doi:10.3969/j.issn.1000-436x.2012.z1.020

基于内存分块相异数据的虚拟机同步机制

廖剑伟，陈善雄，李莉
(西南大学 计算机与信息科学学院，重庆 400715)

摘 要：定量分析了不同应用程序的内存分块数据相异部分，即某一阶段的内存有效改动页面分块与其他未变化内存分块的数据相异数据比例，提出基于内存数据相异部分的虚拟机同步机制，主虚拟机端通过基于地址和内容的散列函数表寻找与 Dirty 内存页面分块的最优匹配 Non-dirty 页面分块，相异数据通过 XOR 压缩后通过网络发送给备份虚拟机；备份虚拟机解码接收到的同步数据，重组在主虚拟机端的 Dirty 内存页面，从而完成备份虚拟机的状态同步操作。实验结果表明，与传统的标准异步方式相比，基于内存分块相异数据的虚拟机同步机制可以减少 80%左右的同步操作带来的网络通信数据量，大大提高了某些基准测试程序的系统性能。

关键词：检查点容错技术；虚拟机同步操作；内存分块相异数据；地址内容散列表
中图分类号：TP301 **文献标识码**：A **文章编号**：1000-436X(2012)Z1-0157-08

Memory contents patch based virtual machine synchronization

LIAO Jian-wei, CHEN Shan-xiong, LI Li
(College of Computer and Information Science and Technology, Southwest University of China, Chongqing 400715, China)

Abstract: A quantitative analysis has been conducted on memory contents difference for various high availability benchmarks. Based on this study memory contents patch based backup virtual machine (VM) synchronization technique was proposed, which finds the best match memory section for each dirty memory section; then compresses the difference contents in these two sections using hash based XOR compression technique in the primary VM; finally, it sends the packed data to the backup VM. For the backup VM part, it first decodes the received data and re-constructs the dirty pages, and then applies these pages to complete the VM synchronization. The experimental results show that compared with regular asynchronous replication, the proposed mechanism can reduce the synchronization data and network traffic by up to 80%, and then benefit to the synchronization to a great extent.

Key words: checkpoint-based fault tolerance; virtual machine synchronization; memory contents patch; address and content hash table

1 引言

随着云计算技术的发展，虚拟化技术推动着整合计算机硬件和软件系统并向外界提供弹性、可扩展性和可靠性的虚拟化服务[1]。虚拟机（VM, virtual machine）复制技术是云计算环境下提高系统可靠性的重要机制，一旦主虚拟机所属的物理机器发生崩溃或者主虚拟机本身发生崩溃而无法继续提供服务时，位于云计算机环境中的其他物理机器上的备份虚拟机会接替已崩溃的主虚拟机重新向外界提供服务，从

收稿日期：2012-08-06
基金项目：西南大学博士基金资助项目（SWU112025）；重庆市自然科学基金资助项目（CSTS2010BB2006）；国家自然科学基金资助项目（61170192）
Foundation Items: The Scientific Research Fund for Doctor of Southwest University of China (SWU112025); The Natural Science Foundation of Chongqing (CSTS2010BB2006); The National Natural Science Foundation of China (61170192)

而提高服务的可用性和系统的可靠性[2,3,17]。

在主虚拟机正常工作状态下，需要同步主虚拟机和备份虚拟机状态，通常通过检查点技术或者其他技术将主虚拟机的虚拟磁盘和内存状态定期的保存并通过网络发送至备份虚拟机帮助其更新虚拟机状态，使备份虚拟机与主虚拟机最近的检查点时状态一致[2,4,5]。然而系统性能是虚拟机同步操作的最大挑战，通常虚拟机可能占据几 GB 的内存，因此同步虚拟机状态是一个相当重量级的操作[2,5]。为了减少虚拟机同步操作时主虚拟机的停止运行时间、内存复制和网络传送的数据量，诸如预复制（pre-copy）[6]、I/O deduplication[7]和写时复制（copy-on-write）[8]等优化技术相继被提出，尽管这些技术从一定程度上减少了虚拟机同步操作带来的性能代价，但是没有从根本上解决需要处理并通过网络传送的庞大数据量。以写时复制为例，它需要保存自上一同步点以来的发生变化的所有页面，即使某页面仅仅改变了一个字节，也需要保存并发送该完整的内存页面[8]。

本文提出了一种基于内存分块相异数据的虚拟机同步技术。在进行同步操作时，主虚拟机通过基于内容和地址的二维散列表逐一寻找与发生变化的页面（dirty pages）分块最优匹配的未变化页面（non-dirty pages）分块，同时找出页面分块之间的数据相异部分，即补丁；最后通过 XOR 压缩方法对相异数据进行压缩编码并传送给备份虚拟机。备份虚拟机在接收到数据后进行解码并重构主虚拟机端的发生变化的内存页面，并更新其虚拟机状态。同时，为了减少主虚拟机的停止运行时间，在进行 XOR 数据压缩时，主虚拟机在写时复制模式下保持运行状态。不同于传统的异步同步方式，基于内存分块相异数据的虚拟机同步机制仅需处理和保存发生变化的页面分块与对应的未发生变化的最优匹配页面分块的相异数据和描述相异数据位移信息，大大减少了主虚拟机端需要处理、保存和通过网络传送的数据量，从而提高主虚拟机在同步操作时的系统性能。

2 内存分块的数据相异性

定量分析了不同基准测试程序的内存数据信息，说明在应用程序的内存地址空间中不同页面分块的相异数据较小，即通过实验分析在同一应用程序的内存地址空间内，属于不同页面的分块之间相异数据字节数占分块大小的百分比。选取了 Linux Kernel Compile、SPECweb 2005 Banking，SPECweb 2005 E-Commerce 和 Exchange Load Generator 共 4 种不同的高可靠性基准测试程序[9,10]来分析内存页面分块的数据相异性，其中前 3 种基准测试程序运行在 Debian Linux 虚拟机环境，Exchange Load Generator 运行在 Windows Server 2003 虚拟机环境。

1) Linux Kernel Compile：使用默认配置，编译 Linux Kernel Version 2.6.31。

2) SPECweb 2005 Banking：模拟网上银行，多用户同时访问帐号，事务处理等操作，采用基于 SSL 协议进行数据传输[11]。

3) SPECweb 2005 E-commerce：模拟网上商店，多用户同时浏览、选择和购买商品等，采用了基于 SSL 协议和普通 HTTP 协议进行数据传输[11]。

4) Exchange Load Generator：模拟多用户同时发送和阅读邮件，浏览工作议程等[12]。

在实验中，使用大小为 5 000 内存页面的 RUDP(recently used dirty pages)缓存来保存最近使用过的 Dirty 页面，缓存采用 LRU 策略进行页面替换。通过追踪 RUDP 缓存记录基准测试程序的 Non-Zero Dirty 内存页面。同时，将这些 Dirty 内存页面分割成固定大小的页面分块，利用 3.1 节提及的二维散列表查找相异数据最少的 Non-Zero Non-Dirty 内存页面中的分块，称之为最优匹配页面分块。

图 1 显示在不同大小的页面分块的条件下，对应于 Non-Zero Dirty 页面分块的最优匹配的 Non-Zero Non-Dirty 页面分块的数据相异性。从图 1 中可以看出，随着分块逐渐变小，页面分块的数据相异性逐渐变小。以 SPECweb2005 Banking 为例，相比于分块为 4KB，当分块大小为 64B 时(13.4%)，最优匹配页面分块的内容相异度降低超过 13.2%（即 26.6%~13.4%）。但是，页面分块变小直接增加了寻找最优匹配分块的时间，也间接带来额外的整合、保存、发送同步数据的性能代价。通过重复实验比较，将分块大小固定为 512byte 时，可以获取较好的页面分块相异度（即相异数据的大小）与性能代价之间的平衡点。

3 系统设计与实现

基于存储器分块相异数据的虚拟机同步机制

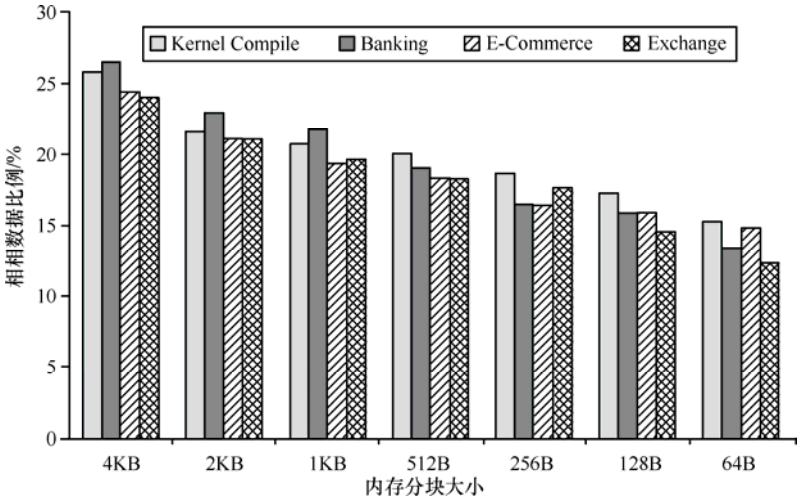


图 1　基准测试程序的内存分块数据相异性

的基本算法如图 2 所示，与传统的直接收集 Dirty 页面并将其从主虚拟机端传送至备份虚拟机端不同，本文算法中的虚拟机同步操作主要分为以下 3 步。

1) 主虚拟机停止运行以确保状态的一致性，找出内存地址空间中所有 Dirty 页面分块的最优匹配页面分块。

2) 将 Dirty 页面分块与最优匹配页面分块之间相异数据（即存储器内容补丁）执行 XOR 操作，附加相异数据的分块位移信息进行编码；将编码后的数据置于 I/O 缓冲区并通过网络传送至备份虚拟机。为了减少主虚拟机的停止时间，一旦编码后的数据置于 I/O 缓冲区，主虚拟机立即重启继续运行。

3) 备份虚拟机接收到主虚拟机发送的数据后，通过解析压缩数据，结合页面分块的位移信息，从而在备份虚拟机中重构在主虚拟机端发生变化的页面；最后将发生变化的页应用于备份虚拟机中完成同步操作。

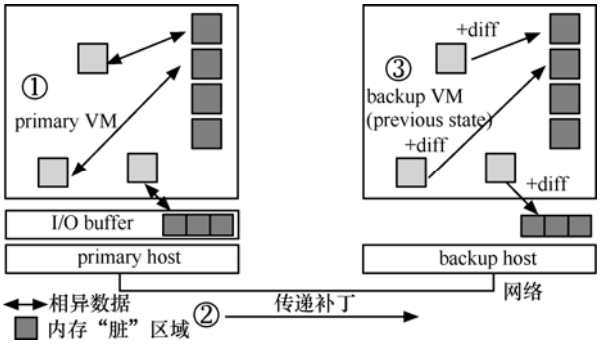


图 2　基于内存分块数据相异性的虚拟机复制技术体系结构

因为 Dirty 页面分块与最优匹配页面分块之间相异数据、分块位移信息和其他辅助信息的数据量要远远小于 Dirty 页面本身的数据量，因此在主虚拟机端处理、保存和通过网络传送的数据量大大减少，从而有效的提高了主虚拟机的系统性能，减少了主虚拟机的停止运行时间。

3.1　查找最优匹配分块

K-clustering[13]和 R-trees[14]等算法可以精确的查找最优匹配内存页面分块，但是这些算法较高的时间复杂度太高，过长的查找时间直接导致主虚拟机的停止服务时间变长，因此该类算法并不适用于虚拟机同步操作。因此本文提出了一种快速、基于内存页面分块内容的查找近似的最优匹配页面分块的算法。

与 Run-Length Encoding (RLE)[15]向量压缩机制类似，本文算法首先将内存页面分块划分为 32 个子区域，当子区域内的比特位为 1 的总位数超过设定的阀值（注：该阀值与页面分块大小和具体的应用程序相关，通过实验发现将该阀值设为子区域位数的 80%较为合适），则该子区域对应的散列值为 1，否则为 0。最终，对每个内存页面分块，都有一个相应的 32 位的内存页面分块的内容散列值。在查找最优匹配页面分块时，认为具有一样或者最为接近的内容散列值的页面分块具有最优匹配性。

为了让内存页面分块内容与地址相对应，提出了包括地址和内容的二维散列函数查找最优匹配内存页面分块。图 3 显示了二维散列表的构成，每一个 None-zero 内存页面分块由一个内存描述符(memory descriptor)表示，该描述符包括内存页面分块的地址散列值以及相对应的页面分块的内容散列值。当不同的内存描述符具有相同的内容

散列值时，说明这些内存页面分块极有可能是最优匹配页面分块。在主虚拟机收集同步数据时，一旦某页面分块被确定为 Dirty 页面分块，则其相对应的内存分块描述符会从二维散列表中删除，以确保存其他 Dirty 页面分块不会寻找它作为最优匹配页面分块。

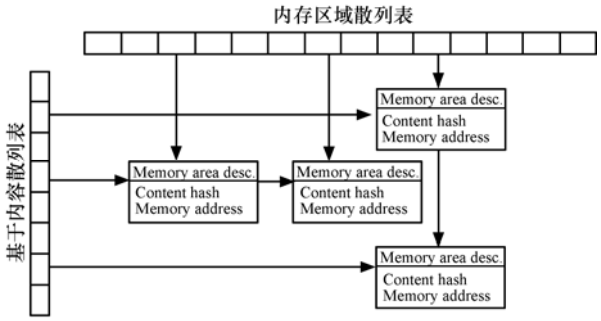


图 3　基于地址和内容的二维散列表

3.2　XOR 数据压缩

通过基于内容和地址的散列表查找到最优匹配内存页面分块，为了减少置于 I/O 缓冲区和通过网络传送的数据量，提出了基于 XOR 的数据压缩机制。如图 4 所示，找出 dirty 页面分块与最优匹配页面分块的数据相异比特位，执行 XOR 操作形成相异页面分块(diff)；最后将相异页面分块的有效比特位作为同步消息的一部分。其中同步消息中的 bitmask 记录相异比特位的分块位移信息，帮助备份虚拟机重构在主虚拟机端发生变化内存页面。

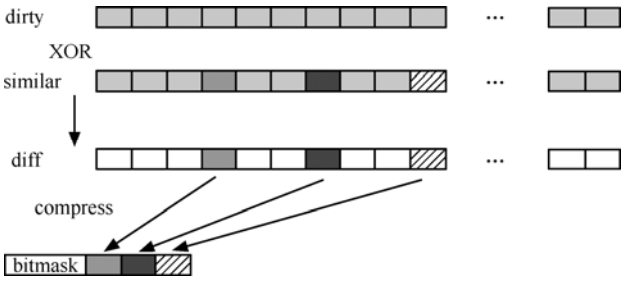


图 4　基于 XOR 的数据压缩

3.3　系统实现

基于 Linux KVM 实现了本文提出的基于内存分块相异数据的虚拟机同步机制。图 5 显示了 Linux KVM 的体系结构，这里不再介绍各模块的具体功能。为了使 Linux KVM 各模块支持虚拟机复制和同步功能，除利用开源的 BLCR 0.80[8]作为检查点模块外，还修改了 Linux KVM 的功能模块，约 4 000 行的源代码。

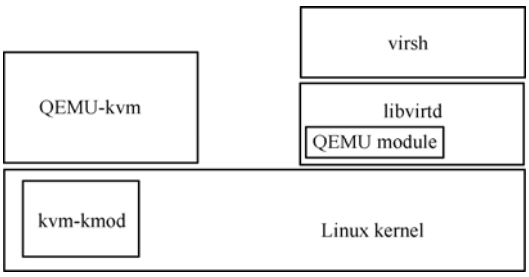


图 5　Linux KVM 体系结构

3.3.1　写时复制（copy-on-write）

写时复制技术是一项在操作系统中的常用的优化技术，为了减少主虚拟机在同步操作时的停止运行时间，在进行 XOR 压缩分块相异数据时，主虚拟机在写时复制模式下运行。与传统意义上的写时复制技术不同，在基于内存分块数据相异性的虚拟机同步操作中，由于 XOR 压缩线程与虚拟机本身共享相同的内存地址空间，一旦虚拟机需要向页面写操作时，新的页面会被通过复制方式创建，虚拟机便向新页面进行写操作。然而，XOR 压缩线程关心的是写入数据之前的页面。为了解决这个问题，本文算法在写时复制之前将原始页面映射到另一虚拟地址，并将映射信息记录在临时转换表中供 XOR 压缩线程访问和查询。因此，在 KVM 的最底层，为了支持 qemu-kvm 模块中 XOR 线程对原始页面的访问，对 KVM 内核模块的写时复制的功能模块进行了修改和一定程度的扩展。

3.3.2　查找、压缩和 I/O 缓冲

虚拟机的同步操作相关的各个功能模块，包括寻找最优内存页面分块和 XOR 数据压缩模埠的实现等都在 qemu-kvm 模块中。另外，通过修改 qemu-kvm 模块中的 virtio 驱动以支持虚拟机同步操作时的磁盘 I/O 和网络缓冲机制，磁盘 I/O 缓冲区利用散列表加快数据的读写操作；网络缓冲主要是在暂存发送的同步信息数据分组，当备份虚拟机准备好接收同步信息数据分组后，处于缓冲区的所有同步数据分组才会通过网络进行传送。

4　实验

为了评价基于内存分块相异数据的虚拟机同步技术，通过各种实验与传统的虚拟机复制技术（即虚拟机同步技术）进行了比较。因为虚拟机同步操作时，备份虚拟机一直处于一种待机状态，所以实验仅测试和记录主虚拟机端进行同步操作时带来的系统性能代价。本节首先介绍实验平台，然

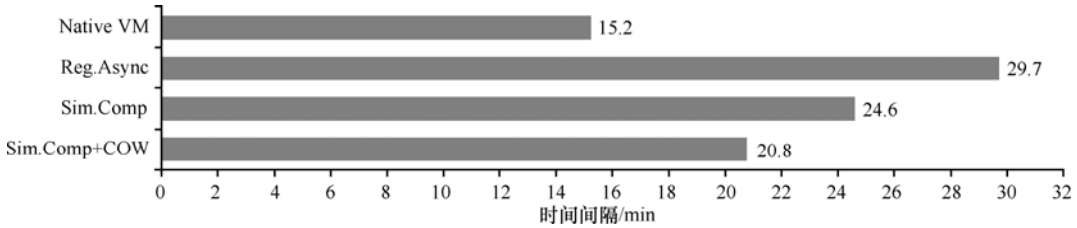


图 6　编译内核时间

后根据基准测试程序类型分别介绍相应的实验结果并进行分析和讨论。

4.1　实验平台

使用了 3 台服务器运行虚拟机，具体性能参数如表 1 所示；在服务器上运行操作系统为 Debian Linux 和 windows server 2003 的虚拟机，均采用 virtio 磁盘和网络驱动程序。同时，使用 2 台 Windows XP 和 Debian Linux 的台试机分别运行 Exchange Load Generator 和 SPECweb 客户端脚本。

表 1　　服务器性能参数

硬件环境	
CPU	Intel Xeon L5520 (2.26GHz, 4 cores) Hyper-Threading,Turbo Boost and Extended Page Tables(EPT) enabled
内存	3GB
网络	NetXtreme II BCM5716 x 2
软件环境	
操作系统	Debian Linux with kernel 2.6.31
虚拟机配置	qemu-kvm 0.12.3 with kvm-kmod 2.6.31.6b

4.2　实验结果-内核编译

通过编译默认设置的 Linux 内核 2.6.31 为 bzImage 文件，分别记录采用原始方式（Native VM）、标准异步同步方式（Reg Async）[3]、本文算法（Sim Comp）、写时复制机制的本文算法（Sim comp+COW）等不同虚拟机运行方式时所需的运行时间。

其中，原始方式指直接在未修改的 Linux KVM 上执行内核编译所需要的运行时间；标准异步虚拟机方式指主虚拟机停止运行以确保状态的一致性，然后收集所有的 Dirty 页面并将其置于 I/O 缓冲区后，主虚拟机立即重启运行，即通过网络传送 Dirty 页面数据操作与主虚拟机的运行是并发完成；本文算法是指在主虚拟机停止运行后，通过寻找最优匹配页面分块、XOR 压缩后并置于 I/O 缓冲区，随即主虚拟机重启运行，最后压缩并编码的同步数据通过网络传送至备份虚拟机；写时复制的本文算法是指在 XOR 数据压缩时，主虚拟机运行于写时复制模式，因此主虚拟机的停止时间相对于本文算法的普通模式会有所减少。

图 6 显示了采用不同方式编译 Linux 内核所需的时间。原始方式不需要产生和处理同步数据并进行传送，所以它需要的运行时间最少；然而由于标准异步方式需要收集所有的 Dirty 页面并将其置于 I/O 缓冲区，所以它需要的运行时间最长；本文算法以及写时复制的本文算法所需要收集、处理和发送的数据量要远少于标准异步方式，因此分别约减少约 17.2%和 30.0%的运行时间。

同时分析了当基准测试程序为内核编译时，分别采用标准异步方式与本文算法进行虚拟机同步操作需要处理的同步消息数据量。从图 7 可知，本文算法收集和处理并通过网络传送的数据量要远小于标准异步方式，仅为标准异步方式约 30%的数据量。

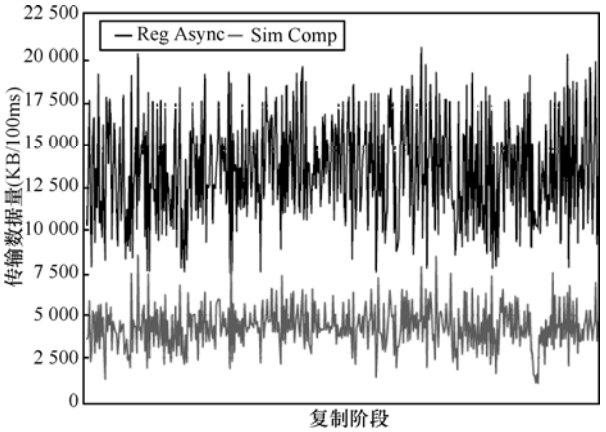


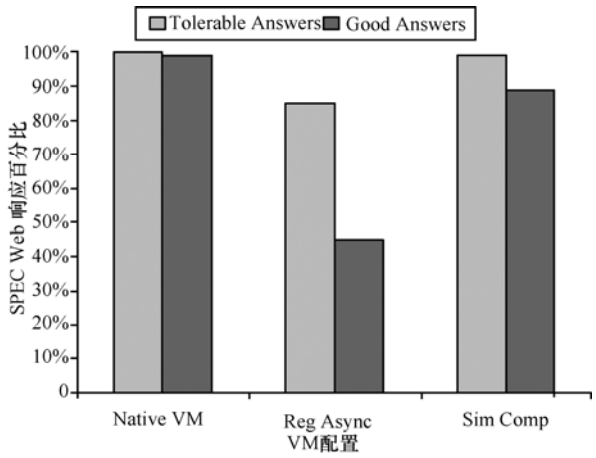
图 7　编译内核的数据传输量

4.3　实验结果-SPECweb 2005-Banking&E-Commerce

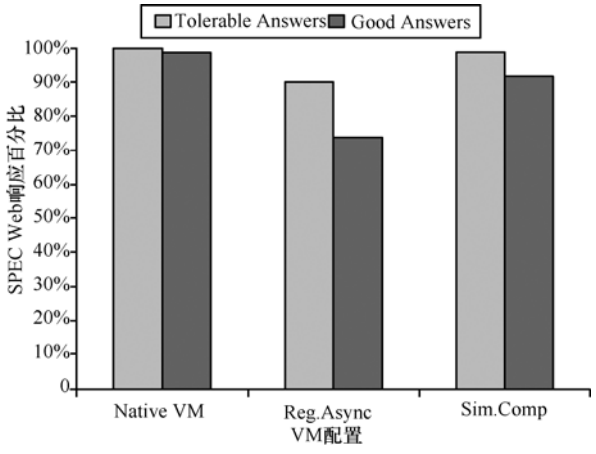
运行 SPECweb 基准测试程序需要 3 台机器，一台机器运行实际的 SPECweb 基准测试程序，另一台作为后备机器，在实验中使用了 2 台虚拟机来模拟实际机器，即一台虚拟机为主虚拟机，另一台为备份虚拟机；第 3 台台式机被用来运行客户端脚本。

实验首先通过调整设置，使得在原始虚拟机中运行得到 99%的“good answer”（该数据是由运行客户端脚本的机器端报告）。在标准异步方式和本文算法进行同步操作时都采用同样的调整设置；由于 SPECweb 对于网络延迟敏感的基准测试程序，所以设置主虚拟机与备份虚拟机的同步操作的时间间隔为 50ms。

图 8 显示了分别采用 2 种不同的同步机制时行虚拟机同步操作时，主虚拟机端带来的系统性能代价。其中，图 8(a)显示了在 Banking 子基准测试程序时客户端的响应报告。在标准异步方式中，仅仅约 45%的“good answers”，本文算法可以带来约 88%的“good answers”；另一方面，在“tolerable answers”指标中，本文提出的算法也要高于标准异步方式。图 8(b)显示当子基准测试程序为 E-Commerce 的客户端响应报告，同样本文明显优于标准异步方式。



(a)SPEC Web banking 负载



(b) SPEC Web E-Commerce 负载

图 8　SPECweb 2005 Banking & E-Commerce 数据

同时分析了当基准测试程序为 SPECweb2005-Banking 时，采用标准异步方式与本文算法进行虚拟机同步操作需要处理的同步消息数据量。从图 9 可知，本文算法收集和处理并通过网络传送的数据量要远小于标准异步方式，是它约 20%的数据量。

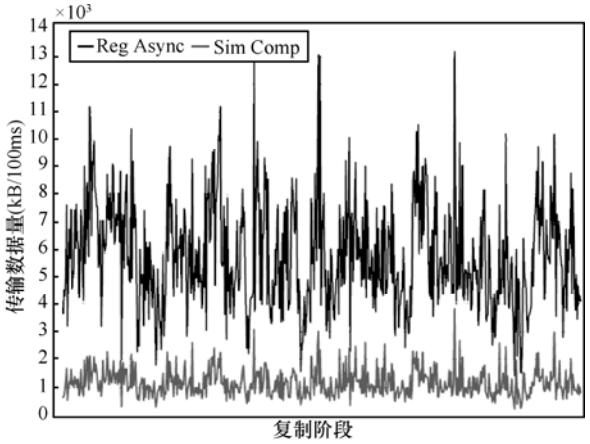


图 9　SPECweb 2005-Banking 的数据压缩比

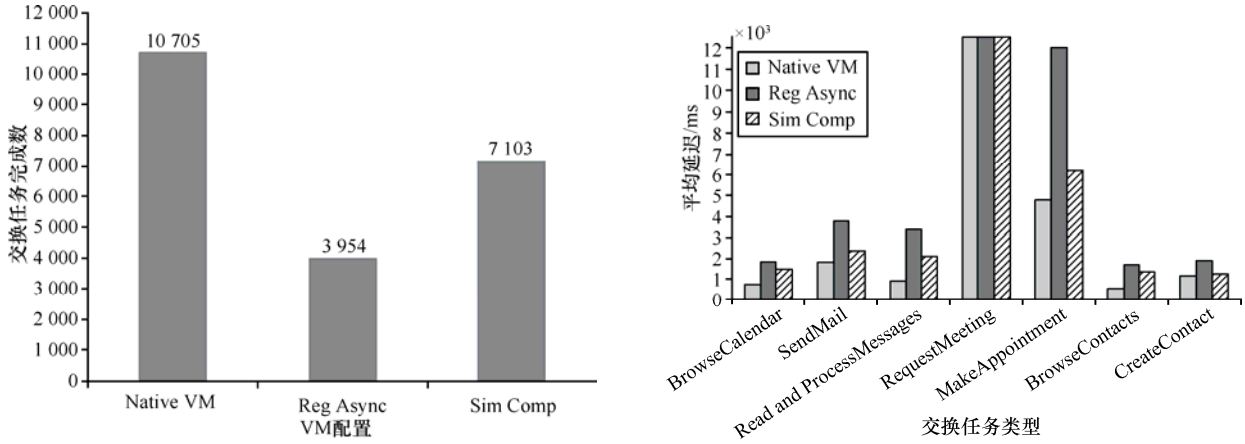
4.4　实验结果-Microsoft Exchange Server

本节的实验均在 Windows 环境下完成，在 Windows2003 Server 虚拟机下部署 Microsoft Exchange Server 2007。Exchange Server 软件向外界客户端提供 e-mail，日志，通讯录等系统服务。同时，在运行 Windows XP 的台试机中运行 Windows Exchange Load Generator 应用程序模拟多用户同时请求 Exchange Server 服务。需要指出实验中仅仅服务器端是使用备份虚拟机以提供系统容错，虚拟机同步操作的时间间隔为 50ms；本文算法支持写时复制功能。

图 10(a)显示了使用不同的虚拟机同步机制时，10 分钟内 Exchange Server 完成的总任务数。与在原始虚拟机方式下完成超过 10 000 任务数相比，采用标准异步方式进行虚拟机同步操作环境下的完成任务数下降为 3 954，而采用本文算法进行虚拟机同步操作环境下的完成任务数达到了 7 100。性能提高近 80%。图 10(b)显示了不同子任务完成所需的时间延迟，因为 RequestMeeting 子任务的时间延迟大大超出其他子任务，所以完成该子任务时间延迟并没有完成显示出来，同样，本文算法要明显优于标准异步方式。

5　结束语

定量分析不同应用程序的内存地址空间的不同



(a)平均任务完数(Stree-mode, 10min)　　(b)任务延迟细节

图 10　Microsoft Exchange Server 数据

页面分块数据相异性。通过实验观察，发现内存页面分块与前一阶段的某些页面分块（即从前一阶段开始，该内存页面数据未发生变化）的数据相异性较小。当内存页面分块为 512byte 时，内存分块的相异性为 20%左右。基于该发现，本文提出了一种基于内存分块数据相异性的虚拟机同步技术，该方法在主虚拟机端通过基于地址和内容的散列函数表寻找与 Dirty 内存页面分块的最优匹配 Non-dirty 内存页面分块，将两个内存分块的相异数据通过 XOR 压缩技术进行压缩后加上分块的位移信息整合进行编码，通过网络传递给备份虚拟机。备份虚拟机对接收到数据进行解码，重组在主虚拟机端的 Dirty 内存页面，从而完成备份虚拟机的同步操作。

与传统的标准异步虚拟机复制技术直接收集所有 Dirty 内存页面相比，通过基于内容与地址的散列表技术查找指定内存分块的最优匹配内存分块可以减少近 30%的性能代价。另外，基于 XOR 的压缩方法可以减少 80%的网络通信量，对于部分基准测试程序而言，本文算法可提高性能高达 90%，适用于对延迟敏感的应用程序的主虚拟机与备份虚拟机之间的同步操作。不过，需要指出本文提出的算法需要在主虚拟机端查找最优匹配内存分块和进行数据压缩，当虚拟机的内存空间为 1GB 时，大约需要额外的 70MB 的内存空间存储散列表和部分中间数据，同时也需要额外 30%CPU 计算机资源进行 XOR 数据压缩。

参考文献：

[1] MORENO-VOZMEDIANO R, MONTERO R, LLORENTE I. Elastic management of cluster-based services in the cloud [A]. Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds[A]. Barcelona, Spain, 2009. 19-24.

[2] VOORSLUYS W, BROBERG J, VENUGOPAL S, *et al*. Cost of virtual machine live migration in clouds: a performance evaluation[A]. Proceedings of the 1st International Conference on Cloud Computing[C]. Beijing, China, 2009. 254-265.

[3] CULLY B, LEFEBVRE G, MEYER D, *et al*. Remus: high availability via asynchronous virtual machine replication[A]. Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation[C]. San Francisco, USA, 2008. 161-174.

[4] SAPUNTZAKIS C P, CHANDRA R, PFAFF B, *et al*. Optimizing the migration of virtual computers[J]. ACM SIGOPS Operating Systems Review, 2002, 36(SI): 377-390.

[5] ZHU J, DONG W, JIANG Z, *et al*. Improving the performance of hypervisor-based fault tolerance[A]. Proceedings of International Parallel and Distributed Processing Symposium[C]. Georgia, USA, 2010. 1-10.

[6] CLARK C, FRASER K, HAND S, *et al*. Live migration of virtual machines[A]. Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation[C]. Berkeley, CA, USA, 2005. 273-286.

[7] KOLLER R, RANGASWAMI R. I/O deduplication: utilizing content similarity to improve I/O performance [J]. ACM Transaction on Storage, 6(13): 1-26, 2010.

[8] VRABLE M, MA J, CHEN J, *et al*. Scalability, fidelity, and containment in the potemkin virtual honeyfarm[J]. SIGOPS Oper Syst Rev, 200539(5): 148-162.

[9] LAGAR-CAVILLA H A, TOLIA N, LARA E, *et al*. Interactive resource-intensive applications made easy[A]. Proceedings of 8th International Middleware Conference[C]. Grenoble, France, 2005. 143-163.

[10] LU M, CKER CHIUEH T. Fast memory state synchronization for

virtualization-based fault tolerance[A]. Proceedings of IEEE/IFIP International Conference on Dependable Systems Networks[C]. Lisbon, Portugal, 2009. 534-543.

[11] HARIHARAN R, SUN N. Workload characterization of SPECweb2005[EB/OL].http://www.spec.org/workshops/2006/papers/02_ Workload_char_SPECweb2005_Final.pdf, 2012.

[12] Microsoft Corporation. Microsoft exchange load generator [EB/OL]. http://www.msexchange.org/articles/Microsoft-Exchange-Load-Generator.html, 2012.

[13] GUTTMAN A. R-trees: a dynamic index structure for spatial searching[A]. Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD ’84)[C]. Massachusetts, USA, 1984. 47-57.

[14] HARTIGAN J A. Clustering Algorithms[M]. New York: John Wiley &Sons, Inc, 1975.

[15] ZHANG X, HUO Z, MA J, *et al*. Exploiting data deduplication to accelerate live virtual machine migration[A]. proceedings of 2010 IEEE International Conference on Cluster Computing (CLUSTER ‘10)[C]. Crete, Greece, 2010. 88-96.

[16] 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.
HUAI J P, LI Q, HU C M, Research and design on hypervisor based virtual computing environment[J]. Journal of Software, 2007,18(8): 2016-2026.

[17] 孙昱. 虚拟机 Xen 及其实时迁移技术研究[D]. 上海:上海交通大学, 2008.
SUN Y. A study of Xen and Live Migration[D]. Shanghai: ShanghaiJiaotong University, 2008.

作者简介：

廖剑伟（1981-），男，湖南怀化人，博士，西南大学副教授，主要研究方向为计算机容错技术和可靠性操作系统。

陈善雄（1981-），男，重庆渝北人，西南大学讲师，主要研究方向为分布式系统、自组织网络和模式识别。

李莉（1967-），女，重庆北碚人，博士，西南大学教授、博士生导师，主要研究方向为分布式计算与智能信息处理、自组织网络。