研究与开发

基于 KVM 的虚拟机 Post-Copy 动态迁移算法稳定性优化

陈双喜[1,2,3]，赵若琰[1]，刘会[2]，吴春明[1]，阮伟[1]
（1. 浙江大学，浙江 杭州 310058；2. 嘉兴职业技术学院，浙江 嘉兴 314036；
3. 嘉兴市工业互联网安全重点实验室，浙江 嘉兴 314036）

摘　要：虚拟机动态迁移技术为虚拟化系统的资源调度提供了强有力的支撑，Post-Copy 算法作为虚拟机动态迁移的两个核心算法之一，凭借其总体迁移时间稳定与迁移停机时间短的优点，一直是国内外学者研究的热点问题。对虚拟机的故障容错机制、迁移过程中的内存页面传输方式与缺页错误的关联性，以及 QEMU-KVM 平台源码进行了深入的研究，提出了基于事务同步的故障容错方法以提升 Post-Copy 迁移算法的稳定性。试验结果表明，提出的虚拟机 Post-Copy 迁移优化算法，能保证迁移过程中源端虚拟机故障、目标端虚拟机故障以及网络故障的迅速修复，能通过较小的代价解决稳定性问题，所提出的方法有效地提升了 Post-Copy 迁移算法的稳定性，也为以后的优化研究方向提供了参考。

关键词：虚拟机动态迁移；Post-Copy；故障容错；QEMU-KVM
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000−0801.2021145

Stability optimization of dynamic migration algorithm for Post-Copy of virtual machine based on KVM

CHEN Shuangxi[1,2,3], ZHAO Ruoyan[1], LIU Hui[2], WU Chunming[1], RUAN Wei[1]
1. Zhejiang University, Hangzhou 310058, China
2. Jiaxing Vocational Technical College, Jiaxing 314036, China
3. Jiaxing Key Laboratory of Industrial Internet, Jiaxing 314036, China

Abstract: The dynamic migration technology of virtual machine provides a strong support for the resource scheduling of virtualization system. As one of the two core algorithms of virtual machine dynamic migration, Post-Copy algorithm has been a hot issue for scholars at home and abroad for its advantages of stable migration time and short migration downtime. The fault tolerance mechanism of virtual machine, the connection between the transfer mode of memory page and the page missing error during the migration process, and the source code of QEMU-KVM platform

收稿日期：2021−01−07；**修回日期**：2021−06−08
通信作者：阮伟，ruanwei@zju.edu.cn
基金项目：国家重点研发计划项目（No.2018YFB2100400）；浙江省公益技术研究计划项目（No.LGF20F020003）；浙江省重点研发计划项目（No.2020C01077,No.2021C01036,No.2020C01021）；之江实验室重大科研项目（No.2018FD0ZX01）
Foundation Items: The National Key Research and Development Program (No.2018YFB2100400), Zhejiang Public Welfare Technology Research Project (No.LGF20F020003), Zhejiang Key Research and Development Program (No.2020C01077, No.2021C01036, No.2020C01021), Zhijiang Laboratory Major Scientific Research Project (No.2018FD0ZX01)

were studied. A fault tolerant method based on transaction synchronization was proposed to improve the stability of the Post-Copy migration algorithm. The experimental results show that the proposed algorithm can ensure the fast repair of deep end virtual machine failure, target virtual machine failure and network fault during the migration process, and solve the stability problem at a small cost. The method proposed effectively improves the stability of the Post-Copy migration algorithm, and provides reference for the future optimization research direction.

Key words: dynamic migration of virtual machine, Post-Copy, fault tolerance, QEMU-KVM

1 引言

云计算的思想源于 20 世纪 60 年代，真正进入大众视野是在 2006 年 8 月，由 Amazon 子公司 Amazon Web Service（AWS）推出第一款云计算产品 Elastic Compute Cloud[1]。近年来，随着云计算的蓬勃发展，网络服务功能和占比总量快速增加，云计算技术受到了广泛关注和应用，其中最重要的是虚拟化技术。虚拟化技术是一种在计算机物理资源之上抽象并进行有效管理的技术[2]。目前常见的虚拟化管理工具有 VMware[3]、KVM[4]、Xen[5]、Hyper-V[6]，借助虚拟化技术能提供比原先多台物理主机更有效的计算，同时还能节省硬件成本和降低能源损耗。虚拟机迁移大体可以分为两种，分别为静态迁移（non-live migration）和动态迁移（live migration），本文着重讨论虚拟机的动态迁移。目前，虚拟机动态迁移有两种常用的算法：Pre-Copy（预拷贝）和 Post-Copy（后拷贝）。不同算法有不同的特点，预拷贝迁移算法的停机时间较短，但是其迭代复制的操作，可能会导致迁移过程无法收敛，总体迁移时间过长；后拷贝迁移算法的停机时间短，总体迁移时间稳定，但虚拟机在迁移过程中会出现缺页错误导致虚拟机性能下降，并且整个迁移过程缺乏稳定性[7]。综上所述，虚拟机的动态迁移在云计算领域有特殊的地位，各种动态迁移算法还存在许多亟待解决的问题，因此研究优化动态迁移算法，对解决现有算法存在的问题具有十分重要的意义。

主流动态迁移算法中，预拷贝迁移算法被研究得相对较多。预拷贝迁移算法由 Clark 等[8]在 2005 年首次提出，如今大多数虚拟机管理程序，例如 Xen、KVM 和 VMware，都将预拷贝作为默认的虚拟机迁移算法。预拷贝迁移算法的问题在于，面对写密集型程序，内存不断被写入形成大量脏页，迁移过程中会将同样一个页面传输多次，且有可能整个过程无法收敛，因此大部分的研究着重解决这个问题。后拷贝迁移算法由于其实现相对复杂，算法中存在的问题比较突出，可优化的点较多，近年来成为了国内外研究的重点。针对后拷贝迁移算法存在的问题，研究主要分为两个方向：减少缺页错误和容错机制。为了减少后拷贝迁移中缺页错误产生的次数，Chashoo 等[9]研究了将预拷贝的思想引入后拷贝中，提出了一种使用内存膨胀技术，内存膨胀技术可以在需要时减少空闲页面和未使用页面从而减少虚拟机内存页面的数量，使得源端虚拟机需要传输到目标端虚拟机的页面数减少，降低内存利用率。Shan 等[10]在 Xen 上设计并实现了远程页表助手（remote page table assistant）的功能，当发生缺页错误时，Xen 通过该功能直接获取当前虚拟机是否正在迁移的信息，当虚拟机正常运行时，Xen 忽略虚拟机监视程序的判断过程，并快速调用预设的缺页处理函数处理缺页错误。Su 等[11]提出了一种远程缺页错误过滤器（RPFF），在后拷贝迁移过程中，通过将客户程序的内存页面重写操作请求重定向到一块新分配的本地内存中，从而消除了不必要的缺页错误，避免从源端虚拟机获取页面，减少因为重写造成缺页错误所带来的额外消耗，特别强调针对内存写入密集型的工作负载，其整体性能会有极大提高。Jalaei 等[12]通过降低

后拷贝迁移算法中虚拟 CPU 的执行速度，将目标端虚拟机中的页面处理速度与源端虚拟机的页面传输速度平衡，从而降低缺页错误产生的次数。孙淑娴[13]提出通过预先执行虚拟机程序预测目标端虚拟机将要访问的内存页面，提前传输到目标端虚拟机从而降低缺页错误率，并利用局部性原理动态调整所需传输的页面数量。对于后拷贝迁移的容错机制，现有的研究还比较少，目前的研究方向主要基于检查点，检查点简单来说是一种备份记录的方法，通过检查点记录虚拟机历史的状态。Fernando 等[14]提出了一种反向连续检查点技术，通过设置检查点定期记录虚拟机的增量状态变化信息，当虚拟机出现故障时可以由保存的最新检查点状态进行虚拟机状态的恢复。Chou 等[15]提出一种基于检查点和 FAM（fabric-attached memory）的后拷贝迁移算法，FAM 是一种特殊的可以使内存进行光纤级通信的机制，通过将整个迁移过程的内存空间映射到 FAM，将数据通信简化成内存语义，极大地减少了缺页错误的处理时间，并使用 Linux 开源检查点工具 CRIU 实现了检查点的功能。

可以看出，目前针对后拷贝迁移优化研究相对较少，虽然取得了一定的成果，但是仍然有许多地方值得挖掘，无论解决缺页错误问题还是容错机制都存在很大的提升空间，因此针对该领域的研究是很有挑战且充满意义的工作。本文通过分析后拷贝迁移算法中存在的稳定性问题进行深入研究和分析，提出了基于事件同步的故障容错方法，结合检查点与事件同步技术实现了故障场景检测以及对应场景的故障恢复。

2　问题分析及优化方案

2.1　后拷贝迁移算法基本流程

后拷贝迁移算法的基本流程如图 1 所示。在整个迁移过程中，源端虚拟机和目标端虚拟机主要在两个阶段进行数据交互。

- 第一阶段：在迁移之初，通过暂停源端虚拟机，将必要的 CPU 状态和设备信息传输

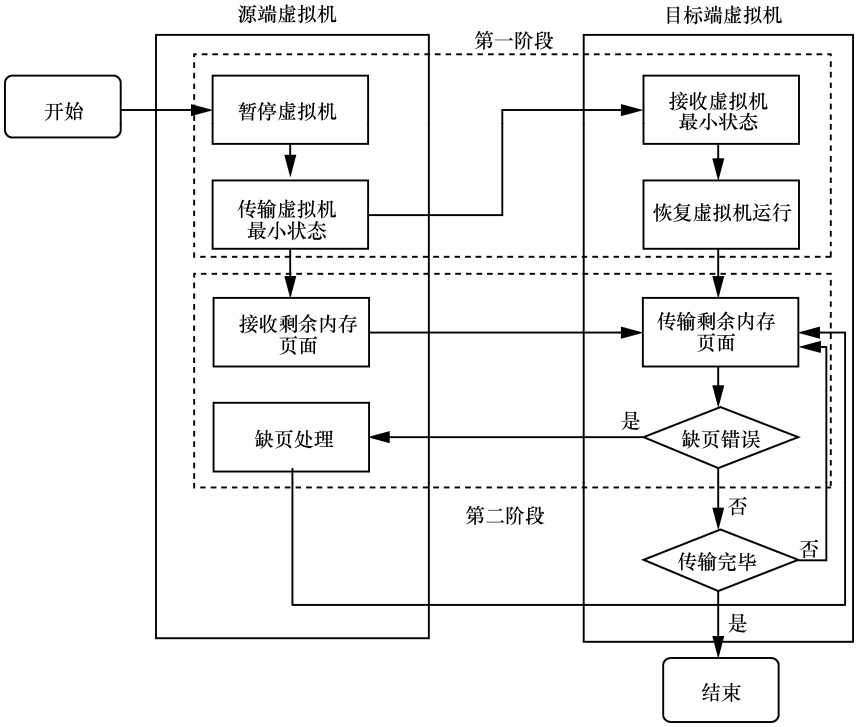

图 1　后拷贝迁移算法的基本流程

到目标端虚拟机，由目标端虚拟机继承并恢复运行被暂停的虚拟机状态。

- 第二阶段：在迁移过程中，由暂停的源端虚拟机主动传输剩余的内存页面到目标端虚拟机。当目标端虚拟机正常运行的过程中出现缺页错误时，将发送请求到源端虚拟机，请求提前获取缺失的内存页面。

2.2 稳定性问题

迁移机制本身的稳定性是虚拟机动态迁移算法优劣评价标准中一项重要的考虑因素。迁移过程中虚拟机的状态分别分布在两台虚拟机上，源端虚拟机拥有内存页面，而目标端虚拟机拥有正在运行的虚拟机上下文信息，总体呈现一种割裂的状态，因此保证两台虚拟机之间数据交互渠道的稳定性至关重要。但是在虚拟机实际运行过程中，很可能频繁受到攻击，特别在迁移过程中受到攻击的影响更大。虚拟机迁移可能持续几秒到几分钟不等，导致漏洞被攻击的窗口可能性很大[14]。因此原始的后拷贝迁移算法中两台虚拟机的交互渠道不能保证绝对的可靠。

后拷贝迁移过程中虚拟机的最新状态分布如图 2 所示。虚拟机的最新状态分别存在于源端虚拟机和目标端虚拟机之上，目标端虚拟机具有最新的虚拟机 CPU 运行状态、设备信息和内存页面的部分子集，而源端虚拟机具有尚未发送到目标端主机的剩余内存面。如果目标端虚拟机出现故障，此时虚拟机的 CPU 状态、设备状态和已经传输到目标端虚拟机的内存页面将丢失，并且由于源端虚拟机上并没有可以支持虚拟机重新运行的相关数据，不仅会导致后拷贝迁移失败，还会造成虚拟机和使用虚拟机服务的用户程序丢失。而如果源端虚拟机出现故障，目标端虚拟机也会因为当发生缺页错误时无法请求源端虚拟机对应的内存页面，一直处于暂停等待的状态，导致整个迁移过程陷入停滞，最终导致迁移失败。

2.3 优化方案

对于稳定性问题，通过分析发现无论是源端虚拟机的故障还是目标端虚拟机的故障，都会导致后拷贝迁移的失败，因此本文对两种虚拟机故障场景提出对应的容错解决策略，具体采用检查点与帧同步相结合的方式，通过记录检查点确定故障恢复的起始状态，利用帧同步对可能发生的不确定性事件进行同步重播。并且为使用更少的虚拟机资源减少额外开销，选择在不增加额外的备份虚拟机基础上进行故障容错。优化方案设计如图 3 所示。

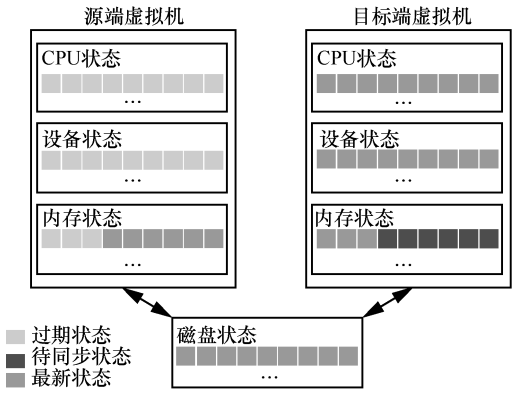

图 2 虚拟机最新状态分布

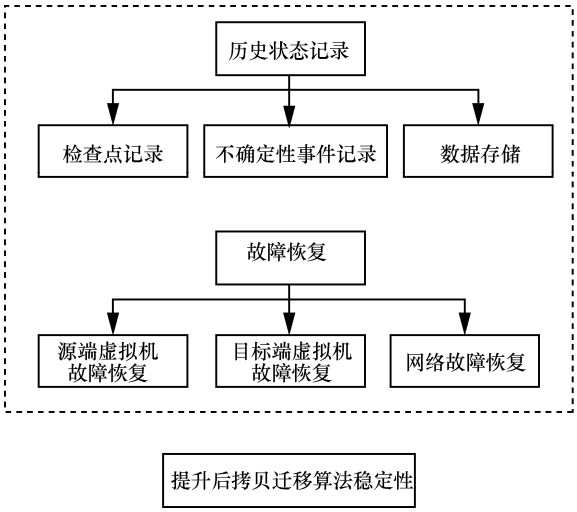

图 3 优化方案设计

3 基于事件同步的故障容错方法

针对后拷贝迁移算法中存在的稳定性问题，提出了基于事件同步的故障容错方法，首先介绍了基本的故障容错架构，其次针对技术细节的实现进行阐述，最后对典型的故障场景进行分析并

提出了对应的故障恢复方案。

3.1　基本架构

基于事件同步的故障容错方法，在 QEMU-KVM 平台基础上进行修改实验，主要目的是为后拷贝迁移提供故障容错机制，保证当源端虚拟机或目标端虚拟机出现故障时，对应的虚拟机状态不会丢失。本文方法实现的关键点主要可以分为不确定事件的记录和同步、增量式检查点的记录、数据的传输与存储、故障检测及修复。

3.2　不确定事件记录和同步

首先需要知道如何正确记录所有的不确定性事件。就整个 QEMU-KVM 平台而言，不确定性事件等同于记录不确定性敏感指令，即会对当前宿主机行为造成影响的指令。当虚拟机需要执行指令时，会触发 VM exit，由非根模式进入根模式进行对应的处理操作。在后拷贝迁移过程中，可能会造成行为发生变化的不确定性敏感指令可分为 4 种类型：中断、PIO（programmed I/O）、MMIO（memory mapped I/O）和缺页错误访存指令。除了需要记录不确定性事件具体的操作指令和相关数据，还需要正确记录指令发生的时间点。记录指令发生的时间点等同于记录一段时间内 CPU 总共执行的指令次数，但是由于指令次数没有特定的 CPU 寄存器保存，无法直接从基本的寄存器中取到，因此需要额外的操作获取执行的指令次数。

3.3　增量式检查点记录

当迁移过程发生故障时，如果从虚拟机的最初状态开始进行恢复，除了需要保存大量的事件信息，还会导致恢复所花费的时间过长。因此本文在目标端虚拟机运行过程中引入检查点机制，检查点与检查点之间定义为一个回合，将整个运行过程划分为多个独立的回合。记录每个回合开始的检查点以及当前回合内发生不确定性事件的相对时间点，当出现故障需要进行恢复时，可以从距离当前错误时间点最近的上一个检查点开始进行具体的恢复流程。

事实上，在虚拟机实际运行过程中，大多数的内存页面是保持不变的，如果只记录产生变化的内存页面，就能有效降低总体的快照消耗。对于内存 1 GB 的虚拟机，CPU 状态和设备状态只占约 600 KB，因此对于占比较少的 CPU 状态和设备状态选择直接进行记录，不做增量的计算，只对内存状态的增量变化进行记录。内存状态的增量变化情况如图 4 所示。

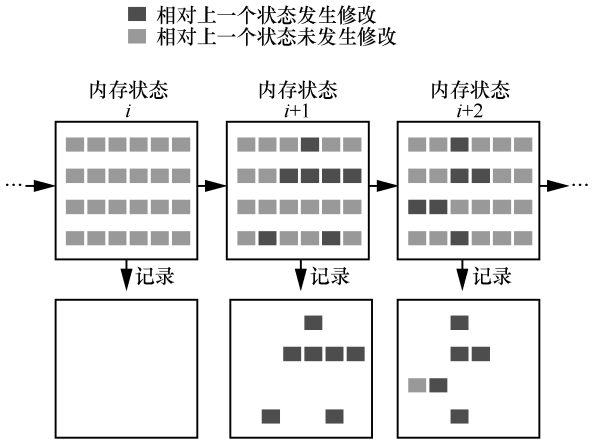

图 4　内存状态的增量变化情况

记录内存状态的增量变化，需要获取当前检查点相对于上一个检查点时的内存页面写脏情况，利用 KVM 内核提供的写保护机制可以完成对脏页情况的追踪。增量式检查点记录的起点位于后拷贝迁移开始前，由源端虚拟机记录完整的内存状态保存到共享磁盘。当迁移开始后，目标端虚拟机接管虚拟机状态并开启脏页同步，通过内核定时器定时记录 CPU、设备状态和内存增量变化，构建相应的检查点数据并进行传输和保存。当故障发生时会从最初的完整内存数据开始，按顺序对每个增量式检查点进行同步，直到同步到最后一个检查点为止。但是顺序同步检查点数据可能带来重复的页面读写，因为内存页面很可能在多个检查点被记录。因此，针对这一问题使用哈希图的方式，保存每一个页面被记录最新的检

查点编号。在进行故障恢复时，可以通过哈希图对每个页面只同步一次，从而减少冗余的内存数据读取，减少故障恢复时间。

3.4 数据的传输与存储

对于记录的不确定性事件信息与检查点信息，需要使用合理的数据格式保存到合适的数据存储工具中，以便在故障恢复时能够及时获取需要的数据。检查点与不确定性事件的传输流程如图 5 所示。具体而言，由 QEMU 通过 kvm_vm_ioctl（KVM_GET_DIRTY_LOG）系统调用设置内核定时器，定时触发 VM exit 使客户程序退出非根模式开始记录检查点数据，QEMU 获取内核数据通过内存映射的方式实现。

相邻检查点之间作为一个回合。每个回合之初，记录 CPU、设备状态以及内存页面增量变化，构建检查点数据，传输检查点数据到 Redis 服务器，并重新设置内核定时器以便进行下一次检查点的记录。每个回合中，当客户程序产生不确定性事件时，记录不确定性事件的信息和发生的时间点，构建不确定性事件数据。为了避免每次发生不确定性事件都进行一次数据传输，通过创建内核事件缓冲区缓存不确定性事件，当事件缓冲区已满或者当前为输出事件时，将缓存的所有事件一并传输到 Redis 服务器。每个回合结束即触发定时器开始下一个检查点回合时，会将事件缓冲区清空。每个回合重复上述流程对检查点数据与不确定性事件数据进行存储，直到迁移过程结束或者迁移出现故障开始对应的恢复流程。

3.5 故障检测及修复

后拷贝迁移过程中可能出现的故障错误包括源端虚拟机故障、目标端虚拟机故障和网络故障。对于源端虚拟机故障恢复，此时目标端虚拟机仍处于运行状态但后拷贝迁移过程被中断，导致部分内存页面仍未传输到目标端虚拟机，当目标端虚拟机再次出现缺页错误时无法请求对应的页面。因此目标端虚拟机需要在产生缺页错误时检

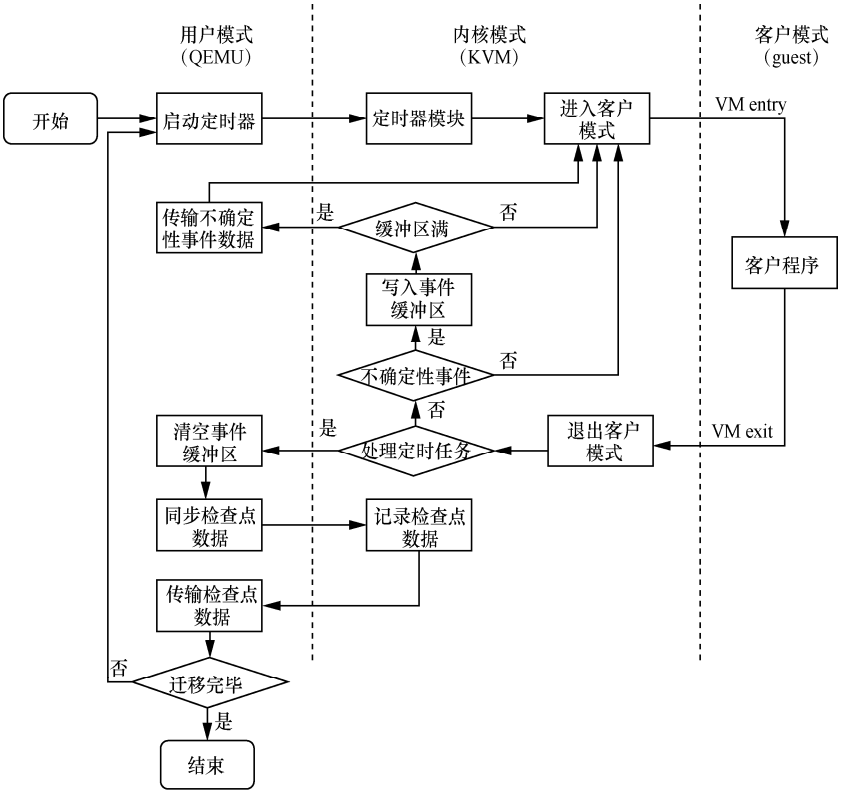

图 5 检查点与不确定事件数据传输流程

测源端虚拟机是否正常，如果检测源端虚拟机故障，那么不再记录不确定性事件和检查点信息，直接从共享磁盘中，读取后拷贝迁移开始时由源端虚拟机保存的最初的内存页面状态，并将仍未传输到目标端虚拟机上的内存页面进行同步。整个过程完成后代表源端虚拟机故障的情况成功恢复，同时也代表后拷贝迁移完成。

对于目标端虚拟机的故障。此时需要由源端虚拟机接管虚拟机的运行状态，相较源端虚拟机故障恢复过程会更复杂。基本步骤是，源端虚拟机读取历史增量式检查点信息，载入检查点中保存的 CPU 状态、设备状态和内存状态到虚拟机之中，载入完毕后开始进行不确定性事件的同步。要进行不确定性事件的同步重播，意味着在正确的指令执行时间点暂停客户机程序插入对应的事件指令，其重点在于如何正确进行事件指令的插入。

对于网络故障的检测与恢复，无论上述两种虚拟机故障还是网络故障的检测，采用的方法都是一致的。对于源端虚拟机和目标端虚拟机，互相使用心跳机制发送 UDP 报文检测对方是否出现故障。

4　实验及评估

4.1　实验环境及设置

本文的实验环境为两台物理主机配合一台交换机所连接的虚拟化平台，其中一台作为源端主机，另一台作为目标端主机，分别于对应主机上创建虚拟机进行迁移实验。此外源端主机还作为 NFS 服务器和 Redis 服务器，提供虚拟机的共享存储服务与故障容错数据的缓存服务，实验网络架构如图 6 所示。

两台实体主机的配置完全一致，分别配备了 Intel(R) Core(TM) i5-10400F 2.90 GHz 处理器，8 GB 内存，RTL8125 吉比特以太网卡，250 GB M2 固态硬盘，以及 Ubuntu-16.04-desktop-amd64 操作系统。两台主机均通过 Bios 开启 Intel-VT 硬

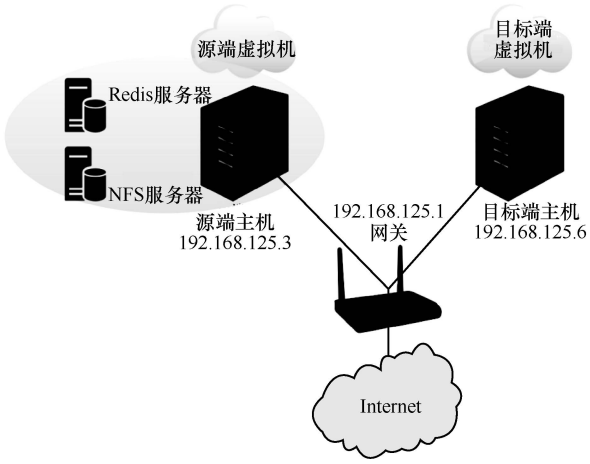

图 6　实验网络架构

件虚拟化服务，并通过吉比特交换机进行数据传输。其中，源端主机作为 NFS 系统服务器，存放修改后的 QEMU-KVM 虚拟机代码与磁盘镜像，并将对应的 NFS 文件夹挂载到另一台主机实现共享存储。宿主机上创建的虚拟机，使用相同的 IMG 磁盘镜像，均配置一个虚拟 CPU。

虚拟机通过桥接网络的形式，获取独立的 IP 地址与外界进行网络通信。测试基于事件同步的故障容错方法。故障容错方法通过定时任务的方式，每间隔一段时间记录检查点和不确定性事件数据。引入故障容错方法必然会对正在运行的客户程序性能造成一定的影响，因此需要通过实验测试并选择合适的数据记录间隔时间，尽可能降低对性能的影响。具体测试的内容有平均每回合记录数据的时间消耗、整个迁移过程中记录数据的时间消耗、平均故障恢复时间以及故障容错方法对客户程序性能的影响。

4.2　实验负载

实验负载分别为 SPECjbb2005、kernel-build、netperf 和 sysbench-oltp，具体的实验负载说明如下。

（1）SPECjbb2005：用于测试工业级 Java 应用性能的基准测试工具，通过修改 SPECCjbb.props 文件设置测试线程的并发数量，通过生成高并发业务操作测试 Java 应用的处理能力。属于

CPU 密集型程序，同时伴随有少量的 I/O 操作。

（2）kernel-build：使用单线程编译 Linux-3.1.0 版本的内核代码，内核编译属于可以测试综合性能的平衡型程序。

（3）netperf：netperf 基于服务器/客户端模式，通过 TCP 和 UDP 传输，对不同网络传输方式的性能进行测试。其中，以 netserver 作为服务端监听客户端请求，netperf 本身作为客户端向服务端发起网络测试。通过配置建立连接，使用不同的网络传输方式传递数据测试网络性能。实验中通过每次连接都重新创建 TCP 连接模拟 HTTP 的访问模式，模拟大量的网络 I/O 操作，属于网络 I/O 密集型程序。

（4）sysbench-oltp：sysbench 是一种多功能的基准测试工具，可以用于测试模拟不同系统参数下的数据库负载情况。实验中使用 OLTP 基准测试，测试 MySQL 数据库的联机事物处理性能，模拟大量访问数据库的磁盘 I/O 操作，属于磁盘 I/O 密集型程序。

4.3 实验结果及分析

不同负载下平均每回合数据记录时间消耗如图 7 所示。平均每回合数据记录所需的时间与数据记录的间隔时间长度成正比。主要原因是，数据记录的时间取决于记录间隔时间内产生的脏页和不确定性事件数量，因此数据量越多，记录所需要花费的时间越长。但是间隔时间短反而会导致记录的回合数更多，单从平均每回合的数据记录时间来看，并不能决定最合适的数据记录间隔时间，因此还需要对比整个迁移过程中记录数据的时间消耗。

不同负载下整个迁移过程中记录数据的时间消耗如图 8 所示。由图 8 中可以看出，随着数据记录间隔时间的增长，数据记录所消耗的总时间呈下降趋势。这是因为，数据记录间隔的时间越短，同步脏页位图与不确定性事件的频率越高，导致整个迁移过程中需要记录的数据量越多，

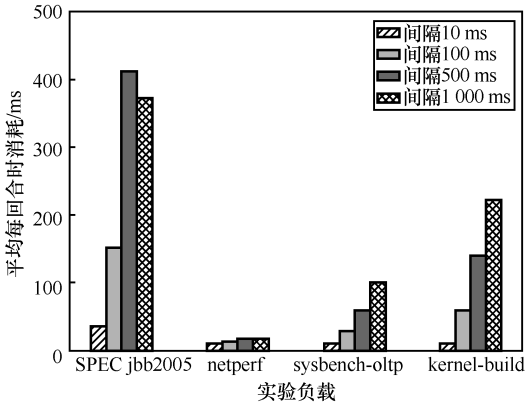

图 7 不同负载下平均每回合数据记录时间消耗

从而增加了数据记录所消耗的总时间。因此可以认为，数据记录间隔时间为 1 000 ms 时，所带来的数据记录消耗最小。但是选择合适的数据记录间隔时间，除了需要考虑数据记录的时间消耗性能指标之外，还需要考虑平均故障恢复时间这一重要性能指标。

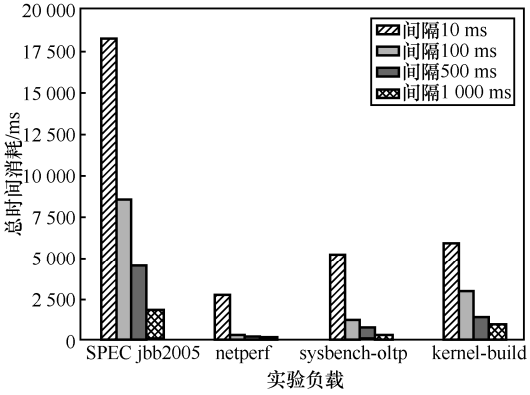

图 8 不同负载下整个迁移过程中记录数据的时间消耗

不同负载下的平均故障恢复时间如图9所示。由图 9 可以看出，不同的数据记录间隔时间平均故障恢复时间相差不大。故障恢复时间代表当迁移过程中虚拟机出现故障时，需要停止虚拟机服务重新恢复虚拟机运行的这段时间。故障恢复时间的长短取决于需要读取的页面和不确定性事件数据的数量，数据量越多导致故障恢复的时间越长。对于相同的负载，在整个迁移过程中单位时间内的页面修改量大致相同，并且根据哈希图方法，统计负载运行过程被修改过页面的数量后

发现，在负载运行一段时间后被修改过的内存页面数量将达到一定的极值并保持稳定。因此，从图 9 的结果中呈现出平均恢复时间大致相同的现象。

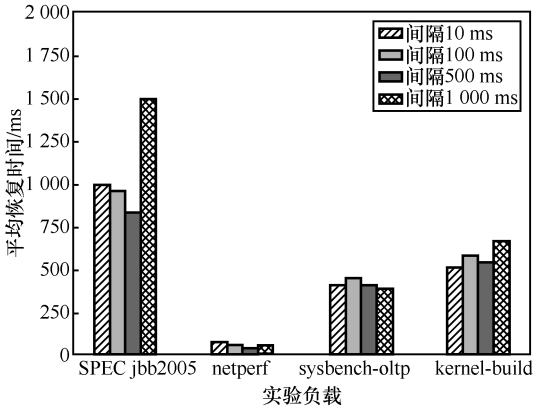

图 9　不同负载下的平均故障恢复时间

不同负载下的客户程序性能影响情况如图 10 所示。由图 10 可以看出，数据记录间隔时间为 1 000 ms 时，对客户程序的影响程度最低。显然当需要记录的数据量越大时，数据记录所消耗的时间也就越多，从而导致退出客户程序的时间越长，降低客户程序的执行性能。

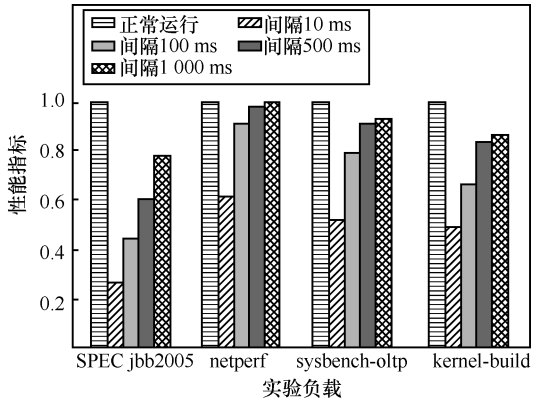

图 10　不同负载下的客户程序性能影响情况

可以得出，引入故障容错方法必然会对现有的程序带来一定程度的性能损耗，考虑数据记录的时间消耗、平均故障恢复时间以及对客户程序性能造成影响的实验结果，可以认为当故障容错方法的数据记录间隔时间设置为 1 000 ms 时，所造成的性能损耗最低。因此，最终选择 1 000 ms 作为故障容错方法的数据记录间隔时间。

传统的后拷贝迁移算法在迁移过程中缺乏稳定性，源端虚拟机故障或目标端虚拟机故障都会造成虚拟机状态的丢失，从而导致迁移失败。因此本文提出了基于事件同步的故障容错方法，通过学习现有的虚拟机故障容错机制，包括检查点和帧同步。结合两者的优点，以检查点为故障恢复起点，并使用帧同步的方式对检查点间产生的事件进行同步，保障了故障恢复完整性的同时，减少了相关的数据传输消耗。最终实现了后拷贝迁移过程中可能发生故障场景的正确恢复。

5　结束语

后拷贝迁移算法是虚拟机动态迁移调度的重要支撑之一，但是目前相关的研究工作还比较少，因此研究如何优化后拷贝迁移算法有重要的实际意义。本文深入研究 QEMU-KVM 源代码，对后拷贝迁移算法展开了研究。分析提炼后拷贝迁移算法存在的稳定性问题，并结合现有文献资料以及相关领域的研究方法，对其优化思想进行了详细的阐述，最终针对稳定性问题提出了基于事件同步的故障容错方法。传统的后拷贝迁移算法在迁移过程中缺乏稳定性，无论源端虚拟机故障还是目标端虚拟机的故障都会造成虚拟机状态的丢失，从而导致迁移失败。因此本文提出了基于事件同步的故障容错方法，通过学习现有的虚拟机故障容错机制，包括检查点和帧同步。结合两者的优点，以检查点为故障恢复起点，并使用帧同步的方式对检查点间产生的事件进行同步，保障了故障恢复完整性的同时，减少了相关的数据传输消耗。最终实现了后拷贝迁移过程中可能发生故障场景的正确恢复。

参考文献：

[1]　张建勋, 古志民, 郑超. 云计算研究进展综述[J]. 计算机应

用研究, 2010, 27(2): 429-433.
ZHANG J X, GU Z M, ZHENG C. Survey of research progress on cloud computing[J]. Application Research of Computers, 2010, 27(2): 429-433.

[2] 张耀祥. 云计算和虚拟化技术[J]. 计算机安全, 2011(5): 80-82.
ZHANG Y X. Cloud computing and virtualization security[J]. Computer Security, 2011(5): 80-82.

[3] ROSENBLUM M, GARFINKEL T. Virtual machine monitors: current technology and future trends[J]. Computer, 2005, 38(5): 39-47.

[4] AVI KIVITY, YANIV KAMAY, DOR LAOR, et al. kvm: the Linux virtual machine monitor[C]// Linux Symposium. [S. l. : s. n.], 2007.

[5] BARHAM P, DRAGOVIC B, FRASER K, et al. Xen and the art of virtualization[J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 164-177.

[6] VELTE A, VELTE T. Microsoft virtualization with Hyper-V[M]. [S. l. : s. n.], 2009.

[7] NOSHY M, IBRAHIM A, ALI H A. Optimization of live virtual machine migration in cloud computing: a survey and future directions[J]. Journal of Network and Computer Applications, 2018(110): 1-10.

[8] CLARK C , FRASER K , HAND S , et al. Live Migration of Virtual Machines[C]//Conference on Symposium on Networked Systems Design & Implementation. [S. l. : s. n.], 2005.

[9] CHASHOO S F, MALHOTRA D. VM_Mig_Framework: virtual machine migration with and without ballooning[C]//Proceedings of 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC). Piscataway: IEEE Press, 2018: 368-373.

[10] SHAN Z Y, QIAO J Z, LIN S K. Fix page fault in post-copy live migration with RemotePF page table assistant[C]//Proceedings of 2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES). Piscataway: IEEE Press, 2018: 40-43.

[11] SU K, CHEN W Z, LI G X, et al. RPFF: a remote page-fault filter for post-copy live migration[C]//Proceedings of 2015 IEEE International Conference on Smart City/SocialCom/ SustainCom (SmartCity). Piscataway: IEEE Press, 2015: 938-943.

[12] JALAEI N, SAFI-ESFAHANI F. VCSP: virtual CPU scheduling for post-copy live migration of virtual machines[J]. International Journal of Information Technology, 2021, 13(1): 239-250.

[13] 孙淑娴. 基于 Post-Copy 算法的虚拟机动态迁移机制研究[D]. 沈阳: 东北大学, 2015.
SUN S X. Research of post-copy algorithm based live virtual machine migration[D]. Shenyang: Northeastern University, 2015.

[14] FERNANDO D, TERNER J, GOPALAN K, et al. Live migration ate my VM: recovering a virtual machine after failure of post-copy live migration[C]//Proceedings of IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 343-351.

[15] CHOU C C, CHEN Y, MILOJICIC D, et al. Optimizing post-copy live migration with system-level checkpoint using fabric-attached memory[C]//Proceedings of 2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC). Piscataway: IEEE Press, 2019: 16-24.

[16] EGGER B, CHO Y, JO C, et al. Efficient checkpointing of live virtual machines[J]. IEEE Transactions on Computers, 2016, 65(10): 3041-3054.

[作者简介]

陈双喜（1980− ），男，浙江大学博士生，嘉兴职业技术学院副教授，主要研究方向为网络空间安全拟态防御。

赵若琰（1998− ），女，浙江大学硕士生，主要研究方向为网络空间安全。

刘会（1980− ），女，嘉兴职业技术学院讲师，主要研究方向为优化的 BP 神经网络在计算机测配色系统中的应用。

吴春明（1967− ），男，博士，浙江大学教授、博士生导师，主要研究方向为网络安全、互联网体系结构、柔性可重构网络、网络资源弹性管控。

阮伟（1969− ），男，浙江大学教授，主要研究方向为工业控制系统信息安全、主动防御的工业控制系统、工业控制系统攻防试验场关键技术等。