# 实验报告

学院：计算机科学与技术 班级：硕1709班 学号：M201773115 姓名：张阳

学院：计算机科学与技术 班级：硕1708班 学号：M201773145 姓名：李呈隆

****论文中实验概述：****

在论文中我们通过三个benchmark（Postmark，Memory microbenchmark，Clonebenchmark）对多个虚拟机进行同时操作，以了解在不同压力条件下虚拟机中的RCU抢占效应对虚拟机性能的影响。

1. Benchmark介绍

Postmark：主要用于测试文件系统在邮件系统或电子商务系统中的性能，其特点是需要频繁，大量的存取小文件。Postmark的测试的原理是创建一个测试文件池，文件的数量和最大、最小长度可以设定,数据总量是一定的，创建完成后,Postmark对文件池进行一系列的事务(transaction)操作，根据从实际应用中统计的结果，设定每一个事务包括一次创建或删除操作和一次读或添加操作，在有些情况下,文件系统的缓存策略可能对性能造成影响，Postmark可以通过对创建/删除以及读/添加操作的比例进行修改来抵消这种影响，事务操作进行完毕后，Post对文件池进行删除操作，并结束测试,输出结果。

Microbenchmark（基准测试）：Memory microbenchmark：作为内核模块实现，分配一个大小为1K的对象，通过回调函数来回收GP之后的对象。测试RCU抢占对内存的影响。Clone microbenchmark：通过在用户空间程序的循环中调用clone（）系统调用来测量新命名空间的克隆速度。测试RCU抢占问题对运行速度的影响。

1. 实验配置

Intel Xeon E5-4640处理器，4组CPU，每组8个双向超线程的核，合计64核。虚拟机的操作系统为Linux kernel 4.5.0.由于本实验所需配置太高，我们组实在无法实现，因此具体实验中的结果均为论文中的结果。

1. 具体实验

实验一：

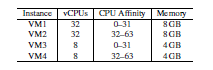


图 1 实验一处理器核心分配

VM1:运行Postmark benchmark，VM2运行Memory microbenchmark。VM3与VM4运行随机间隙抢占资源的程序。Postmark实验结果如图 2所示：Memory microbenchmark实验结果如

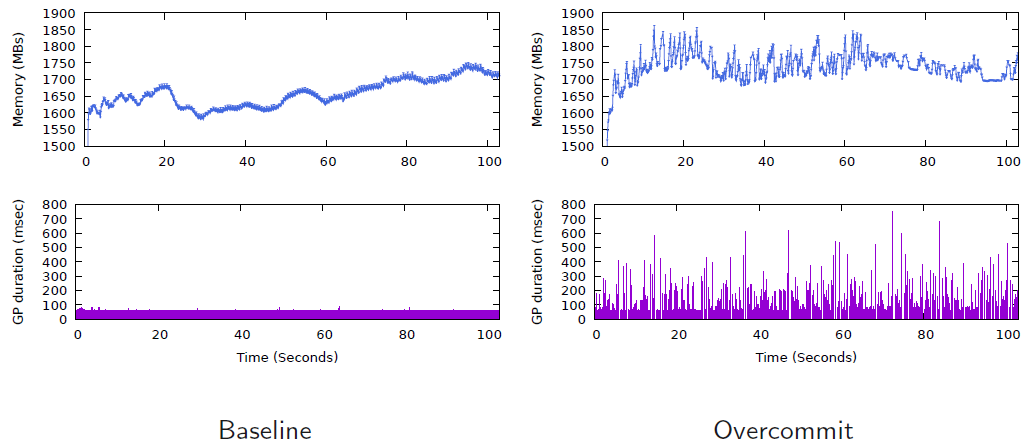


图 2 Postmark benchmark的执行结果，Baseline是只运行VM1，VM2的结果。

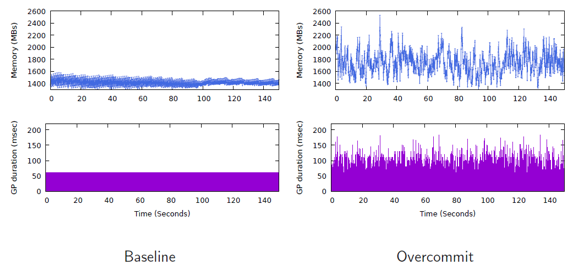


图 3 Memory microbenchmark执行结果，Baseline是只运行VM1，VM2的结果。

实验结果分析：对于Postmark benchmark ，GP时间最大值增加了26.37 倍，GP 时间平均值增加了2.18 倍，在每个GP 时间，CPU占用率增加了2.9 倍。

对于Memory microbenchmark：GP 时间最大值增加了3.62 倍，GP 时间平均值增加了30.26% ，内存占用峰值增加了约50%。

实验二：

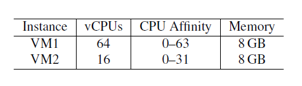


图 4实验二虚拟机核心分配

VM1：运行clone microbenchmark，VM2运行CPU-hogging user processes。

实验结果如图 5所示：

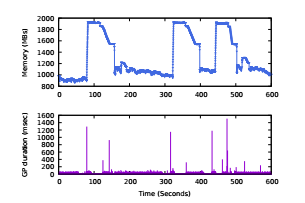


图 5 主机过载时Memory trace and GP duration变化图

实验结果分析：RCU 读抢占可能会影响虚拟机的密度和整合度。GP 段内的同步等待导致时延达到峰值；内存占用达到峰值，内存占用的峰值大小增加，会增加内存碎片，导致内存膨胀；CPU利用率增加；跨VM交互时，一个虚拟机中的CPU 利用率达到峰值可能导致另一个虚拟机进入GP。

总结：论文中的实验逻辑十分清晰，所给实验结果图对比明确，效果明显，能够直观的看出RCU 读抢占问题对虚拟机的系统性能有着显著影响。但是重现实验所需配置要求太高，我们组也不是做这个方向的，因此在相关实验的操作中，主要放在实验中所用到的benchmark的理解和实现。

# 相关实验

Postmark的安装和测试：

1. Postmark的安装

在虚拟机（Linux kernel 4.5.0系统）中通过apt-get install postmark来安装。安装好后，直接使用命令postmark，就可以使用。

1. Postmark的实用测试

有两种方式可以使用Postmark，一种是配置xxx. config文件。一种是直接在命令行中执行。test.config文件如下：

|  |
| --- |
| set size 128000 //设计文件大小的上限和下限  set umnbers 1000 //设置并发文件数  set transactions 20000 //设置事物数  run  quit |

运行结果如下：

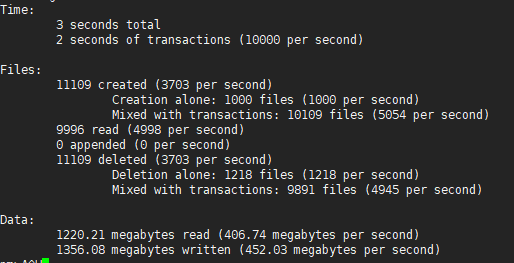


图 6 Postmark 运行实例

Hackbench的安装和测试：

文中没有提到Microbenchmark用的具体是哪一个benchmark，因此我选择了Hackbench进行测试。Hackbench是一个测试Linux进程调度器的性能开销和可伸缩性的基准测试程序，Hackbench测试可以运行在不同的硬件平台上和不同的Linux操作系统版本上，通过模拟一组C/S模式下的客户端进程和服务器端进程的通信来测试Linux进程调度器的性能开销和可伸缩性。

Hackbench测试分为若干组进行，组数num\_group由命令行给出，每个组有num\_fds组读写进程，num\_fds可以设置初始值。在每一个测试组中，创建num\_fds个客户端进程（读进程）和num\_fds个服务器进程（写进程）分别负责信息的读写工作，创建num\_fds个管道负责读写进程之间的通信。读进程负责从从管道中读取数据，写进程负责向管道中写数据。参数loops为每个写进程向每个读进程写入的信息条数，每条信息可设定为100个字符。每个写进程向每个管道中分别发送loops条信息，故每个写进程共发送num\_fds\*loops条信息，所有写进程共发送num\_fds\*num\_fds\*loops条信息。每个读进程从与之相连的管道中读取num\_fds\*loops条信息（每个写进程发送loops条，共num\_fds个写进程），所有读进程共读取num\_fds\*num\_fds\*loops条信息，与写进程发送的信息数相等。   测试中用到两个管道readyfds[2]和wakefds[2]来进行主进程和所有读写子进程的同步工作。主进程在创建好所有的管道和读写进程之后，通过readyfds[2]和wakefds[2]的同步，在所有读写进程准备好之后，开始计时，用start标识此时间点。计时开始后，各个读写进程之间开始相互通信，读进程在读完数据和写进程在写完数据后，进程结束。所有读写进程都完成工作之后，计时结束，用stop标识此时间点。然后计算此段所消耗的时间diff = stop – start。最后输出diff。

Hackbench测试运行结果如下：

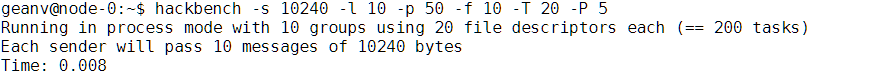


图 7 Hackbench测试结果

此时datasize的规模为10240个字节，每个写进程向读进程发送的信息数为10，组数num\_groups为10，管道的数量为50，线程的数量为50，进程的数量为5.

# 总结

因为能力有限，我们小组没有完全复现出本次实验。虽然我们组三个人都不是做系统结构方向，但是通过该课程的学习，我们三个人学到了很多系统结构相关的知识。段沛云同学主要负责文献综述部分，李呈隆和张阳同学负责PPT的制作，同时二人共同完成了该论文在课堂上的讲述。张阳同学一直在试图复现出实验，最终和李呈隆同学一起完成了这份实验室报告。虽然本次课程时间较短，但是在一定程度上锻炼我们的动手能力，让我们受益匪浅。