# 测试分析

## 1.1 实验环境搭建

Our experimental platform is a HP Compaq 8100 Elite CMT PC with an Intel Core i7-870 running at 2.93 GHz with vt-d feature enabled and we use Xen version 4.2.1 as the hypervisor while domain0 uses the kernel version 3.2.0-51-generic-pae.

In addition, Intel VT-d technique is enabled in the BIOS.

The grub.conf of the para-guest domain 0:

menuentry 'Ubuntu GNU/Linux，Xen 4.2.1' --class ubuntu --class gnu-linux --class gnu --class os --class xen {

        insmod part\_msdos

        insmod ext2

        set root='(hd1,msdos2)'

  search --no-floppy --fs-uuid --set=root b23f1bc8-31e1-4590-a063-2109e68309fd

  multiboot /boot/xen-4.2.1.gz **iommu=verbose iommu=dom0-strict** placeholder sched=credit loglvl=all guest\_loglvl=all debug\_stack\_lines=80 console=vga,com1 com1=115200,8n1

      module /boot/vmlinuz-3.2.0-51-generic-pae placeholder root=UUID=b23f1bc8-31e1-4590-a063-2109e68309fd ro  quiet splash console=tty0 max\_loop=128 xencons=ttyS0,115200

module  /boot/initrd.img-3.2.0-51-generic-pae

}

### 1.1.1 串口配置（如何配置串口输出打印）

串口的一端是实验机，另一端用的是windows系统。接下来我们对这两台机器的串口配置作说明：

1. windows: 首先是安装USB转串口线的驱动；其次是设置串口的波特率115200，并选择一个尚未被使用的端口值。其路径是：设备管理器->端口属性->端口设置->高级。

2. 修改主机xen启动选项： /boot/grub/grub.conf

如上所示

## 1.2 实验方案

根据Intel vt-d文档，我们知道iommu提供了三种粒度的机制刷新iotlb，包括单页，单个虚拟机域，所有虚拟机域。通过源码可以看到，Xen在处理页类型变化时，所使用的iotlb刷新机制是基于单个页的queued invalidation interface.

简单介绍下单个页刷新的两种机制：register-based invalidation和queued invalidation，指出xen选择queued invalidation是基于性能上的考虑。

### 1.2.1 微基准测试

接下来，我们测试新设计的算法在微观上如何改善iotlb的刷新频率。总共做了三组测试，测试1是在算法加入之前iotlb的刷新频率；测试2是算法加入之后的iotlb刷新频率；测试3是根据我们之前向xen开源社区所提交的漏洞cve-2013-6375来评估新算法。

测试1：需要测出客户机系统未打算法补丁时，其运行时iotlb的刷新频率。具体来说，我们设计了一个名为fork的小程序在一分钟之内来大量创建进程，进而申请大量的页表页，从而触发页类型的频繁变化，导致xen对iotlb的频繁刷新。通过在最底层iotlb刷新函数内flush\_iotlb\_qi加上测试代码，计算出iotlb每秒钟的刷新次数。通过修改fork程序中进程创建的频率，得到iotlb每秒钟刷新次数的最大值。

测试2：将算法应用到客户机内核中，在与测试1取得次数最大值时的相同条件下，测出此时iotlb的刷新次数。

测试3：根据漏洞cve-2013-6375，我们知道如果不修复该bug，那么实际上xen是不会刷新iotlb的，在与测试1，2相同条件下，测出其刷新次数，这应该是所有改进算法的理论最佳值，可以将其作为基准线来评价本算法的优劣。

三个测试的结果如下图所示：

## 1.2.2 宏基准测试（先不放上去）

上述测试是用微观的方法评估算法对IO的性能优化程度，接着我们从利用网络测试工具netperf从宏观上对该算法进行评估。

目标测试iotlb flush和non-flush对network throughput的影响

## 1.3 实验小结

通过…，可以看出该算法从微观上对性能的改善。