带宽和业务运行产生的内存脏页是影响迁移性能的重要因素,因此合理分配迁移带宽和减少脏页率能够有效缩短迭代时间和宕机时间.该文提出了一种面向业务特征的自适应虚拟机迁移带宽分配算法,通过对迁移过程中脏页率的分析,预测运行业务的网络带宽使用量,自适应分配虚拟机迁移带宽;引入带宽调整系数,有效处理迁移过程中的业务数据抖动现象,从而确保预测的合理性.这一算法能够在保证迁移性能和系统可靠性的同时,减少迭代时间和宕机时间.实验表明在带宽资源有限的前提下,该方法能够合理利用空闲带宽资源,提高迁移性能,确保业务服务质量.

关键词 虚拟机;动态迁移;带宽分配;脏页率;绿色计算;云计算

中图法分类号 TP393 **DOI号** 10.3724/SP.J.1016.2013.01816

An Adaptive Bandwidth Allocation Algorithm for Virtual Machine Migration Based on Service Features

LIU Shi-Hai[1),2),3)] SUN Yu-Qing[1)] LIU Gu-Yue[4)]

[1)](*School of Computer Science and Technology, Shandong University, Jinan* 250101)

[2)](*People's Liberation Army* 68067, *Lanzhou* 733000)

[3)](*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[4)](*International School, Beijing University of Posts and Telecommunications, Beijing* 102209)

Abstract Live migration of virtual machine (VM) is an important technology in green cloud computing environment, by which a virtual machine monitor moves an entire running VM from one physical machine to another. Iteration time and downtime are regarded as important indexes to evaluate migration performance, which highly depends on the assigned network bandwidth and the dirty-page rate of transferred VM memory page in a migration process. In this paper, an Adaptive Bandwidth Allocation Algorithm is proposed for live migration of virtual machine based on service features. By analyzing the dirty-page rate in each iteration of migration process, the algorithm predicts the network traffic of current tasks and allocates the migration bandwidth accordingly. By introducing the adjustment coefficient of bandwidth, our algorithm can handle the data jitter in practical service running so as to make the assignment of migration bandwidth more reasonable. We perform a set of experiments to evaluate our method on running services

收稿日期:2011-10-21;**最终修改稿收到日期**:2013-05-07.**本课题得到国家自然科学基金**(61173140)、**国家科技支撑计划项目**(2012BAF10B03-3)、**中国科学院计算机系统结构重点实验室开放课题**(ICT-ARCH200904)**资助**.**刘诗海**,**男**,1982**年生**,**硕士**,**主要研究方向为虚拟化技术和系统安全**.E-mail:liushihai1982@qq.com.**孙宇清**(**通信作者**),**女**,1967**年生**,**博士**,**教授**,**中国计算机学会**(CCF)**高级会员**,**主要研究领域为系统安全与隐私保护**.E-mail:sun_yuqing@sdu.edu.cn.**刘古月**,**女**,1990**年生**,**硕士**,**主要研究方向为虚拟化技术**.

with different features. The results show that it can well utilize the idle bandwidth and improve the performance of live migration on both downtime and the overall migration time. It also improves the system reliability and provides a better service quality for users.

Keywords virtual machine; live migration; bandwidth allocation; dirty-page rate; green computing; cloud computing

1 引　言

云计算环境能够为用户提供安全可靠的计算和存储服务，被称为下一代计算模式. 它利用虚拟化技术管理共享资源，通过创建多个虚拟机来装载不同服务，以隔离执行环境，满足不同用户的应用需求；并可以自适应分配和调度各种物理资源，从而高效绿色地管理云计算平台中的资源和服务[1]. 其中，虚拟机动态迁移(Live Migration)技术[2]是在不影响服务的前提下，将服务从源主机迁移到目的主机继续运行，以达到动态负载均衡、在线系统维护等目的，是实现绿色资源管理的重要手段[3]. 然而，虚拟机迁移过程有少量的宕机时间，会使服务产生短暂中断，部分地影响服务质量和用户体验；另一方面，虚拟机迁移还占用额外的 CPU、网络带宽、内存等物理资源，也一定程度影响到云计算环境中的其他服务. 因此，提高虚拟机迁移性能，是维护高效绿色资源管理的核心，也是虚拟化技术的研究热点.

虚拟机动态迁移的关键是从源主机高效复制和传输虚拟机内存状态到目的主机，现有的主流虚拟化平台如 Xen、VMware 等普遍采用预拷贝算法[2]进行虚拟机动态迁移，并且有许多成功的应用如亚马逊、IBM 等云计算服务平台[4-6]. 预拷贝算法分为初始化、迭代拷贝和宕机拷贝 3 个阶段，通过多轮迭代将虚拟机内存状态从源主机发送到目的主机，待内存同步后，目的主机继续运行虚拟机[7]. 这种方法可以缩短虚拟机迁移时间，提高迁移性能. 然而，在实际应用中，虚拟机运行不同业务所使用的物理资源如网络带宽和业务运行产生的内存脏页等因素会很大程度上影响迁移性能. 例如，当虚拟机中运行 I/O 密集型业务时，内存修改速率过大，在宕机拷贝时需要拷贝大量内存脏页，增加宕机时间，影响用户访问虚拟机内部的应用. 另一方面，当可用物理网络带宽小于虚拟机脏页的传输量时，预拷贝算法中迭代发送过程就会强制停止，从而造成内存同步收敛性问题[8].

针对现有虚拟化平台中动态迁移的预拷贝算法的不足，本文提出了一种面向业务特征的自适应虚拟机迁移带宽分配算法，通过分析迁移迭代过程中的脏页率，使用带宽调整系数，预测运行业务网络带宽使用量，从而自适应分配虚拟机迁移带宽，在保证迁移性能和系统可靠性的同时，减少迭代时间和迁移时间.

本文第 2 节介绍相关工作；第 3 节详细描述算法的整体思路；第 4 节介绍具体实现以及系统性能分析；最后总结全文.

2 相关工作

虚拟机现有的动态迁移技术可分为预拷贝和后拷贝两种方法[9-10]，基于后拷贝(Post-copy)算法的动态迁移技术是将内存的同步推迟到 VM 在目的主机上恢复运行之后. 迁移时先将被迁移虚拟机除内存以外的设备状态在两台主机之间同步，并在目的主机中启动虚拟机，然后使用按需取页的方式实现两端虚拟机内存同步. 虽然该机制能够保证每个内存页至多复制一次，从根本上避免了冗余数据对网络带宽的占用，但是当虚拟机在目的主机上运行，并访问未同步页面时，该机制会使虚拟机频繁中断以同步内存页，造成虚拟机性能下降以及内存应用执行延时增加. 同时，也不能够很好地满足动态迁移透明性要求.

预拷贝方法是现有虚拟化平台普遍采用的迁移方案，它通过迭代拷贝内存，以较短的迭代拷贝时间和宕机时间，实现“无缝”的虚拟机动态迁移，以下讨论将主要针对这种方案进行改进. 在实际应用中，由于虚拟机中运行业务特征的不同，会造成不同程度地延长迭代拷贝时间和宕机时间，影响整体迁移性能，甚至会造成动态迁移失败. 目前，针对上述问题简单的解决方法是在执行迁移前用户根据经验以及当前业务特征，手动调整迭代发送次数以及每轮发送的缓存，强行制约发送过程中对内存状态的复制. 但对于 I/O 密集型业务，在宕机拷贝时可能还存在大量需发送的内存脏页，延长虚拟机宕机结束时间，

影响动态迁移过程中外部程序的可访问性.为了减少迁移过程中发送的内存数量,现阶段使用最多的是气球驱动(Balloon Driver)机制[7],在迁移准备阶段减少被迁移虚拟机空闲和不使用的内存,但这种方法只能够缩短首轮迭代发送的时间.文献[11]设计并实现了一种虚拟机迁移的选择策略,用来选择哪个虚拟机应该进行迁移.但该方法仅从内存尺寸、脏页率等角度进行虚拟机的选择,未结合业务特征,动态改变带宽分配.

在使用预拷贝机制进行内存迭代发送时,如果出现迁移数据冗余性过高,就会造成物理资源的损耗.为了减少冗余性和内存发送数量,Jin 等人[12-13]提出了利用数据压缩算法将每轮发送的内存页面压缩后传输,由于压缩算法的限制,会出现压缩后传输的时间大于未压缩传输的时间,以及对宿主机 CPU 等物理资源的额外消耗.孙国飞等人[10]利用马尔可夫预测模型优化了预拷贝机制中工作集的选取方法,对传输页面进行了预判,减少了对冗余数据的传输,但该方法只对脏页率较高的情况有较好效果.文献[15]提出虚拟机迁移算法 HMDC,将 Pull 阶段产生的内存副本与 Push 阶段产生的内存副本进行组合,再利用异或二进制 RLE 算法,将内存页进行压缩,将压缩的内存页面发送给目的主机.通过差值压缩,HMDC 能够增加吞吐率并减少传递的数据量,从而能较快地复制内存脏页,但是该方法的鲁棒性需进一步研究,如停电或机器崩溃时,如何从源主机进行恢复.

远程内存访问(Remote Direct Memory Access,RDMA)技术是另一种基于预拷贝算法的虚拟机迁移策略[14],它根据主机资源需求和负载情况,选择不同的迁移协议,可以有效减少传输数据量,但是这种方法需要高性能网络支持,在实际应用中会受到局限.利用调度 CPU 状态等硬件控制方法限制内存访问次数的迁移策略可以减少迭代发送量[16],但会影响用户访问虚拟机内部的资源和服务质量.基于"记录-重放"技术的迁移从一个检查点状态出发,使用记录的日志向前回滚,达到提高迁移性能的目的[17].但在多处理环境中,记录和重放虚拟 CPU 之间的内存同步状态代价很高,增加了迁移难度,性能提高也不是很明显,同时当脏页率大于预拷贝速度时就会失效.文献[18]中在集群环境中利用分布式文件系统的可扩展性减少迁移时间,只适用于 GlusterFS 文件系统.结合内存推送复制以及按需复制两种方式实现虚拟机的快速迁移[8],虽然避免了不必要的状态切换,优化了迁移时间,但是虚拟机的完整状态需要由源和目的宿主机共同维护,否则会导致虚拟机迁移失败和运行状态丢失.针对虚拟集群迁移,文献[19]提出了一个动态的虚拟机迁移机制 Shrinker,通过检查同一个虚拟集群中的内存页面以及磁盘块,避免将相同内容的副本进行多次发送,从而减少迁移过程中的迁移数据量以及迁移时间,该方法同样没有结合业务特征,动态改变带宽分配.

作为虚拟化领域的主流平台,开源虚拟机管理系统 Xen 在使用预拷贝算法实施动态迁移的过程中,使用默认网络带宽传输分配和增量网络带宽分配两种内存传输方式[20].其中,默认带宽分配模式是将本轮迭代需要发送的脏页面放入内部定义的缓存中,当需要发送的内存页面被存放完毕或缓存被放满后,通过原生设备驱动或设备模型利用安全硬件接口将缓存中的内存页面全部传输到物理网卡,由物理网卡选择发送时机.在实际应用过程,受到实际网络带宽和运行业务特征等影响,当物理带宽使用量较大时,可能会导致大量缓存拥塞在物理网卡处排队,出现迭代过程的收敛性问题[7],并且还会出现发送内存与业务运行抢占带宽的现象,加之预拷贝算法受自身特征约束,影响迁移性能和运行业务服务质量,甚至造成迁移失败.增量带宽分配模式是将首轮发送带宽初始化为常量,后续每轮迭代发送带宽是在上一轮迭代时产生的脏页率基础上增加一个常数量,但最大使用带宽不能超过系统限定的常数级.该模式受到内部实现机制的限制,在百兆以下网络(包括百兆)环境是失效的,在千兆以上网络(包括千兆)环境中,物理带宽使用量较少时,会造成物理带宽资源利用不充分,限制虚拟机迁移性能提升,同时根据单轮脏页率分配发送带宽,对于处理业务突发性和数据抖动性是不合理的.

3 自适应虚拟机迁移带宽分配算法

3.1 整体思路

基于预拷贝机制的虚拟机动态迁移过程包括迁移初始化阶段、迭代拷贝阶段和宕机同步阶段 3 个方面.初始化阶段是对被迁移的虚拟机的基本情况进行收集、判断与确定,并根据迁移虚拟机的信息,在目的主机中申请并检查预定资源,确定迁移的基本条件.此阶段所需要的时间(表示为 T_{init})不受虚拟机内存大小以及网络带宽的影响,因此,不影响虚拟机动态迁移性能.

迭代拷贝是指利用迭代拷贝的方法将被迁移虚拟机内存从源主机拷贝到目的主机. 初始化阶段完成后，会进入迭代拷贝阶段. 迭代拷贝时间越短，业务应用与迁移竞争资源的时间越短，服务质量受到的影响越少. 若迭代拷贝轮数为 n，第 1 轮迁移迭代发送时需要将被迁移虚拟机的所有内存页面从源主机发送到目的主机；从第 2 轮到 $n-1$ 轮迭代，需要根据前一轮产生的内存脏页数量判断和计算本轮所需发送的虚拟机内存页面数，所以在虚拟机内存被修改的速率一定时，被发送的内存脏页数量跟上一轮迭代的时间相关，迭代时间越长产生的脏页量可能会越多，而且迭代发送带宽的大小直接影响到每轮迭代迁移的时间. 设被迁移虚拟机的内存大小为 M_{mem}，每轮发送的内存脏页数量为 M_iter_i，传输内存脏页使用的网络带宽为 B_iter_i，每轮迭代使用的时间为 T_iter_i，根据预拷贝算法可知，第 1 轮迭代需要发送被迁移虚拟机的全部内存，其它迭代轮次（除最后一轮迭代）只发送前一轮迭代时由于业务运行产生的内存脏页的数量. 为此，我们引入式(1)计算每轮迭代发送内存所需的时间.

$$T\_iter_i=\begin{cases}\dfrac{M_{\mathrm{mem}}}{B\_iter_1}, & i=1\\ \dfrac{M\_iter_i}{B\_iter_i}, & 1<i<n\end{cases} \tag{1}$$

宕机拷贝是指动态迁移过程中虚拟机暂停后将剩余的内存拷贝到目的主机，并在两台主机之间同步内存和相关设备状态的阶段. 宕机拷贝时间受被迁移虚拟机剩余内存和发送带宽限定，宕机时间越短，越能保证业务的服务质量；如果宕机时间过长，就会影响业务服务请求. 设剩余的内存脏页拷贝时间为 T_{down}，内存和设备状态同步时间为 T_{dev}. 为此，我们引入式(2)计算宕机拷贝所需的时间.

$$T_{\mathrm{down}}+T_{\mathrm{dev}}=\frac{M\_iter_n}{B\_iter_n}+T_{\mathrm{dev}} \tag{2}$$

根据上述分析可知，虚拟机动态迁移所需的整体时间（表示为 T_{total}）包括迁移初始化时间、迭代拷贝时间和宕机同步拷贝时间，如式(3)所示. 现有研究表明，虚拟机动态迁移时间主要取决于迭代拷贝和宕机拷贝所需时间，动态迁移过程中发送内存使用的物理网络带宽和虚拟机脏页率是影响上述性能指标的主要因素. 当需要传输的内存脏页确定时，网络带宽的大小直接影响到每轮迭代迁移的时间；在最后一轮（宕机拷贝阶段）网络带宽的合理分配也会影响虚拟机宕机时间. 综上分析，如果物理网络带宽的使用分配不合理，就会增加后续每轮迭代的时间，迭代时间的增加，势必会造成下一轮被发送的脏页数量增加，造成恶性循环，还可能使迭代过程中活跃页面的数量增加，最后一轮发送时间增加，造成虚拟机宕机时间增加，极端情况下，还会使每轮产生的脏页都为活跃页，全部积攒到最后一轮发送，此时预拷贝算法失效，使动态迁移变为静态迁移，最终不仅会影响虚拟机的迁移的性能，还会影响在虚拟机内部运行业务的服务质量.

$$T_{\mathrm{total}}=T_{\mathrm{init}}+\sum_{i=1}^{n-1}T\_iter_i+T_{\mathrm{down}}+T_{\mathrm{dev}} \tag{3}$$

为此，本文提出了自适应虚拟机迁移带宽分配算法，根据迭代轮数的不同，确定不同的带宽分配方式. 当第 1 轮迭代时，根据现阶段网络状态初始化迭代发送带宽；当最后一轮迭代时，将全部剩余带宽发送分配给迭代发送带宽；当其它轮迭代时，根据前一轮迭代产生的脏页率分析当前被迁移虚拟机中运行业务需要的带宽，并结合当前网络状态分配迭代发送带宽. 这样通过合理调整迭代拷贝过程中的带宽使用量，可以合理利用空闲带宽资源，减少迁移迭代的时间，提高迁移性能和保证业务服务质量. 具体流程如图 1 所示.

3.2　数据抖动处理

在实际应用中，不同服务占用物理资源不同，如流媒体服务主要是占用网络资源，而电子商务服务则占用网络资源相对较小；相同服务在不同时刻的业务量的变化也带来对物理资源占用量的差异，会引起对内存访问和修改的频率和数量变化，因此，在虚拟机动态迁移过程中每轮迭代发送的脏页数也是变化的. 所以，迭代过程中产生脏页率客观反映了业务特征，是迭代过程中分配物理网络带宽的主要依据. 在实际应用中，相同服务的业务量也会随着时间变化，存在应用服务对访问内存数据量突然增大的情况. 若此时处于虚拟机迁移过程，内存被修改的页面数量瞬间增加，带来本轮产生的脏页率也会随之增大，从而导致后续迭代时间增加和内存脏页的大量重复发送，影响迭代效率. 最终造成迁移发送带宽分配错误，影响动态迁移性能甚至导致动态迁移失败.

为了解决上述问题，本文引入带宽调整系数 $\alpha\in[0,1]$，不仅考虑业务当前运行所使用的带宽和虚拟机迁移过程中历史脏页率反映的业务特征，在考虑业务运行过程数据量抖动性的同时，还可以较准确得到实时业务特征以及本轮迭代业务的带宽使用量. 因此，我们使用式(4)计算第 k 轮迭代过程中

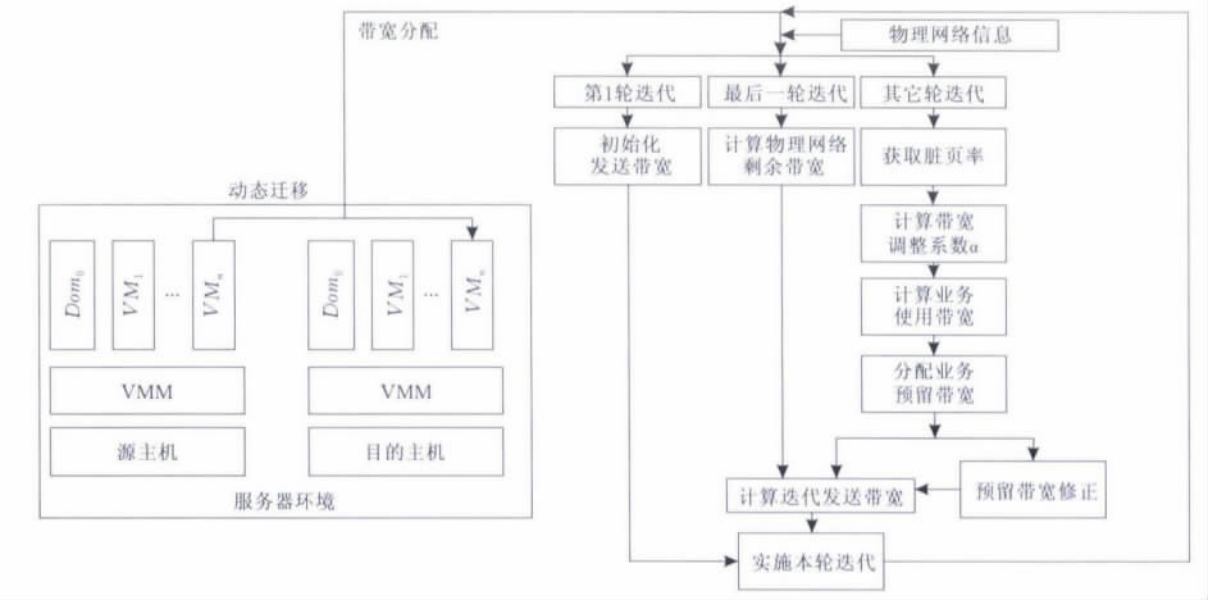

图 1 迁移带宽分配流程

业务使用带宽量.

$$B\_bw=\alpha\times AVG\left(\sum_{i=1}^{k-1}U\_dirty_i\right)+(1-\alpha)\times U\_dirty_k \tag{4}$$

U_dirty 为历史脏页率,例如在 Xen 中,通过函数 $xc\_shadow\_control()$ 可以获取每轮迭代发送时产生的脏页率. 带宽调整系数 α 反映了分配带宽时历史数据和当前数据的影响比例,当 $\alpha\in[0,0.5)$ 时,表示业务量突然减少,前 i 次迭代期间网络带宽使用量比本轮的大时,应当侧重使用历史数据作为参考;当 $\alpha=0.5$ 时,表示当前业务量稳定,网络带宽使用量比较均匀;当 $\alpha\in(0.5,1]$ 时,表示当前业务量突然增大,为了防止后续业务量增加与迁移拷贝过程发生带宽抢占现象,应当侧重使用本轮值作为参考.

带宽调整系数 α 不仅影响预留带宽值,还会影响优化算法的可行性以及迁移性能和业务服务质量. 由经验可知,该系数可在每轮迭代开始前,利用当前物理带宽使用量与物理带宽总量的比值确定是比较恰当的. 动态调整 α 值能够真实地反映当前物理网络资源的使用情况,及时有效地根据被迁移虚拟机产生的脏页率预测出本轮迭代过程中业务所需的物理带宽.

3.3 算 法

本节详细给出面向业务的虚拟机迁移带宽自适应分配算法(Adaptive Bandwidth Allocation,ABA). 在初始化阶段,一是获取当前物理机所连接网络的信息,包括总带宽量、现阶段网络使用量,并计算物理空闲网络带宽;二是根据迭代轮数,确定带宽分配规则. 当为首轮拷贝时,需要拷贝所有内存页面,因此根据现阶段网络状态初始化发送带宽量即可,当出于宕机拷贝(last_iter)阶段时,虚拟机为宕机状态,对业务无响应,因此将空闲带宽全部用于传送内存页面,以保证最短时间内内存页面传送完毕;当为其它迭代拷贝轮数时,进入第 2 阶段.

迭代运行阶段分计算预留带宽和修正预留带宽两个步骤. 第 1 步业务预留带宽计算,根据初始化阶段获取的物理网络信息计算系数 α,并利用脏页率计算业务预留带宽量,为了避免数据抖动对预留带宽计算结果的影响,利用式(4)进行计算. 第 2 步预留带宽修正,第 1 步中虽然优化了数据抖动对预留带宽计算的影响,但业务变化引起脏页率变化,可能会造成计算得出的业务预留带宽大于现阶段空闲带宽. 如果出现此种情况,需要对业务预留带宽量进行修正. 具体方法为,利用现阶段使用带宽量和计算得出的业务预留带宽之间的比值,对现有空闲带宽进行调配. 第 3 步,计算迭代发送带宽. 根据第 2 步计算结果,计算出本轮迭代发送使用带宽,然后进入页面迭代发送. 详细算法见算法 1.

算法 1. 自适应迁移带宽分配算法(ABA 算法).

输入:迭代轮次 $iter$

输出:本轮次分配带宽 E_bw

1. IF($iter==1$) THEN //第 1 轮迭代
2. 初始化迭代发送带宽 E_bw
3. return E_bw
4. ENDIF
5. 获取物理网络总带宽 T_bw,物理使用带宽 U_bw

6. 计算物理空闲带宽 $F\_bw=T\_bw-U\_bw$
7. IF($iter==last\_iter$) THEN //最后一轮迭代
8. $E\_bw=F\_bw$
9. return F_bw
10. ENDIF
11. 计算调节系数 $\alpha=U\_bw/T\_bw$
12. 获取本轮次脏页率 $U\_dirty[iter]$
13. 计算前 $iter-1$ 轮脏页率的平均值

$$D\_avg=\Sigma_i U\_dirty[i]/(iter-1)$$

14. 计算业务当前预留带宽

$$B\_bw=\alpha\times D\_avg+(1-\alpha)\times U\_dirty[iter]$$

15. IF($B\_bw>F\_bw$) THEN
16. 修正业务预留带宽

$$B\_bw=F\_bw\times(B\_bw/(B\_bw+U\_bw))$$

17. ENDIF
18. return $E\_bw=F\_bw-B\_bw$

4 系统实现与性能分析

4.1 基于 Xen 的系统实现

开源的 Xen 虚拟化平台已经成为虚拟化领域的主流技术，因此本文基于 Xen 平台实现了 ABA 算法. 由于 Xen 自身不具有获取当前物理网络环境的功能，根据本文算法的需求，需要在迁移源码中增加物理带宽获取模块，由于在 Xen 的半虚拟环境中迁移过程需要陷入到操作系统中调用超级调用(Hypercall)实现对敏感指令的模拟执行，当在迁移源码中调用 Linux 系统函数带宽获取模块时会出现系统调用冲突，造成动态迁移失败. 为此，本文将获取物理网络函数编写成独立的多线程守护进程，迁移过程中所需要的物理网络信息全部以文件流的方式获取. 这样可以减少对迁移过程的影响.

本文获取当前物理网络带宽的基本思路为，利用 Linux 系统函数，首先根据物理机网卡的相关信息获取当前网络的物理总带宽，然后每间隔 t 秒收集并记录一次网络收包和发包的具体数量，最后利用式(5)和(6)计算出当前已用物理带宽和空闲物理带宽.

$$U_{bw}=\frac{(S_{pack_i}-S_{pack_{i-1}})+(R_{pack_i}-R_{pack_{i-1}})}{t} \quad (5)$$

$$F_{bw}=T_{bw}-U_{bw} \quad (6)$$

其中，U_{bw} 为当前已用物理带宽，F_{bw} 为当前空闲物理带宽，T_{bw} 为当前物理总带宽，S_{pack} 和 R_{pack} 分别存放物理机发送和接收数据包的数量. 在式(6)中通过物理机的接收和发送数据包的数量计算出现有物理网络中已使用的网络带宽，此带宽不仅包括当前迁移虚拟机时所使用的带宽，还包括物理机中其它虚拟机和其它程序所使用的网络带宽量. 现实中业务量的大小是随机变化的，这里时间 t 的选择会直接影响 U_{bw} 取值，例如：当间隔时间 t 取值过小时，会出现 S_{pack_i} 与 $S_{pack_{i-1}}$ 或 R_{pack_i} 与 $R_{pack_{i-1}}$ 两两之间的差值为零，无法得到 U_{bw} 的值，造成带宽分配失败；当间隔时间 t 过大时，由于业务量随机变化，造成 U_{bw} 不能精确地体现当前物理带宽的使用量. 根据经验，本文设定间隔时间 t 为 1 s.

4.2 实验环境

实验平台选用了配置为 AMD Opteron 四核处理器(2.0 GHz)、40 GB 内存和 SCSI 磁盘接口的曙光服务器. 在千兆网络中使用 NFS(Network File System)共享存储模式，并采用内核 2.6.18 的 Linux 操作系统和 Xen-4.1.0 虚拟化平台，在半虚拟化环境中配置 1 个 VCPU，2 GB 内存和 10 GB 磁盘空间的 Linux 操作系统的虚拟机. 迁移不使用专用网络，与物理机共享同一网络. 并选择 TPC-W① 和 SPECweb2009② 两种 Benchmark 代表网络密集型业务和内存密集型业务.

为了验证在 I/O 密集环境下利用 ABA 算法实施虚拟机迁移的优化效果，利用上述 Benchmark 分别对 Xen 自有的两种传输算法和 ABA 算法进行迁移时间、宕机时间、设备迁移时间和服务质量等方面的对比性测试. 下述实验中 Xen_0 表示 Xen 自有的无带宽分配算法，Xen_50 表示 Xen 自有的增量带宽分配算法，Xen_ABA 表示本文提出的 ABA 迁移算法.

4.3 迁移性能测试与分析

首先测试不同环境下业务量对宕机时间、设备传输时间、迭代发送时间和迁移总时间的影响. TPC-W 环境下的测试结果如图 2(a)、(b)所示. 在图 2(a)中，柱形框表示不同迁移算法下的宕机时间，折线表示了不同迁移算法下的设备传输时间. 实验结果表明 ABA 算法比 Xen 自有的算法带来宕机时间缩短了 50% 以上，设备传输时间也有明显减少，并且随着业务量的增加宕机时间波动很小. 从图 2(b)记录的迭代时间和迁移总时间可以看出，ABA 算法在迭代时间和迁移总时间两方面都低于增量分配算法，略高于无带宽分配算法，结合图 2

① TPC Transaction Processing Performance Council. http://www.tpc.org/tpcw. 2012-03-10
② Spec Standard Performance Evaluation Corporation. http://www.spec.org/web2009. 2012-03-10

(a)可知,ABA 算法在迭代期间需要预留业务使用带宽,才造成迭代时间略高于无带宽分配算法,迭代发送过程虚拟机处于运行状态,加之业务预留带宽的存在,因此不会对业务性能有过大的影响.

SPECweb 环境下的测试结果如图 2(c)和(d)所示. 在图 2(c)中,柱形框表示不同迁移算法时的宕机时间,折线表示了不同迁移算法时的设备传输时间. 从图 2(c)中可以看出对于内存密集型业务来说 3 种算法在宕机时间上的差别很小,但 ABA 算法可以减少设备传输时间. 整体波动较为平缓. 在图 2(d)中的显示可知,ABA 算法的迭代时间和迁移总时间还是优于增量带宽分配算法,由于预留业务使用带宽的原因,迭代时间和迁移总时间略高于无带宽分配算法.

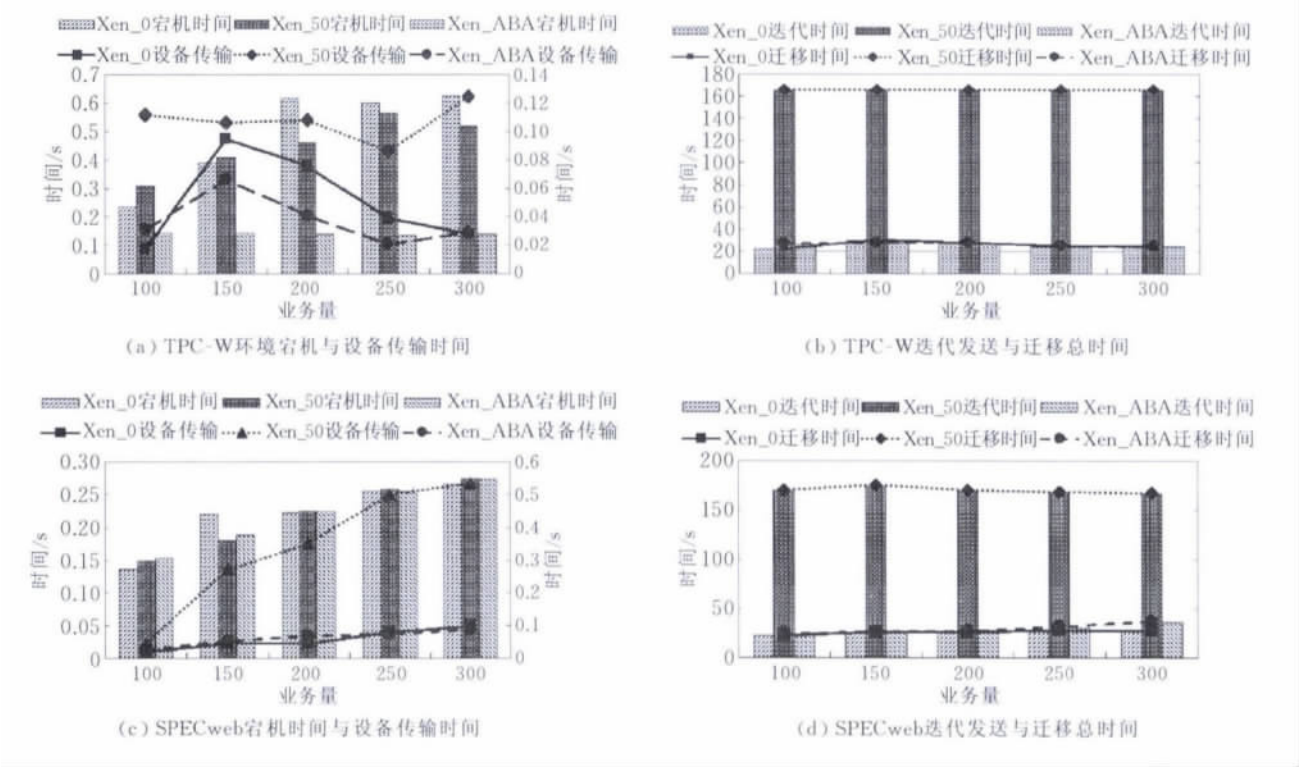

图 2 迁移性能测试与分析

4.4 业务服务质量测试与分析

随后,我们测试了不同环境下虚拟机动态迁移的内存发送速率以及对运行业务的服务质量的影响,其中运行业务的服务质量是由测试使用的 Benchmark 自动统计生成. TPC-W 环境下的测试结果如图 3(a)和(b)所示. 从图 3(a)可以看出 ABA 算法在发送速率方面总是低于 Xen 自有的两种算法,在无带宽控制时脏页率与带宽发送时间无明显规律;增量带宽分配时,随着脏页率的增加发送速率也呈上升趋势,ABA 算法中,发送速率会随脏页率的变化而动态调整并为业务预留部分带宽. 在图 3(b)中,基准 QoS 表示系统自动计算的理论业务服务质量参考值,在不同业务量下,实际业务服务质量越接近该值说明其服务质量越好,实验结果表明使用 ABA 算法迁移时,业务的服务质量总是优于 Xen 自有的两种方法,这说明在 ABA 算法中预留业务带宽提高了迁移过程中的业务服务质量. 结合图 2(a)、(b)与图 3(a)、(b)可以得出在网络密集型业务环境中 ABA 算法在迁移过程中预留了网络带宽,使迭代发送时间和迭代总时间略高于无带宽分配算法,但其宕机时间最短以及业务服务质量最优.

SPECweb 环境下的测试结果如图 3(c)和(d)所示. SPECweb 属于 I/O 密集型业务,对内存使用较为频繁,单位时间内产生的脏页较多,从图 3(c)所示可知,ABA 算法根据脏页率判断出业务需求带宽量较大,为了保证服务质量,在带宽分配时业务预留带宽占的比重较大,因此发送使用带宽量较小,最终造成宕机时间和迁移时间有所增加,这与上节的分析结果吻合. 本文按算法类型分类,将不同业务量下成功率、错误率和容忍率取平均值后进行统计,如图 3(d)所示. 使用 ABA 算法迁移时,

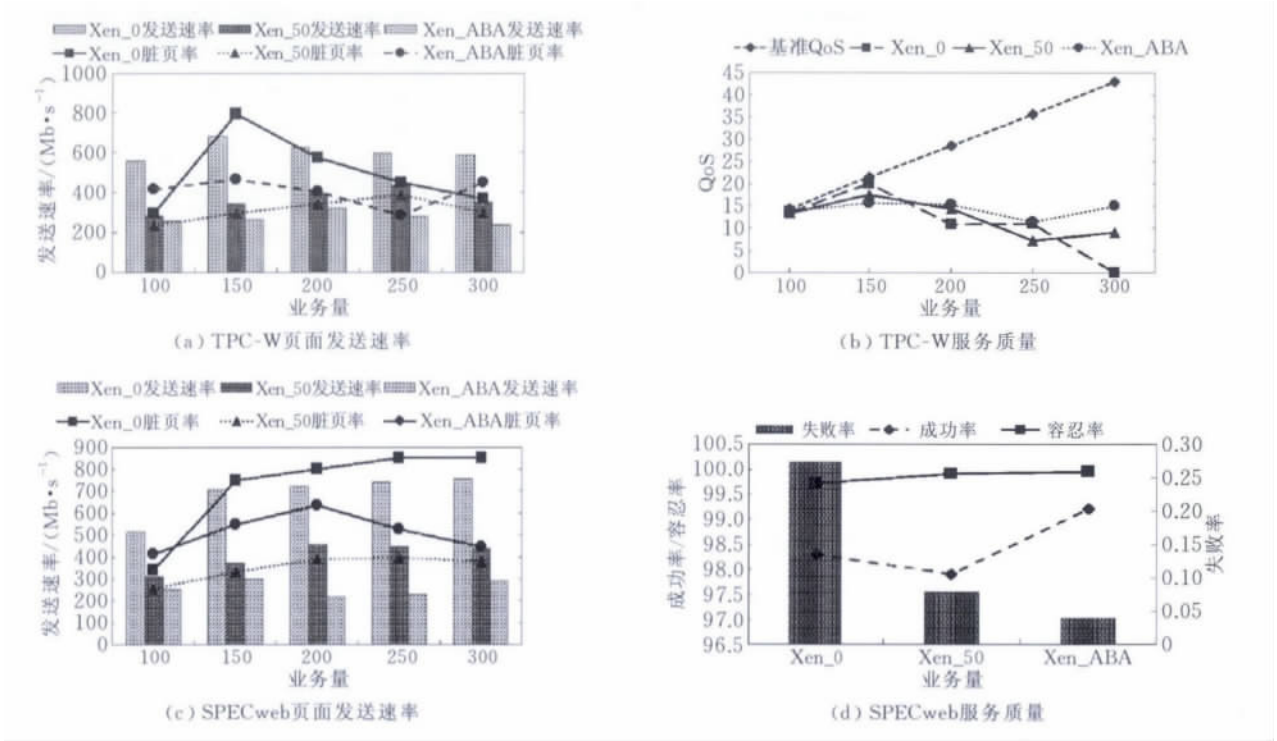

图 3　业务服务质量测试与分析

业务服务的成功率在 99%以上，比使用 Xen 自有方法提高了 2%左右；在容忍度方面达到了 99.96%，业务请求基本上能全都得到响应. 从错误率的角度来看，ABA 算法在 0.05%以下，分别比无带宽算法和增量带宽算法降低了 20%和 10%左右. 因此，可以看出 ABA 算法在迁移过程中确保业务服务质量.

5　总结与展望

虚拟化技术能够在云计算平台中创建多个虚拟机作为虚拟操作平台，满足多用户共享隔离执行环境的需求，并通过虚拟机迁移技术灵活和动态地管理物理资源，实现高效绿色资源管理. 在实际应用过程中，不同业务特性等因素对虚拟机迁移效率有很大影响，虚拟机动态迁移过程需要传输大量内存页面，造成动态迁移与应用程序性能下降. 为解决这一问题，本文提出一种面向业务特征的虚拟机动态迁移带宽分配算法，通过实时分析运行业务的特征，利用迁移迭代过程中内存脏页率和带宽调整系数，预测下一时刻运行业务对物理网络带宽的使用量，从而自适应分配虚拟机动态迁移使用的物理带宽，减少了迭代时间和宕机时间. 实验结果表明，本算法能够在物理带宽资源有限的前提下，合理利用空闲物理带宽资源，减少虚拟机迭代时间和宕机时间，提高虚拟机迁移性能和业务服务质量.

未来工作将针对复杂网络环境的迁移带宽分配优化，解决多层次复杂异构网络环境中虚拟机动态迁移的通用性与适应性；增加相关约束条件，更精确地分配虚拟机动态迁移过程中的使用带宽，以提高虚拟机迁移性能和业务服务质量.

致　谢　感谢中国科学院计算技术研究所计算机体系结构国家重点实验室孙毓忠教授和山东大学计算机科学与技术学院系统安全与隐私保护实验室季大祥同学对本文的建议和帮助！

参　考　文　献

[1] Nathuji R, Schwan K. Virtual power: Coordinated power management in virtualized enterprise systems//Proceedings of ACM Symposium on Operating Systems Principles (SOSP'07). Stevenson, USA, 2007: 265-278

[2] Clark C, Fraser K, Hand S, et al. Live migration of virtual machines//Proceedings of the 2nd Symposium on Networked Systems Design and Implementation NSDI'05. Boston, USA, 2005: 273-286

[3] Wood T P, Shenoy P A, Venkataramani A, Yousif M. Black-box and gray-box strategies for virtual machine migration// Proceedings of the 4th USENIX Symposium on Networked

Systems Design and Implementation (NSDI'07). Cambridge, USA, 2007: 229-242

[4] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization//Proceeding of the 19th ACM Symption on Operating Systems Principles (SOSP'2003), 2003: 164-177

[5] Nelson M, Lim B-H, Hutchins G. Fast transparent migration for virtual machines//Proceedings of the Annual Conference on USENIX Annual Technical Conference. Anaheim, CA: USENIX Association, 2005: 15-35

[6] Nagarajan A B, Mueller F, Engelmann C, Scott S L. Proactive fault tolerance for HPC with xen virtualization//Proceedings of ACM Annual International Conference on Supercomputing (ICS'07). Seattle, USA, 2007: 23-32

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

[8] Chen Yang, Huai Jin-Peng, Hu Chun-Ming. Live migration of virtual machines based on hybrid memory copy approach. Chinese Journal of Computers, 2011, 34(12): 2278-2291(in Chinese)
(陈阳, 怀进鹏, 胡春明. 基于内存混合复用方式的虚拟机在线迁移机制. 计算机学报, 2011, 34(12): 2278-2291)

[9] Hines M R, Deshpande U, Gopalan K. Post-copy live migration of virtual machines. SIGOPS Operation System Review, 2009, 43(3): 14-26

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

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

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

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

[14] Huang W, Gao Q, Liu J, Panda D K. High performance virtual machine migration with RDMA over modern interconnects//Proceedings of the IEEE International Conference on Cluster Computing (Cluster'07). Austin, Texas, USA, 2007: 11-20

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

[16] Jin Hai, Gao Wei, Wu Song, et al. Optimizing the live migration of virtual machine by CPU scheduling. Network and Computer Applications, (2010), doi: 10.1016/j.jnca.2010:1088-1096

[17] Liu H, Jin H, Liao X, et al. Live migration of virtual machine based on full system trace and replay//Proceedings of the 18th International Symposium on High Performance Distributed Computing (HPDC'09). Munich, Germany, 2009: 101-110

[18] Liu Shi-Hai, Sun Yu-Qing, Shi Wei-Qi, Gao Yun-Wei. Rapid migration method of virtual machine for extensible cluster environment. Journal of Southeast University (Natural Science Edition), 2011, 41(3): 468-472(in Chinese)
(刘诗海, 孙宇清, 石维琪, 高云伟. 一种面向可扩展集群环境的快速虚拟机迁移方法. 东南大学学报(自然科学版), 2011, 41(3): 468-472)

[19] Riteau P, Morin C, Prio T. Shrinker: Efficient live migration of virtual clusters over wide area networks. Concurrency Computat: Pract. Exper., 2013, 25: 541-555

[20] Bradford R, Kotsovinos E, Feldmann A, Schioberg H. Live wide-area migration of virtual machines including local persistent state//Proceedings of the 3rd International Conference on Virtual Execution Environment (VEE'07). San Diego, California, USA, 2007: 169-179

LIU Shi-Hai, born in 1982, M. S.. His research interests include virtualization technology and system security.

SUN Yu-Qing, born in 1967, professor. Her research interests include system security and privacy protection.

LIU Gu-Yue, born in 1990, M. S.. Her research interest is virtualization technology.

Background

Live migration of virtual machine (VM) is an important technology in green cloud computing environment, by which a virtual machine monitor moves an entire running VM from one physical machine to another. Iteration time and downtime are regarded as important indexes to evaluate migration performance, which highly depends on the assigned network bandwidth and the dirty-page rate of transferred VM memory page in a migration process. Existing live migration mechanisms have to be in the face of some problem such as convergence of iteration, redundance of copying and guest OS transparence. This paper propose an Adaptive Bandwidth Allocation Algorithm for live migration of virtual machine based on service features. By analyzing the dirty-page rate in each iteration of migration process, the algorithm predicts the network traffic of current tasks and allocates the migration bandwidth accordingly. By introducing the adjustment coefficient of bandwidth, our algorithm can handle the data jitter in practical service running so as to make the assignment of migration bandwidth more reasonable. We perform a set of experiments to evaluate our method on running services with different features. The results show that it can well utilize the idle bandwidth and improves the performance of live migration on both downtime and the overall migration time. It also improves the system reliability and provides a better service quality for users. Part of this work is supported by the National Natural Science Foundation of China (61173140), the National Science & Technology Pillar Program (2012BAF10B03-3), the Open Funding of Key Laboratory of Computer System and Architecture of Chinese Academy of Sciences (ICT-ARCH200904).