Xen原理

Quinton_Lau (/u/f5ef357473a6) + 关注
2017.05.15 18:23* 字数 1200 阅读 305 评论 0 喜欢 3
(/u/f5ef357473a6)

原文链接 (https://link.jianshu.com?t=https://my.oschina.net/davehe/blog/94039)

1 Xen概述

1.1 简介

Xen是由剑桥大学计算机实验室开发的一个开源项目。是一个直接运行在计算机硬件之上的用以替代操作系统的软件层，它能够在计算机硬件上并发的运行多个客户操作系统（Guest OS）。目前已经在开源社区中得到了极大的推动。

Xen支持x86、x86-64、安腾(Itanium)、Power PC和ARM多种处理器，因此Xen可以在大量的计算设备上运行，目前Xen支持Linux、NetBSD、FreeBSD、Solaris、Windows和其他常用的操作系统作为客户操作系统在其管理程序上运行。

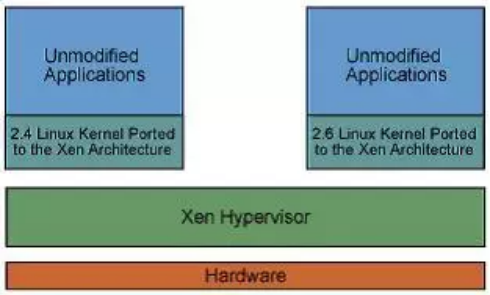
1.2 Xen虚拟化类型

Xen对虚拟机的虚拟化分为两大类，半虚拟化（Paravirtualization）和完全虚拟化（Hardware VirtualMachine）。

1.2.1 半虚拟化

半虚拟化（Paravirtualization）有些资料称为“超虚拟化”，简称为PV，是Xen主导的虚拟化技术。这种技术允许虚拟机操作系统感知到自己运行在Xen Hypervisor上而不是直接运行在硬件上，同时也可以识别出其他运行在相同环境中的客户虚拟机。

在Xen Hypervisor上运行的半虚拟化的操作系统，为了调用系统管理程序（Xen Hypervisor），要有选择地修改操作系统，然而却不需要修改操作系统上运行的应用程序。由于 Xen 需要修改操作系统内核，所以您不能直接让当前的 Linux 内核在 Xen 系统管理程序中运行，除非它已经移植到了Xen 架构。不过，如果当前系统可以使用新的已经移植到 Xen 架构的Linux 内核，那么您就可以不加修改地运行现有的系统。



半虚拟化虚拟机示意图

1.2.2 完全虚拟化

完全虚拟化（Hardware Virtual Machine）又称“硬件虚拟化”，简称HVM，是指运行在虚拟环境上的虚拟机在运行过程中始终感觉自己是直接运行在硬件之上的，并且感知不到在相同硬件环境下运行着其他虚拟机的虚拟技术。

在Xen Hypervisor运行的完全虚拟化虚拟机，所运行的操作系统都是标准的操作系统，

即：无需任何修改的操作系统版本。同时也需要提供特殊的硬件设备。
值的注意的是，在Xen上虚拟的Windows虚拟机必须采用完全虚拟化技术。

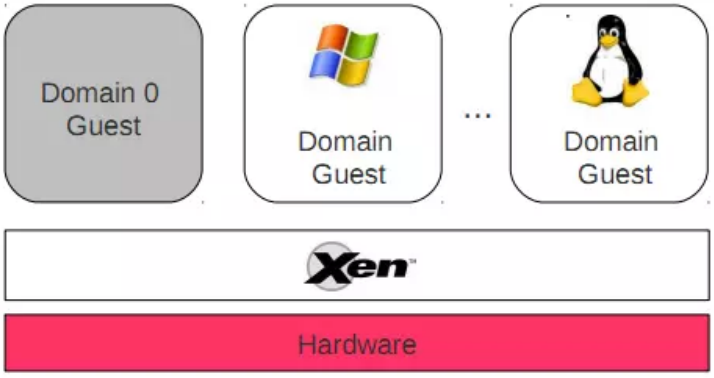
1.3 基本组件

Xen包含三大部分：

Xen Hypervisor：直接运行于硬件之上是Xen客户操作系统与硬件资源之间的访问接口(如：)。通过将客户操作系统与硬件进行分类，Xen管理系统可以允许客户操作系统安全，独立的运行在相同硬件环境之上。

Domain 0**：运行在Xen管理程序之上，具有直接访问硬件和管理其他客户操作系统的特权的客户操作系统。****

DomainU**：运行在Xen管理程序之上的普通客户操作系统或业务操作系统，不能直接访问硬件资源（如：内存，硬盘等），但可以独立并行的存在多个。****



Xen组件结构图

1.3.1 Xen Hypervisor

Xen Hypervisor是直接运行在硬件与所有操作系统之间的基本软件层。它负责为运行在硬件设备上的不同种类的虚拟机（不同操作系统）进行CPU调度和内存分配。Xen Hypervisor对虚拟机来说不单单是硬件的抽象接口，同时也控制虚拟机的执行，让他们之间共享通用的处理环境。

Xen Hypervisor不负责处理诸如网络、外部存储设备、视频或其他通用的I/O处理。

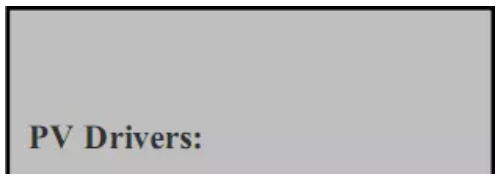
1.3.2 Domain 0

Domain 0 是经过修改的Linux内核，是运行在Xen Hypervisor之上独一无二的虚拟机，拥有访问物理I/O资源的特权，并且可以与其他运行在Xen Hypervisor之上的其他虚拟机进行交互。所有的Xen虚拟环境都需要先运行Domain 0，然后才能运行其他的虚拟客户机。

Domain 0 在Xen中担任管理员的角色，它负责管理其他虚拟客户机。

在Domain 0中包含两个驱动程序，用于支持其他客户虚拟机对于网络和硬盘的访问请求。这两个驱动分别是Network Backend Driver和Block Backend Driver。

Network Backend Driver直接与本地的网络硬件进行通信，用于处理来自Domain U客户机的所有关于网络的虚拟机请求。根据Domain U发出的请求Block Backend Driver直接与本地的存储设备进行通信然后，将数据读写到存储设备上。

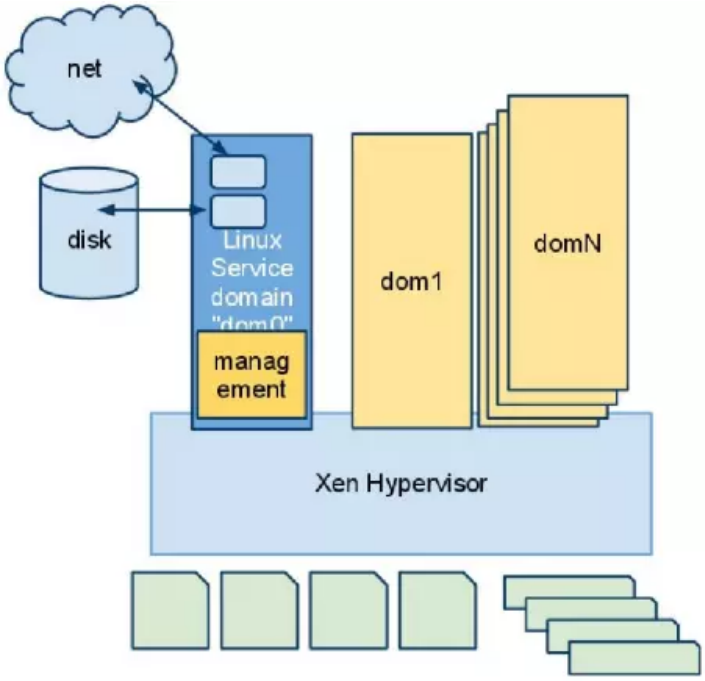


** ** Domain 0虚拟机

1.3.3 Domain U

Domain U客户虚拟机没有直接访问物理硬件的权限。所有在Xen Hypervisor上运行的半虚拟化客户虚拟机（简称：Domain U PV Guests）都是被修改过的基于Linux的操作系统、Solaris、FreeBSD和其他基于UNIX的操作系统。所有完全虚拟化客户虚拟机（简称：Domain U HVM Guests）则是标准的Windows和其他任何一种未被修改过的操作系统。

无论是半虚拟化Domain U还是完全虚拟化Domain U，作为客户虚拟机系统，Domain U在Xen Hypervisor上运行并行的存在多个，他们之间相互独立，每个Domain U都拥有自己所能操作的虚拟资源（如：内存，磁盘等）。而且允许单独一个Domain U进行重启和关机操作而不影响其他Domain U。



** ** Xen虚拟化示意图

2 Xen基本体系架构及运行原理

2.1.1 Xen体系架构

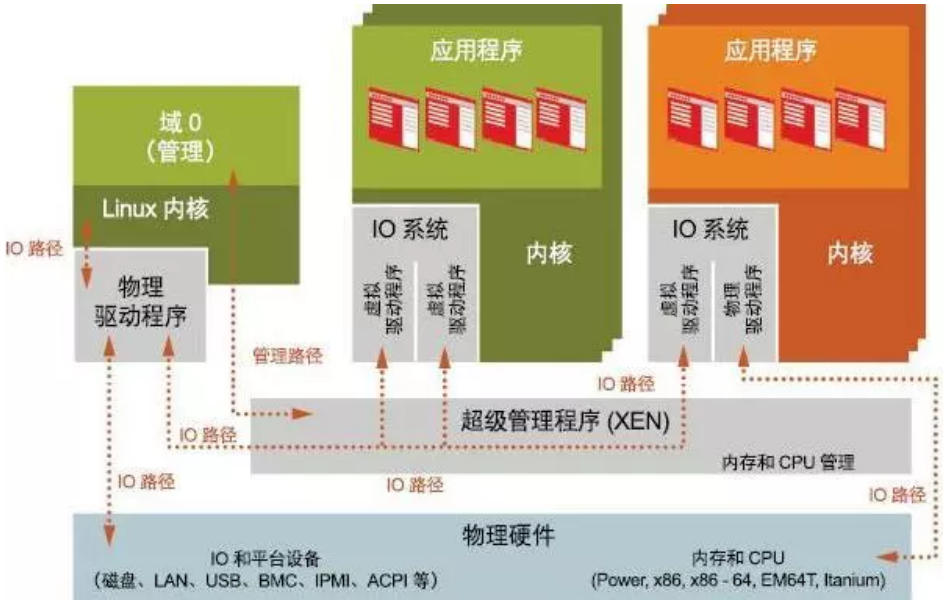
Xen 的 VMM (Xen Hyperviso) 位于操作系统和硬件之间，负责为上层运行的操作系统内核提供虚拟化的硬件资源，负责管理和分配这些资源，并确保上层虚拟机（称为域Domain）之间的相互隔离。Xen采用混合模式，因而设定了一个特权域用以辅助Xen管理其他的域，并提供虚拟的资源服务，该特权域称为Domain 0，而其余的域则称为Domain U。

Xen向Domain提供了一个抽象层，其中包含了管理和虚拟硬件的API。Domain 0内部包含了真实的设备驱动（原生设备驱动），可直接访问物理硬件，负责与 Xen 提供的管理 API 交互，并通过用户模式下的管理工具来管理 Xen 的虚拟机环境。

Xen2.0之后，引入了分离设备驱动模式。该模式在每个用户域中建立前端（front end）设备，在特权域(Dom0)中建立后端(back end)设备。所有的用户域操作系统像使用普通设备一样向前端设备发送请求，而前端设备通过IO请求描述符(IO descripror ring)和设备

通道（device channel）将这些请求以及用户域的身份信息发送到处于特权域中的后端设备。这种体系将控制信息传递和数据传递分开处理。

在Xen体系结构设计中，后端设别运行的特权域被赋予一个特有的名字---隔离设备域(Isolation Device Domain, IDD)，而在实际设计中，IDD 就处在Dom0中。所有的真实硬件访问都由特权域的后端设备调用本地设备驱动 (native device drive)发起。前端设备的设计十分简单，只需要完成数据的转发操作，由于它们不是真实的设备驱动程序，所以也不用进行请求调度操作。而运行在IDD中的后端设备，可以利用Linux的现有设备驱动来完成硬件访问，需要增加的只是IO请求的桥接功能---能完成任务的分发和回送。



** ** Xen 的体系架构

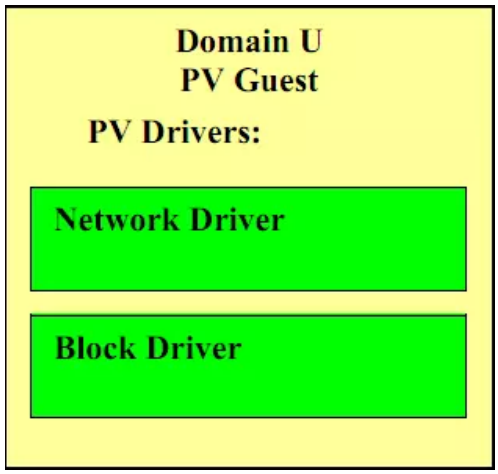
2.1.2 不同虚拟技术的运行机制

半虚拟化技术：

采用半虚拟化技术的虚拟机操作系统能够识别到自己是运行在Xen Hypervisor而非直接运行于硬件之上，并且也可以识别到在相同的机器上运行的其他虚拟机系统。而且运行的操作系统都需要进行相应的修改。

半虚拟化客户机（Domain U PV Guests）包含两个用于操作网络和磁盘的驱动程序，PV Network Driver 和PV Block Driver。

PV Network Driver负责为Domain U提供网络访问功能。PV Block Driver负责为Domain U提供磁盘操作功能。

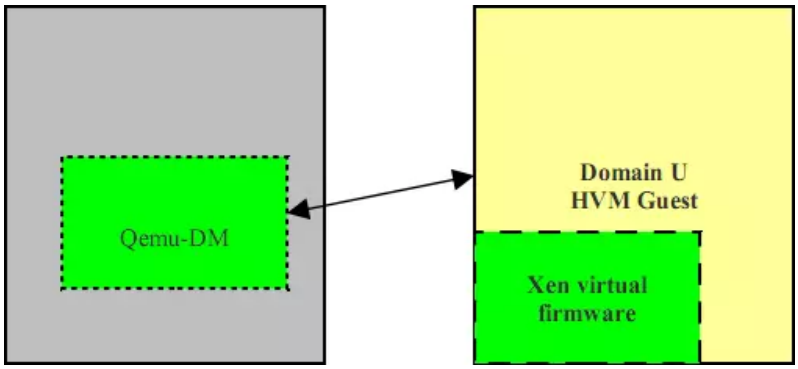


** ** 半虚拟化客户机

完全虚拟化技术：

完全虚拟化客户机（Domain U HVM Guests）运行的是标准版本的操作系统，因此其操作系统中不存在半虚拟化驱动程序（PV Driver），但是在每个完全虚拟化客户机都会在Domain 0中存在一个特殊的精灵程序，称作：Qemu-DM，Qemu-DM帮助完全虚拟化客户机（Domain U HVM Guest）获取网络和磁盘的访问操作。

完全虚拟化客户机必须和在普通硬件环境下一样进行初始化，所以需要在其中加入一个特殊的软件Xen virtual firmware，来模拟操作系统启动时所需要的BIOS。



** ** 完全虚拟化客户机

2.1.3 Domain 管理和控制

开源社区中将一系列的Linux精灵程序分类为“管理”和“控制”两大类。这些服务支撑着整个虚拟环境的管理和控制操作，并且存在于Domain 0虚拟机中。

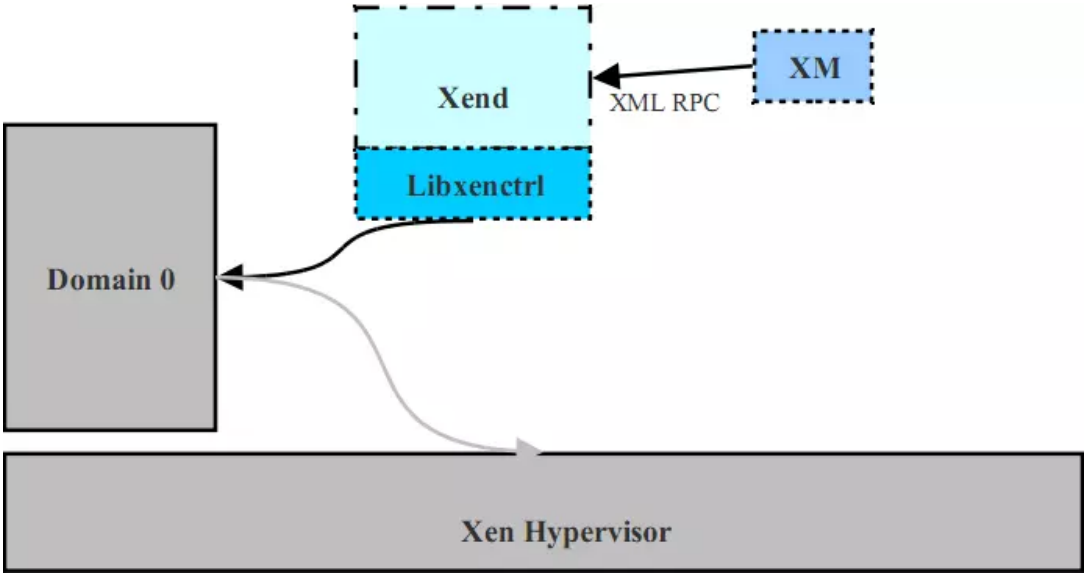
下面将对直接服务进行详细的描述。

注：为了清晰的描述Xen的运行流程，画图时将精灵程序放在Domain 0外部来描述，但事实上所有精灵程序都存在于Domain 0 之中。

2.1.3.1 Xend

Xend精灵线程是一个Python应用程序，它作为Xen环境的系统管理员。它利用Libxenctrl类库向Xen Hypervisor发出请求。

所有Xend处理的请求都是由XM工具使用XML RPC接口发送过来的。



** ** Xend运行示意图

2.1.3.2 Xm

用于将用户输入通过XML RPC接口传递到Xend中的命令行工具。

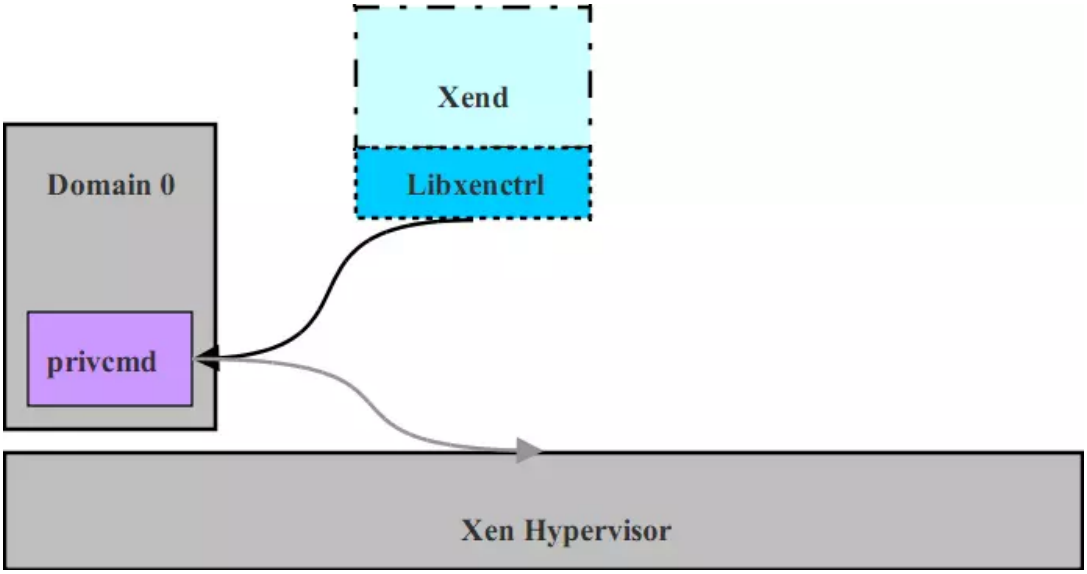
2.1.3.3 Xenstored

Xenstored精灵程序用于维护注册信息，这些信息包括内存和在连接Domain 0和所有其他Domain U之间的事件通道。Domain 0虚拟机利用这些注册信息来与系统中其他虚拟机

建立设备通道，即帮助Domain U虚拟机访问硬件资源。

2.1.3.4 Libxenctrl

Libxenctrl是C程序类库，用于让Xend具有通过Domain 0与Xen Hypervisor进行交互的能力。在Domain 0中存在一个特殊的驱动程序称作privcmd，它将请求发送给Hypervisor。



** ** Libxenctrl示意图

2.1.3.5 Qemu-DM

在Xen环境下，每个完全虚拟化虚拟机都需要拥有自己的Qemu精灵程序。Qemu-DM处理在Xen环境下完全虚拟化客户机所能允许执行的所有关于网络和磁盘请求和操作。Qemu程序必须存在于Hypervisor之外同时又需要访问网络和I/O，所以Qemu-DM必须存在于Domain 0 中（参见前面章节对Domain 0 的描述）。

未来版本的Xen中，一种新的工具Stub-DM将会提供一系列对所有完全虚拟化客户机都可用的服务，以此来替代需要在每个虚拟机上都生成一个Qemu的逻辑。

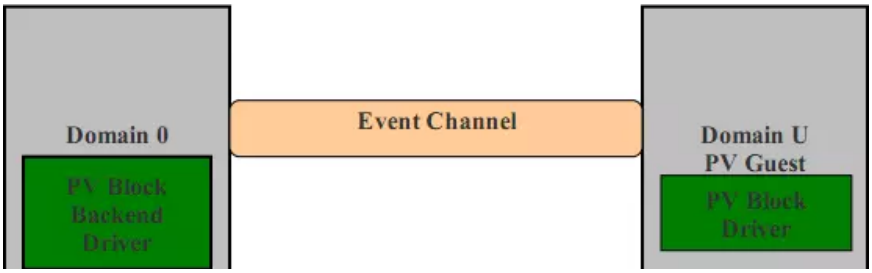
2.1.3.6 Xen Virtual Firmware

Xen Virtual Firmware是被嵌入到所有完全虚拟化客户机中的虚拟的BIOS系统，来确保所有客户操作系统在正常启动操作中接收到标准的启动指令集并提供标准的软件兼容环境。

2.1.4 半虚拟化环境下Domain 0与Domain U通信

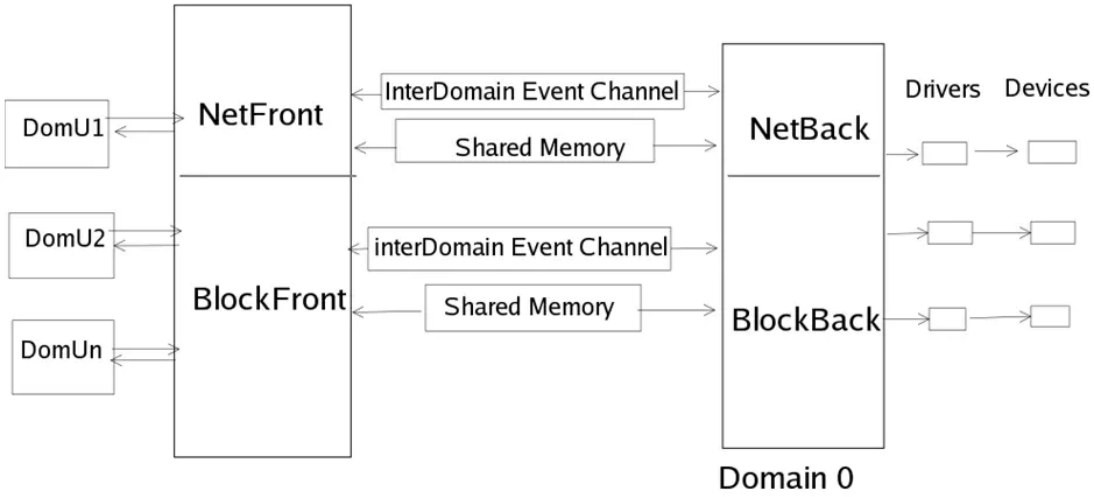
根据前几章节所述，Xen Hypervisor不负责处理网络和磁盘请求，因此半虚拟化客户机（Domain U PV）必须通过Domain 0 与Xen Hypervisor进行通信，从而完成网络和磁盘的操作请求。下面以半虚拟化客户机（Domain U PV）执行向本地磁盘写入数据为例描述Domain 0与Domain U PV的交互过程。

半虚拟化客户机（Domain U PV）的PV Block Driver接收到要向本地磁盘写入数据的请求，然后通过Xen Hypervisor将与Domain 0共享的本地内存中的数据写入到本地磁盘中。在Domain 0 和半虚拟化Domain U之间存在事件通道，这个通道允许它们之间通过存在于Xen Hypervisor内的异步中断来进行通信。Domain 0将会接收到一个来自于Xen Hypervisor的系统中断，并触发Domain 0中的Block Backend驱动程序去访问本地系统内容，并从与半虚拟化客户机的共享内存中读取适合的数据块。从共享内存中读取的数据随后被写入到本地磁盘的指定位置中。



Domain U PV**执行磁盘操作实例******

上图中所显示的事件通道是直接连接Domain 0 和Domain U PV是为了清晰和简单的描述系统是如何运行的。但事实上，事件通道（Event Channel）运行于Xen Hypervisor中，并在Xenstored中注册特定的系统中断，以此来让Domain 0 和Domain U PV能够通过本地内存快速的共享信息。



Domain 0与Domain U PV启动交互图

3 Xen的网络架构

3.1 Xen支持三种网络工作模式

1. `Bridge 安装虚拟机时默认使用 Bridge模式`
2. `Route`
3. `NAT`

各工作模式下虚拟机启动流程：

Bridge**模式**

Xend启动时流程：

1） 创建虚拟网桥 xenbr0。

2） 停止物理网卡 eth0。

3） 物理网卡 eth0 的 MAC 地址和 IP 地址被复制到虚拟网卡 veth0。

4） 物理网卡 eth0 重命名为 peth0。

5） Veth0 重命名为 eth0。

6） Peth0 的 MAC 地址更改（FE:FF:FF:FF:FF:FF），ARP 功能关闭。

7） 连接 peth0、vif0.0 到网桥 xenbr0

8） 启动 peth0、vif0.0、xenbr0

Domain U 启动时的流程：

1） vif<domainID>.0 连接到 xenbr0

2） 启动vif<domainID>.0

Route ****模式

Xend启动时的流程：

1） 开启Domain 0的IP Forward。

Domain U启动时的流程：

1） 创建 vif<domainID>.0 ， dom U eth0的IP地址被拷贝到vif<domainID>。

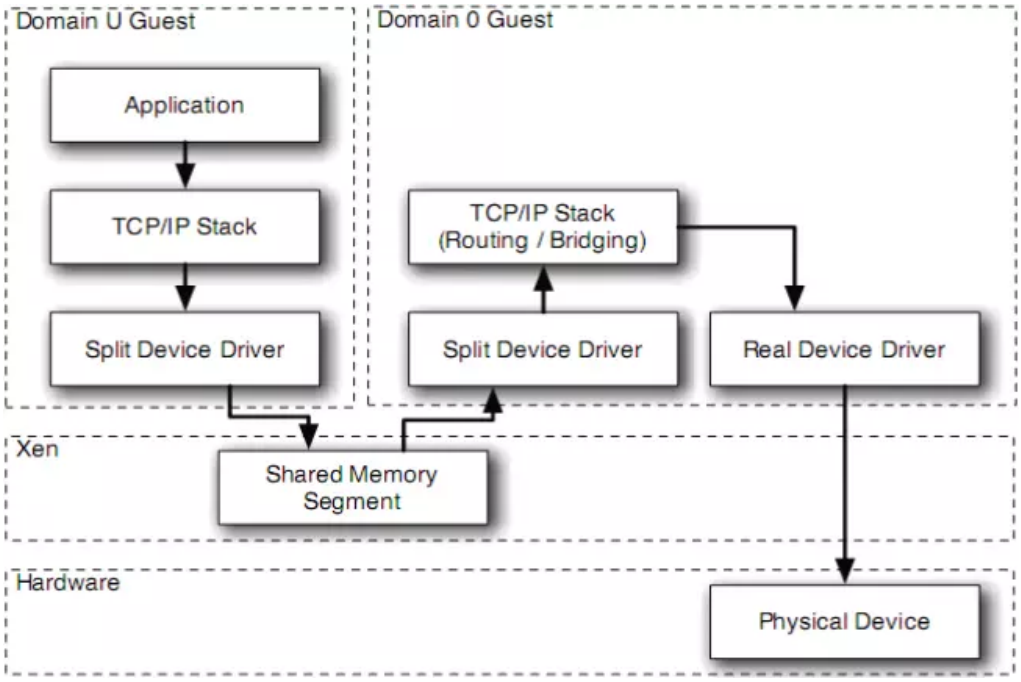
2） 启动 vif<domainID>.0。

3） 为domU的配置文件中指向虚拟接口vif.0分配的IP地址增加静态路由。

NAT**模式**

NAT 模式会使用虚拟局域网 virbr0

3.2 Xen Domain U Guests 发送数据包处理流程



Xen Domain U Guests发送数据包处理流程

3.3 Xen中虚拟网卡与物理网卡之间的关系

安装了Xen的Linux机器，在Dom 0中能看到以下几类网卡（网络接口设备 ）：

(X ， Y都为数字)

pethY

ethY

xenbrY

virbrY

vifX.Y（X为DomaiID，Y表示该虚拟网卡是该Domain的第几块虚拟网卡）

vethY （一般在Xend启动完成以后就不存在了）

小礼物走一走，来简书关注我

赞赏支持