


HumJb & HaHa

[首页](#) | [博文目录](#) | [关于我](#)



humjb_1983

博客访问：9497
博文数量：80
博客积分：0
博客等级：民兵
技术积分：685
用户组：普通用户
注册时间：2014-02-20 08:27

加关注

短消息

论坛

加好友

文章分类

全部博文 (80)

硬件相关 (5)

虚拟化 (13)

其他 (1)

Linux其他方面 (3)


Linux内核 (57)

未分配的博文 (1)


文章存档

2014年 (80)

我的朋友




asuka20




321leon


最近访客




arm-linu




码出一片



pisming



jeppeter



刘一痕



SCvsCS

KVM基本原理及架构四-内存虚拟化2014-04-10 12:48:26

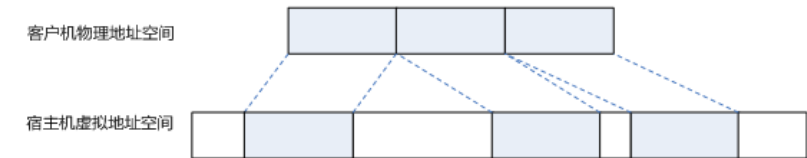
分类：LINUX

4 内存虚拟化

4.1 客户机物理地址空间

在物理机上，虚拟地址通过Guest页表即可转换为物理地址。但是在虚拟化环境中，由于VMM和VM都需要独立的地址空间，则产生了冲突。

为实现内存虚拟化，让客户机使用一个隔离的、从零开始且具有连续的内存空间，KVM 引入一层新的地址空间，即客户机物理地址空间 (Guest Physical Address, GPA)，该地址空间并不是真正的物理地址空间，它只是宿主机(Host主机)虚拟地址空间在Guest地址空间的一个映射。对Guest来说，客户机物理地址空间都是从零开始的连续地址空间，但对于宿主机来说，客户机的物理地址空间并不一定是连续的，客户机物理地址空间有可能映射在若干个不连续的宿主机地址区间，如下图所示：



由于物理MMU只能通过Host机的物理地址(Host Physical Address, HPA)进行寻址，所以实现内存虚拟化，关键是需要将Guest机的虚拟地址(Guest Virtual Address, GVA)转换为HPA。传统的实现方案中，这个过程需要经历：GVA→GPA→HVA→HPA的转换过程，需要对地址进行多次转换，而且需要KVM的介入，效率非常低。为提供GVA到HPA的地址转换效率，KVM提供了两种地址转换方式：

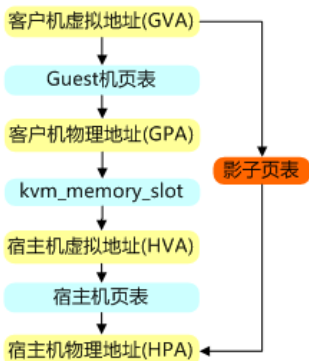
1、影子页表(Shadow Page Table)，是纯软件的实现方式

2、基于硬件特性的地址转换。如基于Intel EPT(Extended Page Table，扩展页表)，或AMD NPT(Nested Page Table，嵌套页表)

4.2 影子页表

4.2.1 基本原理

由于内存虚拟化在将GVA转换为HPA的过程中，需要经历多次转换，无法直接使用Guest机页表和CR3。使用影子页表(Shadow Page Table)可以实现客户机虚拟地址(GVA)到宿主机物理地址(HPA)的直接转换，与传统方式的转换过程对比如下：



影子页表中记录的是GVA跟HPA的对应关系，每个页表项指向的都是宿主机的物理地址。由于客户机中每个进程都有自己的虚拟地址空间，所以 KVM 需要为客户机中的每个进程页表都要维护一套相应的影子页表。在Guest机访问内存时，VMM在物理MMU中载入的是Guest机当前页表所对应的影子页表，从而



风铃之音



embedde



chrxy

订阅

推荐博文

- 云计算-Azure-3.负载均衡集...
- 读书与写论文的引导书——leo...
- 在framework层添加自己的jar...
- tcpdump工具浅析
- python json ajax django四星...
- Solaris文件管理和目录管理...
- Solaris退出系统,改变系统运...
- 监控Data Guard实时同步...
- Oracle的告警日志之v\$diag_al...
- 使用AWR生成报表

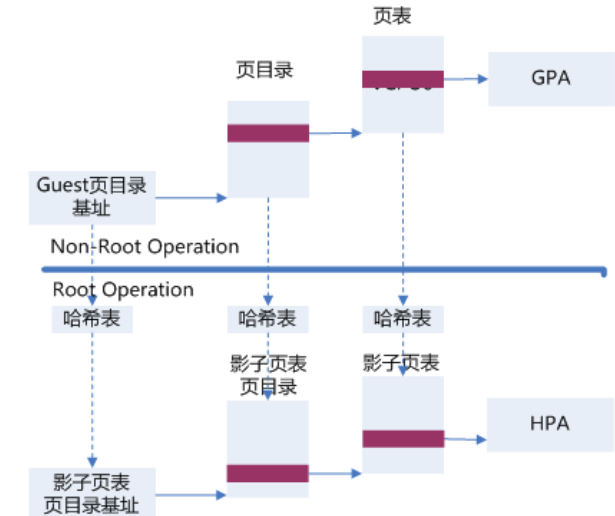
热词专题

- Debian设置
- 欢迎kkkkkkkybbb在ChinaUnix...
- 虚拟机ping不通win7宿主机...
- 安装oracle
- 关于STM32的SPI的问题

实现GVA到HPA的直接转换。

Guest机中的每一个页表项都有一个影子页表项与之相对应。为了快速检索Guest机页表所对应的影子页表，KVM 为每个客户机都维护了一个哈希表，影子页表和Guest机页表通过此哈希表进行映射，基本原理如下：

对于每一个Guest机来说，Guest机的页目录/页表都有唯一的GPA，通过页目录/页表的GPA就可以在哈希链表中快速找到对应的影子页目录/页表。在检索哈希表时，KVM 把Guest页目录/页表的客户机物理地址低10位作为键值进行索引，根据其键值定位到对应的链表，然后遍历此链表找到对应的影子页目录/页表。当然，如果没有找到对应的影子页目录/页表，则说明影子页表项和Guest页表项的对应关系还没有建立，此时KVM 会为其分配新的物理页，并建立起Guest页目录/页表和对应的影子页目录/页表之间的映射。



4.2.2 影子页表的建立与更新

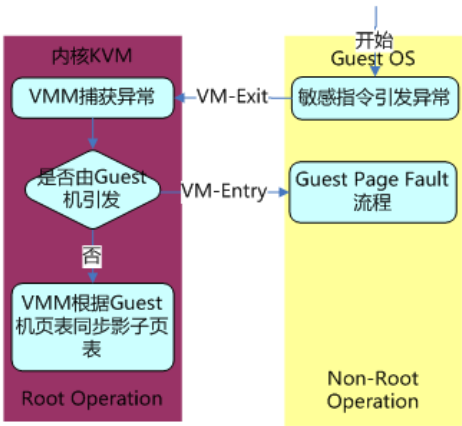
影子页表的建立和更新过程交织在一起，影子页表的建立和更新主要发生在如下3中情况下：

- 1、 Guest OS修改Guest CR3寄存器。由于相关指令为敏感指令，所以相关操作会被VMM截获，此时VMM会根据相关情况进行影子页表的维护。比如，当客户机切换进程时，客户机操作系统会把待切换进程的页表基址载入 CR3，而该特权指令将被VMM截获，进行新的处理，即在哈希表中找到与此页表基址对应的影子页表基址，载入客户机 CR3，使客户机在恢复运行时 CR3 实际指向的是新切换进程对应的影子页表。
- 2、 因Guest机页表和影子页表不一致而触发的缺页异常，此时也会VM-Exit到VMM，进而可进行相关维护操作。
- 3、 Guest OS中执行INVLPG指令刷新TLB时，由于INVLPG指令为敏感指令，所以该操作也会被VMM进行截获，并进行影子页表相关维护操作。

其中，第2中情况发生几率最高，相关处理也最复杂。如下做重点描述。不同的缺页异常，处理方式不用，常见的缺页异常包括如下3类：

- 1、 影子页表初始化时产生的缺页异常。在虚拟机运行之初，VMM中与Guest机页表对应的影子页表都没有建立，而物理CR3中载入的却是影子页目录地址，所以，此时任何的内存操作都会引发异常，如果此时Guest机的相应页表已经建立，那么处理这种异常即是建立相应的影子页表即可；如果Guest机的页表项尚未建立，那就是Guest机自身的缺页异常，即为如下的第2中情况。
- 2、 Guest机上的缺页异常。如果Guest OS尚未给这个GVA分配Guest机物理页，即相应的Guest机页表项尚未建立，此时将引发缺页异常。另外，当Guest机访问的Guest页表项存在位(Present Bit)为0，或相关访问权限不匹配时，也将引发缺页异常。
- 3、 VMM将Host机物理页换出到硬盘上时引发的缺页异常。

影子页表缺页异常的默认处理流程



VMM捕获缺页异常(VM-Exit)，并检查此异常是否由Guest即自身引发，如果是，则将直接返回Guest OS(Vm-Entry)，然后由Guest OS自身的page fault流程处理；如果不是，则为影子页表和Guest机页表不一致导致，这样的异常也叫“影子缺页异常”，此时，VMM会根据Guest机页表同步影子页表，过程如下：

- 1、 VMM根据Guest机页表项建立影子页目录和页表结构
- 2、 VMM根据发生缺页异常的GVA，在Guest机页表的相应表项中得到对应的GPA
- 3、 VMM根据GPA，在GPA与HPA的映射表中(通过之前描述的HASH表建立)，得到相应的HPA，再将HPA填入到影子页表的相应表项中。

影子页表和Guest机页表不是时刻同步的，只有在需要时才进行通过，从某种角度看，影子页表可以看做是Guest页表的TLB，常称为虚拟TLB(VTLB)。

影子页表解决了传统IA32架构下的内存虚拟化问题，由于影子页表可被载入物理 MMU 为客户机直接寻址使用，所以客户机的大多数内存访问都可以在没有 KVM 介入的情况下正常执行，没有额外的地址转换开销，也就大大提高了客户机运行的效率。但也有比较明显的缺点：

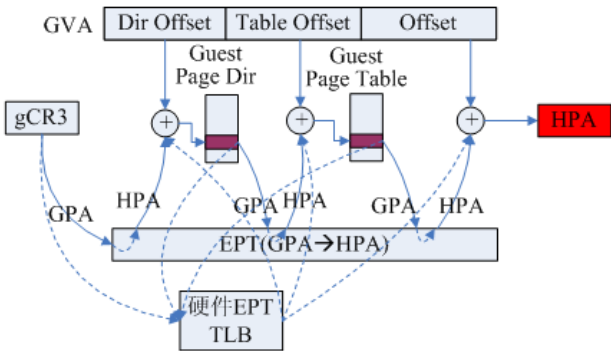
- 1、 实现复杂。影子页表同步需要考虑各种情况。
- 2、 内存开销大。需要为每个Guest机进程维护一个影子页表。

4.3 EPT 页表

4.3.1 基本原理

为解决影子页表的问题，Intel和AMD都提供了相应的硬件技术，直接在硬件上支持GPA到HPA的转换，从而大大降低了内存虚拟化的难度，并提升了相关性能。本文主要描述Intel EPT技术。

EPT 技术在原有Guest机页表对GVA到GPA转换的基础上，又引入了 EPT 页表来实现GPA到HPA转换，这两次地址转换都是由硬件自动完成，可高效的实现地址转换。Guest运行时，Guest页表被载入CR3，而 EPT 页表被载入专门的 EPT 页表指针寄存器 EPTP。从GVA到HPA的具体转换过程如下(以经典的2级页表为例)：



完整的地址翻译流程描述为：

- 1、 Guest OS加载Guest进程的gCR3，gCR3中存放的是Guest进程页目录表的GPA。
- 2、 处于非根模式的CPU的MMU查询硬件EPT TLB，如果有所请求的GPA到HPA的映射，则使用其对应的HPA作为Guest页目录表的基址。
- 3、 如没有所请求的GPA到HPA的映射，则查询EPT，获得gCR3所映射的HPA，并将其作为Guest页目录表的基址。
- 4、 根据GVA获得页目录偏移(图中的Dir Offset)，获得用于索引Guest页表的基址，该地址为GPA。
- 5、 再由VCPU的MMU查询硬件EPT TLB，如果有所请求的GPA到HPA的映射，则使用其对应的HPA作为Guest页表的基址。

- 6、如没有所请求的GPA到HPA的映射，则查询EPT，将其转换为HPA，使用该HPA再加上GVA中的页表偏移(图中的Table Offset)，即可得到PTE(页表项)的GPA。
- 7、再由VCPU的MMU查询硬件EPT TLB，如果有所请求的GPA到HPA的映射，则其对应的HPA加上GVA中的Offset即为最终的宿主机物理地址(HPA)。
- 8、如没有所请求的GPA到HPA的映射，则查询EPT，将其转换为HPA，使用该HPA加上GVA中的Offset即为最终的宿主机物理地址(HPA)。

EPT页表实现GPA到HPA的转换的原理，与Guest页表实现GVA到GPA的转换原理相同，需要经历多级页表的查询，图中没有详细画出。假设Guest机有m级页表，宿主机EPT有n级，在TLB均miss的最坏情况下，会产生m*n次内存访问，完成一次客户机的地址翻译，EPT硬件通过增大硬件EPT TLB来尽量减少内存访问。

4.3.2 EPT缺页异常处理

在GPA到HPA转换的过程中，由于缺页、写权限不足等原因也会导致客户机退出，产生 EPT 异常。对于 EPT 缺页异常，处理过程大致如下：

- 1、KVM 首先根据引起异常的GHA，映射到对应的HVA；
- 2、然后为此虚拟地址分配新的物理页；
- 3、最后 KVM 再更新 EPT 页表，建立起引起异常的GPA到HPA的映射。

EPT 页表相对于影子页表，其实现方式大大简化，主要地址转换工作都由硬件自动完成，而且Guest内部的缺页异常也不会导致VM-Exit，因此Guest运行性能更好，开销更小。

阅读(145) | 评论(0) | 转发(0) |

上一篇：KVM基本原理及架构三-CPU虚拟化
下一篇：linux下进程堆栈下溢出判断及扩展实现

0

相关热门文章

ARM linux kernel启动流程 he...	linux 常见服务端口	C语言 如何在一个整型左边补0...
怎么找回回收站里的文件...	【ROOTFS搭建】busybox的httpd...	python无法爬取阿里巴巴的数据...
用C语言写的程序不安全...	xmanager 2.0 for linux配置	linux-2.6.28 和linux-2.6.32....
台湾外贸仿牌服务器特价了 工...	什么是shell	linux su - username -c 命...
戒毒患者为什么屡戒屡吸？戒毒...	linux socket的bug??	我不得不在这里问一下网站使用...

给主人留下些什么吧！~~

评论热议

登录后评论。

[登录](#) [注册](#)

2014年7月20日

KVM基本原理及架构四-内存虚拟化-humjb_1983-ChinaUnix博客

感谢所有关心和支持过ChinaUnix的朋友们

京ICP证041476号 京ICP证060528号