PAMM:一种面向基于内存共享的域间通信的优化模型

孙瑞辰 孙 磊

(信息工程大学密码工程学院 郑州 450000)

摘 要 云计算平台和虚拟化技术的结合为虚拟机域间通信带来了新的需求,基于内存共享的域间通信可以提高运行在同一物理机上的虚拟机间的通信效率。但是,基于内存共享的域间过程中产生的上下文状态切换限制了其优化能力。引入一种新的内存共享模型PAMM,即通过添加一个管理模块对内存共享过程中所传递的内存页进行聚合管理,减少申请超级调用的次数,以达到减少状态切换的目的。实验表明,PAMM能够提升基于内存共享的域间通信的通信效率。

关键词 云计算,Xen,域间通信,内存共享

中图法分类号 TP314 **文献标识码** A

PAMM:An Optimized Module for Inter-domain Communication Based on Shared Memory

SUN Rui-chen SUN Lei

(Institute of Cryptographic Engineering, Information Engineering University, Zhengzhou 450000, China)

Abstract The combination of cloud computing platform and virtualization technologies has brought new communication requirements to us. The inter-domain communication based on shared memory can improve the efficiency of the communication between the virtual machines running on the same physical machine. But the status-sitching in the process of memory sharing limits the optimization ability of the proposed method. New memory sharing model PAMM, by adding a management model which can aggregate memory pages in the process of memory sharing and reduce the number of super calling application, achieves the purpose of reducing the number of status-switching. The experiment shows that PAMM can enhance the efficiency of inter-domain communication based on shared memory domains.

Keywords Cloud computing, Xen, Inter-domain communication, Shared memory

1 引言

近年来,云计算持续升温,虚拟化技术得到了快速发展。虚拟化技术能够大大提高硬件的使用率,并具有安全可靠、可管理性强等特点,对资源的高效利用有重大意义。随着VMware、KVM、Xen等一批性能优异、运行稳定的虚拟化产品的出现,越来越多的存储资源和计算资源部署在云端。通过虚拟化技术,云计算服务中心可以动态地对各种计算资源进行组织,最大限度地减少能源开销,灵活满足不同的环境需求,提高云计算资源的运算速度和使用效率。

在虚拟化技术应用中,每台物理服务器上运行多台虚拟机,虚拟机上运行的服务可能分属不同层次,同一物理机器上虚拟机间的域间通信十分频繁。为满足虚拟机间隔离性和安全性需求,虚拟机的域间通信会产生大量的额外性能开销。通过半虚拟化中提供的内存共享机制,不同虚拟机间可以进行共享数据传递,这为同物理节点上不同虚拟机间的域间通信提供了新的实现方式。

本文从优化域间通信中内存共享过程的角度出发,分析了在内存共享过程中导致实际通信性能下降的原因,从而提出了一种基于内存页聚合的新通信模型PAMM(Page Aggregation Manager Model),对基于内存共享的域间通信进行了优化,并通过实验对使用PAMM后,域间通信性能的提升进行了验证。

本文第2节介绍了现有的基于内存共享的域间通信方法和这些方法的不足;第3节对Xen的内存共享机制进行了分析;第4节详细描述了内存页聚合管理模型的实现原理和结构组成;第5节给出了实验平台、实验设计和实验结果;最后对全文进行总结并对未来工作进行展望。

2 相关工作

针对基于内存共享的虚拟机高速的域间通信,已有的解决方案主要分为3类:第一类以Xensocket[1]为代表,通过对应用程序进行修改,使应用程序在调用通信操作时,直接使用xensocket,利用内存共享进行数据传递。第二类以IVC[2]为代表,通过对应用程序库进行修改,使系统在通信时仍使用原有的通信接口,但调用的库函数则被修改成使用内存共享进行通信。第三类以XenLoop[3]为代表,采用内核模块机制,使数据在数据链路层通过内存共享进行发送和接收。在这3类方法的基础上,出现了更多的域间通信性能优化方法[4-7]。

通过内存共享进行通信可以解决同一台物理机器上各虚拟机间通信的瓶颈[8],实现虚拟机管理器和虚拟机间的通信,并用于同物理机上虚拟机间的本地通信。相比传统的Xen

孙瑞辰(1991—),男,硕士生,主要研究方向为虚拟化技术、信息安全;孙 磊(1973—),男,博士,副研究员,主要研究方向为云计算基础设施可信增强、可信虚拟化技术。

通信机制，基于内存共享的域间通信在通信带宽和 CPU 使用效率上都有较大提升[9]。但是随着通信数据输入率的增加，这些方法提供的实际性能与内存共享提供的理论性能间的差距逐步增大。

3 Xen 内存共享机制

现有域间通信方法通常利用内存共享来替代传统的网络通信方式[10]。本节首先介绍 Xen 的内存共享原理，而后对页面传递过程进行分析，最后给出影响通信性能的因素。

3.1 Xen 的内存共享原理

通过虚拟机监视器，Xen 可以对运行在其上的虚拟机进行一些基本控制。内存共享技术基于授权表、事件通道等控制机制，通过内存进行数据通信。Xen 的内存共享方式有页面映射、页面传递、页面拷贝[11]。

页面映射主要用于 Domain 之间的共享访问，在目标 Domain 访问期间，被访问 Domain 不失去该内存页的所有权，目标 Domain 仅仅是映射内存页到自己的地址空间内，在完成访问后还需撤销映射。

页面传递与页面映射在操作上类似，都是映射原 Domain 提供的内存页到目标 Domain 的地址空间中。它们的区别是原 Domain 在页面传递中会将所传递内存页的所有权转交给目标 Domain，自身失去所传递内存页的所有权。页面拷贝主要用于小数据的传递，数据的大小不能超过一个内存页的容量。由于在该过程中只涉及内存页中数据的一部分，不需要进行更新页表操作，因此系统开销相对较低[8]。

虚拟机的域间通信主要通过页面传递的方式进行，本文将针对页面传递方式进行分析与优化。

3.2 页面传递过程分析

在传统的页面传递中，数据的传递通过授权表机制完成[12]，使用授权项(Grant Entry)来定义共享内容的授权信息。授权项中包含授权类型、授权访问的 Domain ID 和授权的共享内存信息。

授权项结构体如下：

```
struct grant_entry{
    uint16_t flags;/*授权类型*/
    domid_t domid;/*授权访问 Domain*/
    unit32_t frame;/*授权共享的内存页*/
};
```

授权项结构体包含 3 个成员 flags、domid 和 frame。授权类型相关标志位(flags)由 Guest OS 修改和访问；目标 Domain 的 ID(domid)只能由 Guest OS 修改，Xen 访问；授权内存页号(frame)，由 Xen 修改，Guest OS 访问。内存共享的授权以内存页为基本单元，每个授权项被一个整数键值标识，称为授权引用(Grant Reference，GR)。

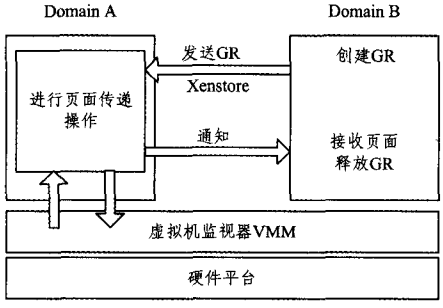


图 1 页面传递过程

在页面传递过程中，Domain A 要将自己的内存页传递给目标 Domain B，其传递过程如图 1 所示，步骤如下：

1)Domain B 创建授权引用 GR，发送给 Domain A；

2)Domain A 申请页面传递操作对应的超级调用，通过上一步接收的 GR 得到共享信息，完成内存页传递；

3)Domain B 接收所传递的内存页；

4)释放授权引用 GR；

整个传递过程中主要的操作有：创建授权引用 GR、发送授权引用 GR、申请超级调用、接收内存页、释放授权引用。下面通过分析这些操作的实现过程来寻找影响数据通信性能的主要因素。

3.3 影响通信性能的因素

3.3.1 授权引用 GR 的相关操作

授权引用 GR 是一个整数键，每一个授权引用对应着一个授权项。如图 1 所示，创建授权引用 GR 是整个页面传递操作的开始，由 Domain B 通过调用函数来完成，操作范围仅在 Domain B。与之相对，释放授权引用 GR 是整个页面传递操作的结束，同样由 Domain B 通过调用函数来完成的。

Xenstore 是一个域间共享的存储系统，由特权虚拟机(Dom0)管理。Xenstore 提供了列出目录、读取键值、写入键值以及键值变化时发出通知等功能。页面传递过程中 GR 的传递由 Guest OS 通过 Xenstore 对其进行读取和写入操作完成。

3.3.2 申请超级调用

页面传递操作对应超级调用的结构体参数为 gnttab_transfer：

```
#define GNTTABOP_transfer
struct gnttab_transfer{
    xen_pfn_t mfn;/*将要传递的内存页*/
    domid_t domid;/*接收者 Domain*/
    grant_ref_t ref;/*接收者 Domain 授权引用*/
    int16_t status;
};
Typedef struct gnttab_transfer gnttab_tra nsfer_t;
DEFINE_Xen_GUEST_HANDLE(gnttab_transfer_t);
```

gnttab_transfer()是操作的核心函数，它通过调用 guest_physmap_add_page()，根据输入的参数将内存页添加到目标 Domain 的地址空间。超级调用类似于本地操作系统中的系统调用，通过软中断由非根状态进入根状态，完成超过自身操作权限的操作。

4 内存页聚合管理模型 PAMM

本节分别从基本原理、设计和实现两方面介绍了内存页聚合管理模型 PAMM(Page Aggregation Manager Model)。

4.1 基本原理

根据第 2 节分析，基于页面传递的内存共享可以在以下 3 个方面进行优化：

1)优化授权引用 GR 的授权和撤销过程；

2)优化授权引用 GR 的接收和发送过程；

3)优化通信机制，减少由于申请超级调用产生的性能开销。

在 Xen 的内存共享数据通信相关操作中，状态切换操作带来的性能开销远大于其他操作[13]，而频繁地申请超级调用

会带来大量状态切换操作，因此本文选择从第 3 个方面进行优化。

页面传递过程中每次页面传递操作进行一次超级调用的申请，但在每次页面传递中，所传递内存页的数量不固定。对于相同数据，通过对传递内存页进行管理，增加每次页面传递中包含的内存页数，可以减少通信过程中申请超级调用的次数。同时，通过对发送条件进行设定，可以保证域间数据通信的实时性。根据此原理，本文设计了内存页聚合管理模型 PAMM。

4.2 PAMM 的设计和实现

为不改变 Xen 保持自身内核尽量小的设计初衷，优化工作选择在 Guest OS 中完成。内存页聚合管理模型 PAMM 在原页面传递通信模型的基础上进行了改进，通过增加一个包含内存页容器的管理模块，对多次页面传递进行整合管理，达到减少超级调用次数的目的。

在接收到对应系统调用后，PAMM 会对页面传递操作中所传递的内存页进行计数，对传递内存页数较少的相邻页面传递操作进行聚合。根据传递的目标 Domain ID，内存页会被装入对应的内存页容器中。

内存页容器的结构体如下：

```
struct pagecontains{
    domid_t dom;/*目标 Domain ID*/
    int num;/*容器中内存页数*/
    int maxsize;/*设定的内存页上限数*/
    struct timeval timeout;/*设定的最大延迟时间*/
    xen_pfn_t mfn;/*将要进行传递的内存页*/
}
```

结构体中包含了 5 个参数 dom、num、maxsize、timeout 和 mfn，分别代表该容器的目标 Domain ID、容器中内存页数、容器内存页上限数、最大延迟时间和容器中存储内存页的编号。页面传递操作发生时，内存页管理模块首先查看授权项中的 domid 参数，若已有与目标 domain 与 domid 参数对应的容器，则选择该容器，进行转入操作；若没有对应容器，则建立与该 domid 对应的新容器。转入操作首先对授权项中的 frame 参数进行计数，并与容器的 num 参数进行累加赋值。若所得结果大于参数 maxsize，则立刻向目标 Domain 进行页面传递；若所得结果小于参数 maxsize，则将所传递的内存页存入容器中进行寄存，暂不进行页面传递操作，转而等待下一个转入操作。为防止等待时间过长而影响传递实时性，PAMM 设置了最大延迟时间 timeout。在初始化后的第一个装入操作发生时，容器定时器开始计时，在等待时间到达参数 timeout 后，将强制进行页面传递操作。一旦页面传递操作进行，则参数 num 和 timeout 归零。PAMM 结构如图 2 所示。

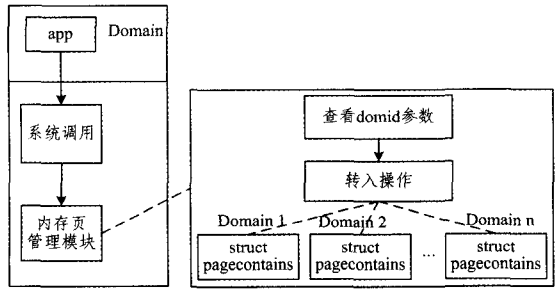


图 2 PAMM 示意图

采用 PAMM 后，页面传递的过程如图 3 所示，其步骤如下：

1）Domain B 创建授权引用 GR，发送给 Domain A；

2）判断是否进行页面传递操作；

3）根据判断结果进行等待或者页面传递操作，若进行等待，将要传递的内存页存入容器，若进行页面传递操作，Domain A 申请超级调用；

4）Domain B 接收所传递内存页；

5）清除授权引用 GR。

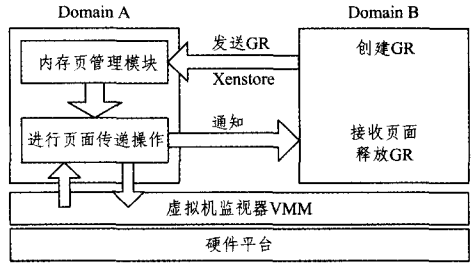


图 3 采用 PAMM 后的页面传递过程

管理模块在收到第一个授权引用 GR 后开始工作，并根据自身的设定值来判定页面传递操作的开始。管理模块在接收到授权引用后的工作流程如图 4 所示。

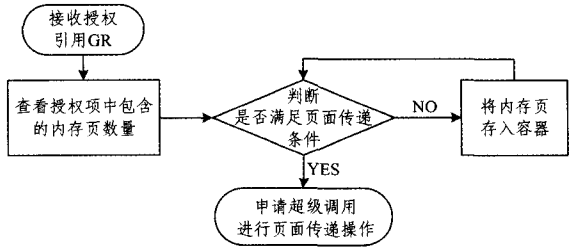


图 4 管理模块工作流程图

PAMM 通过增加每次页面传递的内存页数，减少了申请超级调用的次数，进而达到了减少根-非根状态的切换、提高数据传递性能的目的。下面将通过与未优化页面传递过程进行对比实验来检验 PAMM 的性能。

5 实验与分析

5.1 实验平台

开发环境：CentOS6.5 + Xen4.2.4，在该环境中实施 PAMM 性能测试实验。

测试平台：物理主机 + 特权虚拟机 + 客户虚拟机

物理主机：Core i5 430 CPU，海力士 DDR3 4GB 内存

特权虚拟机：Linux CentOS 6.5 操作系统

客户虚拟机：Linux CentOS 6.5 操作系统

在测试平台上建立两个 Linux CentOS 6.5 虚拟机 VM1 和 VM2，两台虚拟机的虚拟内存设置为 512M。

5.2 实验设计

在虚拟机 VM 中，PAMM 调用是通过对 Guest OS 操作系统的系统调用表进行修改来完成的，将原系统调用修改为内存共享所需要的系统调用，即可实现 PAMM 的调用。

系统调用统一存放在系统调用表（sys_call_t able）内，该表拥有一个起始地址，系统调用根据各自的系统调用号乘以

4 得到偏移地址，通过替换内存中存放的地址指针得到需要完成的服务例程。

通过对 map 文件进行 grep 操作得到地址后，将 recvfrom、sendmsg、write、read、close 等需要修改的调用替换成内存共享对应得系统调用。在修改过程中，由于系统调用表的分配表内存拥有写保护机制，在进行换时，必须将 CR0 的 20 位进行记录后清除，并在完成系统调用替换后，恢复 CR0 原始值。系统调用表的修改过程如表 1 所列。

表 1　系统调用表修改过程

输入：cr0 地址，替换的系统调用名
输出：空
1. 读取 cr0 地址；
2. 记录 cr0，ret←cr0；
3. 清除 cr0，cr0←0；
4. 对系统调用进行替换，volatile("movl" %%替换的系统调用名)；
5. 返回原 cr0 值，return(cr0)；
6. 恢复 cr0，setback_cr0(ret)；

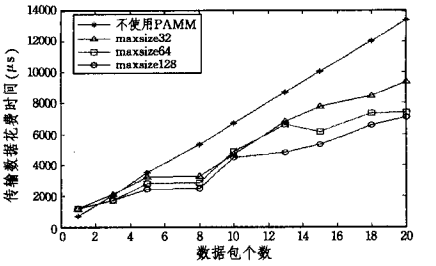
实验从传输时间和 CPU 利用率两个方面对使用普通页面传递机制和使用 PAMM 后的数据传递过程进行对比，使用一个发送对象为可改变数量的多个固定大小文件的发送函数作为虚拟机间的数据通信任务。实验在不使用新模型和使用新模型但改变容器中内存页上限参数 *maxsize* 的多种情况下进行传输时延和各虚拟域的 CPU 利用率测试，每种情况下进行一组测试，每组进行 10 次，每次测试采取相同的传输函数和相同的系统环境，最后对两个指标进行统计，取平均值作为实验结果。

在优化模型中，管理模块的参数设置了不使用 PAMM、使用 PAMM 并设置 *maxsize*＝32、64、128 4 种情况，最大延迟时间参数 struct timevaltimeou 设置{"0","500"}、{"0","1000"}、{"0","1500"}、{"0","2000"} 4 组。为方便对比，发送函数传递内容设定为多个 64kB 大小的文件，从 VM1 传递到 VM2。在传输过程中，使用 Netperf[14] 对通信传输时间进行测试，并使用 xentop 命令提取特权虚拟机和客户虚拟机的 CPU 利用率信息。

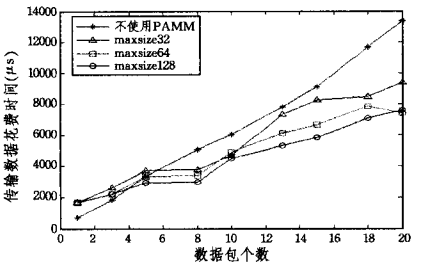
5.3　实验结果和分析

通过对实验结果取平均值，得到的传输数据所花费时间如图 5 所示，图 5(a)－图 5(d)分别对应 struct timevaltimeou 为{"0","500"}、{"0","1000"}、{"0","1500"}、{"0","2000"} 4 种情况。

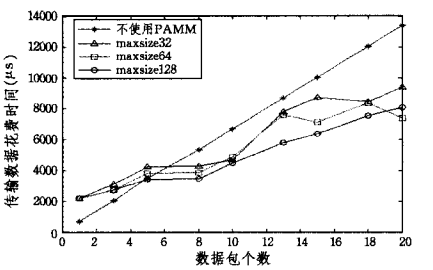
在未使用 PAMM 的数据传递中，花费时间呈稳定上升趋势，且在传输数据量比较小时，其传输耗时低于使用 PAMM 时的几种情况，这种现象是管理模块的等待机制造成的。由于通信数据量较小，未能迅速填满管理模块中的容器，因此会在达到最大等待延时后才进行数据传递操作，此时的等待延时大于页面传递操作本身所耗费时间。但随着传输数据的个数的增加，PAMM 开始体现出优势，通信数据会使容器中内存页数 *num* 迅速达到设定值，管理模块不再需要等待达到最大延时，而是在容器填满后立即进行页面传递操作。在传递文件数量大于 5 个后，优化时间明显增加。根据 *maxsize* 参数的不同设置，PAMM 在不同的传输情况下产生了不同的特征，并有着较大的差异。



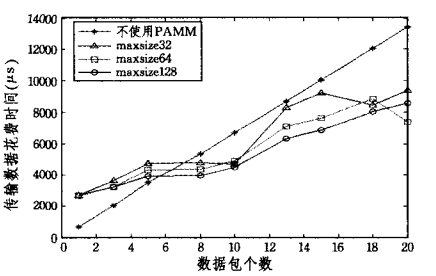
(a) struct timevaltimeou={"0","500"}



(b) struct timevaltimeou={"0","1000"}



(c) struct timevaltimeou={"0","1500"}



(d) struct timevaltimeou={"0","2000"}

图 5　数据传输通信时间

在进行上述步骤的同时，实验通过 xentop 命令得到了特权虚拟机和客户虚拟机的 CPU 利用率信息。在各 *maxsize* 对应情况下，CPU 利用率随传输文件数变化较小，选取传输文件为 8 个时的实验结果作为代表，结果如图 6 所示。

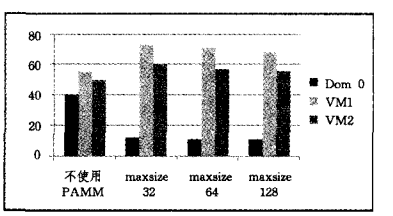


图 6　CPU 利用率信息

采用 PAMM 后，VM1 的 CPU 使用率有较大上升，而特权虚拟机的 CPU 使用率有所下降，这是由于 PAMM 减少了超级调用的申请次数，增加了 Guest OS 的系统操作，优化了 CPU 的使用效率。VM2 的 CPU 利用率则变化较小，这是由

（下转第 235 页）

[7] Ma Lian-bo, Hu Kun-yuan, Zhu Yun-long, et al. Cooperative Artificial Bee Colony Algorithm for Multi-objective Network Planning[J]. Journal of Network and Computer Applications, 2014, 42:143-162

[8] 闫家旻. 嵌入式 RFID 中间件的设计与实现[D]. 大连:大连海事大学,2013

[9] 孙喜策,曹峰,王智. 一种面向多目标关联覆盖的无线传感器网络节点优化调度算法[J]. 信息与控制,2009,38(1):30-38

[10] 邓毅华,谢胜利. 嵌入式 RFID 中间件的设计与实现[J]. 计算机工程与设计,2008,29(7):1716-1719

[11] 王汝传,孙力娟. 无线传感器网络中间件技术[J]. 南京邮电大学学报,2010,30(4):37-41

[12] Jarugumilli S, Grasman S E. RFID-enabled inventory routing problems[J]. International Journal of Manufacturing Technology and Management, 2007, 10(1):92-105

[13] Ayoade J. Security implications in RFID and authentication processing framework[J]. Computers & Security, 2006, 25(3):207-212

[14] C Chun-te, L Kun-lin, W Ying-chieh. Construction of the enterprise-level RFID security and privacy management using role-based key management[C]// IEEE International Conference on Systems, Man and Cybernetics. 2006:3310-3317

[15] Kang J W, Ahn B C, Kim K J. Evaluation of safety for the 900MHz RFID reader of defense ammunition management system[J]. International Conference on Information Science and Security, 2008, 12(66):220-223

[16] Z Min, W Li, W Zhong-yun, et al. A RFID-based material tracking information system[C]// 2007 IEEE International Conference on Automation and Logistics. 2007:2922-2926

[17] 罗刚,郭兵,沈艳,等. 程序级和算法级嵌入式软件功耗特性的分析与优化方法研究 [J]. 计算机学报,2009,32(9):1870-1873

[18] 张文新,邓毅华,谢胜利. 基于嵌入式 RFID 中间件的标签数据处理 [J]. 微计算机信息,2009,5(5-2):182-190

[19] 韩磊,梅佳希,袁巍,等. 基于语义分析的嵌入式 RFID 中间件研究 [J]. 微计算机信息,2010,26(5-2):133-136

[20] 马连博,胡琨元,朱云龙,等. 基于 RFID 的基带通信 IP 核的设计与系统仿真[J]. 计算机工程,2009,35(15):275-276,279

[21] 龙昭华,漆动波. 一种嵌入式系统自适应调度算法研究[J]. 重庆邮电大学学报,2009,21(5):654-657

[22] Ma Lian-bo, Hu Kun-yuan, Zhu Yun-long, et al. A Novel Hybrid Artificial Bee Colony Optimizer[J]. Applied Mathematics and Computation, 2015, 252:133-154

[23] Xiao Z H, Guan Z Q, Zheng Z H. The research and development of the highway's electronic toll collection system[J]. International Workshop on Knowledge Discovery and Data Mining, 2008:359-362

(上接第 221 页)

于发生在 VM2 中的接收操作产生的性能开销较小,PAMM 带来的影响也相对较小。

结束语 Xen 基于内存共享的域间通信可以提高运行在相同物理机上的虚拟机间的通信效率,但是内存共享过程中的根-非根状态切换影响了它的通信优化能力,当通信数据量较大时,大量的状态切换导致了通信性能的下降。为了优化基于内存共享的域间通信,本文通过对内存共享过程的分析,得到了使用内存共享进行数据通信时对通信性能产生影响的因素,引入内存页聚合管理模型 PAMM,对通信进行了性能优化。PAMM 中包含一个页面管理模块,通过聚合相邻的传递内存页较少的页面传递,减少通信中的超级调用申请次数,优化 CPU 的使用效率。实验结果证明,当通信数据达到一定输入量后,PAMM 能够较好地优化通信性能。但是,通过实验发现,管理模块的参数设定变化会对 PAMM 的性能产生影响。

基于本文的工作,未来的研究主要集中在参数设定对内存页聚合管理模型性能的影响以及探索更有效的优化方案。

参 考 文 献

[1] Zhang X, McIntosh S, Rohatgi P, et al. Xensocket: A high-throughput interdomain transport for virtualmachines [C]// Procee- dings of the ACM/IFIP/USENIX 2007 International Conference on Middleware. New York, USA: Springer-Verlag New York, Inc. 2007:184-203

[2] Huang W, Koop M, Gao Q, et al. Virtual machine aware communication libraries for high performance computing[C]// Proceedings of the 2007 ACM/IEEE conference on Sup ercomputing. New York, NY, USA: ACM, 2007

[3] Wang J, Wright K, Gopalan K. Xenloop: A transparent high performance inter-VM network loopback [J]. Cluster Computing, 2009, 12(2):141-152

[4] Bughzala B, Ben Ali R, Lemay M, et al. OpenFlow suppo rting inter-domain virtual machine migration[C]// 2011 Eighth International Conference on Wireless and Optical Communications Networks(WOCN). IEEE, 2011:1-7

[5] Llopis P, Blas J, Isaila F, et al. VIDAS: object-based virtua lized data sharing for high performance storage I/O[C]// Proceedings of the 4th ACM workshop on Scientific cloud c omputing. ACM, 2013:37-44

[6] Ning F, Weng C, Luo Y. Virtualization I/O optimization based on shared memory[C]// 2013 IEEE International Conference on Big Data. IEEE, 2013:70-77

[7] Zhang Q, Liu L, Ren Y, et al. Residency Aware Inter-VM Communication in Virtualized Cloud: Performance Measu rement and Analysis[C]// 2013 IEEE Sixth International Conference on Cloud Computing(CLOUD). IEEE, 2013:204-211

[8] 怀进鹏,李沁,胡春明. 基于虚拟机的虚拟计算环境研究与设计 [J]. 软件学报,2007,18(8):2016-2026

[9] Shi Lin, Chen Hao, Sun Jian-hua, et al. vCUDA: GPU-Accelerated High-Performance Computing in Virtua l Machines[J]. IEEE Transactions on Computers, 2012, 61(6):804-816

[10] Dong Y, Yang X, Li J, et al. High performance network virtualization with SR-IOV[J]. Journal of Parallel and Dist ributed Computing, 2012, 72(11):1471-1480

[11] 石磊,邹德清,金海. Xen 虚拟化技术[M]. 华中科技大学出版社

[12] Barham P, Dragovic B, Fraser K, et al. Xen and the art of vir tualization[C]// Proceedings of nineteenth ACM Symposiu m on Operating Systems Principles(SOSP19). 2003:164-177

[13] Bourguiba M, Haddadou K, Korbi I E, et al. Improving network I/O virtualization for cloud computing [J]. IEEE Transactions on Parallel aDd distribut Ed Systems, 2014, 25(3):673-681

[14] Netperf[EB/OL]. http://www.netperf.org/netperf