二、 概述和工具介绍

在这一章中，我们先介绍一些贯穿全书的概念，比如 Hypervisor，VT-x，VT-d，SVM 等等。然后我们会简略介绍下 NewBluePill 项目背景及其所采用的硬件虚拟化技术。最后我们会介绍一些调试工具，这些调试工具对于我们理解整个代码过程，调试 NewBluePill 程序都有很大的帮助。

1) Hypervisor 概述

a) 虚拟化的历史

在讨论 Hypervisor 之前首先谈谈虚拟，虚拟（virtualization）指对计算机资源的抽象，一种常用的定义是“虚拟就是这样的一种技术，它隐藏掉了系统，应用和终端用户赖以交互的计算机资源的物理性的一面，最常做的方法就是把单一的物理资源转化为多个逻辑资源，当然也可以把多个物理资源转化为一个逻辑资源（这在存储设备和服务器上很常见）”

实际上，虚拟技术早在 20 世纪 60 年代就已出现，最早由 IBM 提出，并且应用于计算技术的许多领域，模拟的对象也多种多样，从整台主机到一个组件，其实打印机就可以看成是一直在使用虚拟化技术的，总是有一个打印机守护进程运行在系统中，在操作系统看来，它就是一个虚拟的打印机，任何打印任务都是与它交互，而只有这个进程才知道如何与真正的物理打印机正确通信，并进行正确的打印管理，保证每个 job 按序完成。

长久以来，用户常见的都是进程虚拟机，也就是作为已有操作系统的一个进程，完全通过软件的手段去模拟硬件，软件再翻译内存地址的方法实现物理机器的模拟，比如较老版本的 VMWare, VirtualPC 软件都属于这种。

在 2005 年和 2006 年，Intel 和 AMD 都开发出了支持硬件虚拟技术的 CPU，也就是在这时，x86 平台才真正有可能实现完全虚拟化[1]。[1]在 2007 年初的时候，Intel 还进一步的发布了 VT-d 技术规范，从而在硬件上支持 I/O 操作的虚拟化。随着硬件虚拟化技术越来越广泛的采用，开发者也开始虚拟技术来做一些其他的事情：当前 HVM 已经在虚拟机，安全，加密等领域上有所应用，例如 VMware Fusion, Parallels Desktop for Mac, Parallels Workstation 和 DNGuard HVM，随着虚拟化办公和应用的兴起，相信虚拟化技术也会在未来得到不断发展。

b) 硬件虚拟化技术

有了虚拟技术的基本概念，下面我们谈谈硬件虚拟化技术。硬件虚拟化技术（Hardware Enabled Virtualization，本书中简称 HEV），也就是在硬件层面上，更确切的说是在 CPU 里(VT-d 技术是在主板上北桥芯片支持)，对虚拟技术提供直接支持。在硬件虚拟化技术诞生前，编写虚拟机过程中，为了实现多个虚拟机上的真实物理地址隔离，需要编程实现把客户机的物理地址翻译为真实机器的物理地址。同时也需要给不同的客户机操作系统编写不同的虚拟设备驱动程序，使之能够共享同一真实硬件资源。硬件虚拟化技术则在硬件上实现了内存地址甚至于 I/O 设备的映射，因此大大简化了编写虚拟机的过程。而其硬件直接支持二次寻址和

[1] 完全虚拟化(Full Virtualization)，完整虚拟底层硬件，这就使得能运行在该底层硬件上的所有操作系统和它的应用程序，也都能运行在这个虚拟机上。

I/O 映射的特性也提升了虚拟机在运行时的性能。[1]

在硬件虚拟化技术中，一个重要的概念就是 HVM。HVM, Hypervisor Virtual Machine 的缩写（在本书中简称为 Hypervisor），是在使用硬件虚拟化技术时创建出来的特权层，该层提供给虚拟机开发者，用来实现虚拟硬件与真实硬件的通信和一些事件处理操作，因此 Hypervisor 的权限级别要高于等于操作系统权限。

c) 虚拟机的启动过程

使用了硬件虚拟化技术的虚拟机可以有三种引导 Guest 操作系统的方式：

1. 存在特殊 OS/Host OS，后启动 Hypervisor 的虚拟机启动过程
2. 存在特殊 OS/Host OS，先启动 Hypervisor 的虚拟机启动过程
3. 不存在特殊 OS/Host OS，先启动 Hypervisor 的虚拟机启动过程

- 存在特殊 OS/Host OS，后启动 Hypervisor 的虚拟机启动过程。采用这种启动过程的虚拟机代表是 KVM，其启动过程如下：
  a) 先启动宿主 Linux 操作系统
  b) 在 Linux 中启动 KVM 设备，从而启动了 Hypervisor
  c) 启动虚拟机，作为 Linux 进程运行

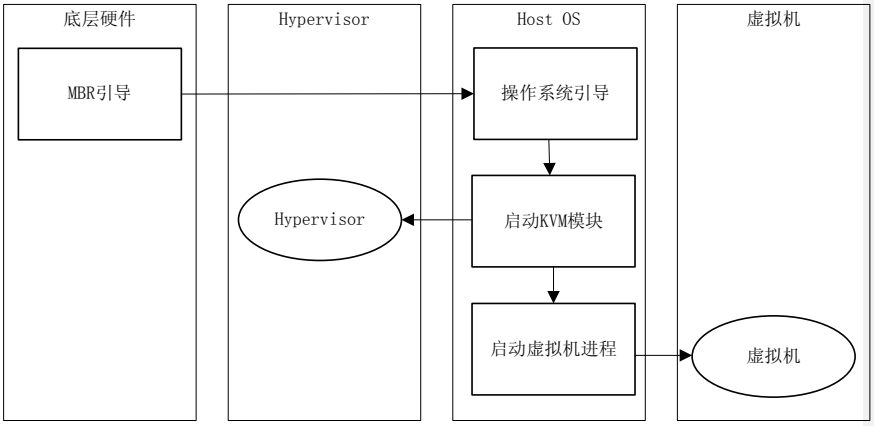

图 2.1 KVM 中虚拟机的启动过程

启动过程如图 2.1，可以看出，KVM 启动虚拟机的模式说明它不想脱离进程级虚拟机的本质，但是它要利用虚拟化技术进行加速。这样做的缺点在于需要一个 Host OS 充当载体。除 KVM 外，VMWare6.5 以上版本也是采用类似的架构，使用支持 HEV 技术的 CPU 进行加速。但是它们都需要再另外安装相应 Guest OS 上的驱动。

- 存在特殊 OS/Host OS，先启动 Hypervisor 的虚拟机启动过程。采用这种启动过程的虚拟机代表是 Xen，其启动过程如下：
  a) 先创建并启动 Hypervisor
  b) 引导 Dom0

[1] 一些优化技术也在硬件中被采用，比如专门用于二次寻址的 TLB，详细信息可以参考 Intel 和 AMD 的手册

c) 由 Hypervisor 和 Dom0 一起协作创建虚拟机
d) 启动该虚拟机[1]

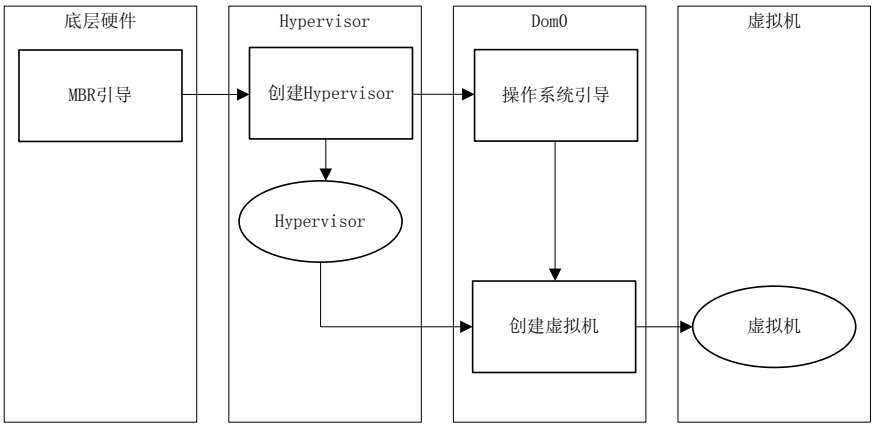

图 2.2 Xen 中虚拟机的启动过程

启动过程如图 2.2，可以看出，Xen 中仍存在 Dom0 是因为它要适应过去未出现 HEV 技术时的架构，所以无论是 Dom0 还是 Hypervisor 的实现都比较笨重，并且安装和配置也比较麻烦，同样需要另外安装相应 Guest OS 上的驱动。但是不可忽视的是 Xen 的虚拟化效率最高。

- 不存在特殊 OS/Host OS，先启动 Hypervisor 的虚拟机启动过程。当前暂时没有采用这种启动过程的虚拟机软件（暂时称之为 UVM, Unknown Virtual Machine），其启动过程如下：
  a) 先创建并启动 Hypervisor
  b) 从 Hypervisor 中创建并启动虚拟机

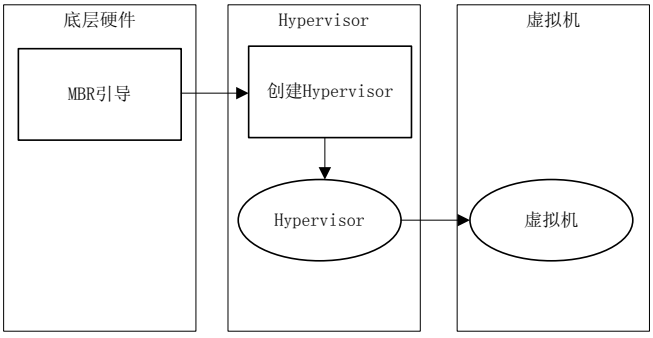

图 2.3 UVM 中虚拟机的启动过程

启动过程如图 2.3，这种虚拟机的设计目标在于：不需要在 Guest OS 中安装任何支持驱动。换句话说，Hypervisor 对于 Guest OS 完全透明，从而实现完全虚拟化（Full Virtualization）。这种方式的缺点是：Hypervisor 可能实现会很笨重，因而虚拟化效率不高，也会影响到系统安全，虚拟机的配置和管理可能也不易呈现给用户。

[1] Xen 中具体创建和启动虚拟机的过程会在“第 14 章 其它有关 HEV 项目”中介绍

d) Hypervisor 的使用架构

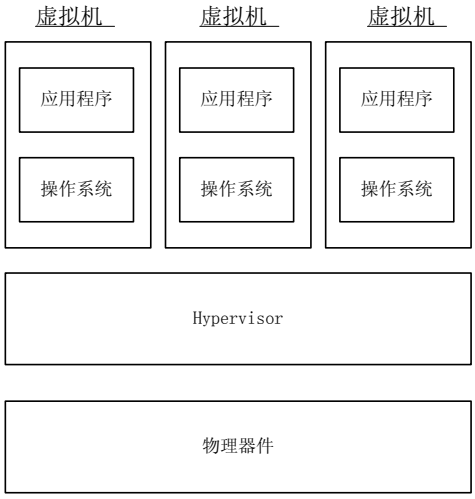

图 2.4：Hypervisor 使用架构图

前文中已经提过,Hypervisor 层的权限要高于等于操作系统的权限。操作系统的内核态已经处在了 Ring0 特权级上，因此 Hypervisor 层实际上要运行在一个新的特权级别上，我们称之为“Ring -1”特权级。同时需要新的指令，寄存器以及标志位去实现这个新增特权级的功能。

作为一种最佳实践方案，一般 Hypervisor 层的实现都是越简单越好。一方面，简单的实现能够尽量降低花在 Hypervisor 上的开销[1]，毕竟大多数这些开销在原先的操作系统上是不存在的，同时仅仅进出 Hypervisor 层就带来了很多额外开销。另一方面，复杂的程序实现容易引入程序漏洞，Hypervisor 也是如此，且一旦 Hypervisor 中的漏洞被恶意使用，由于其所处特权级高于操作系统，将使隐藏在其中的病毒、恶意程序很难被查出。

批注 [S1]: 后面要描述 Hypervisor 的开销问题
Virtualization.pdf 10

2) HVM 特定平台介绍

AMD-V

概述

AMD 芯片支持虚拟化的技术被称作 AMD-V(在技术文档中也被称为 SVM,其全称是 AMD Secure Virtual Machine)。其主要是通过一组能够影响到 VM 管理层（hypervisor）和寄宿层的中断实现的。同时 AMD-V 技术也对下面的要求提供了支持：

- 快速的 VM 监视层（VMM，和 hypervisor 是一个概念，下文均用 VMM 表示）和寄宿机（有时也指寄宿操作系统，若无特别说明均用 guest 表示）之间的切换

[1] 关于 Hypervisor 的开销问题，后面的章节会有介绍