# **virtio 简介**

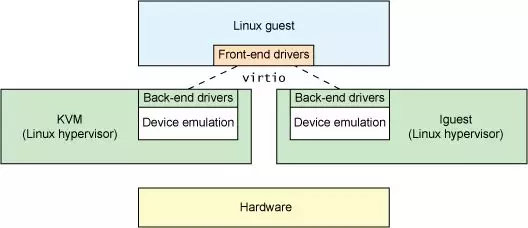
# **1. 什么是 virtio**

virtio 是一种 I/O 半虚拟化解决方案，是一套通用 I/O 设备虚拟化的程序，是对半虚拟化 Hypervisor 中的一组通用 I/O 设备的抽象。提供了一套上层应用与各 Hypervisor 虚拟化设备（KVM，Xen，VMware等）之间的通信框架和编程接口，减少跨平台所带来的兼容性问题，大大提高驱动程序开发效率。

**1.1 为什么是 virtio**

在完全虚拟化的解决方案中，guest VM 要使用底层 host 资源，需要 Hypervisor 来截获所有的请求指令，然后模拟出这些指令的行为，这样势必会带来很多性能上的开销。半虚拟化通过底层硬件辅助的方式，将部分没必要虚拟化的指令通过硬件来完成，Hypervisor 只负责完成部分指令的虚拟化，要做到这点，需要 guest 来配合，guest 完成不同设备的前端驱动程序，Hypervisor 配合 guest 完成相应的后端驱动程序，这样两者之间通过某种交互机制就可以实现高效的虚拟化过程。

由于不同 guest 前端设备其工作逻辑大同小异（如块设备、网络设备、PCI设备、balloon驱动等），单独为每个设备定义一套接口实属没有必要，而且还要考虑扩平台的兼容性问题，另外，不同后端 Hypervisor 的实现方式也大同小异（如KVM、Xen等），这个时候，就需要一套通用框架和标准接口（协议）来完成两者之间的交互过程，virtio 就是这样一套标准，它极大地解决了这些不通用的问题。



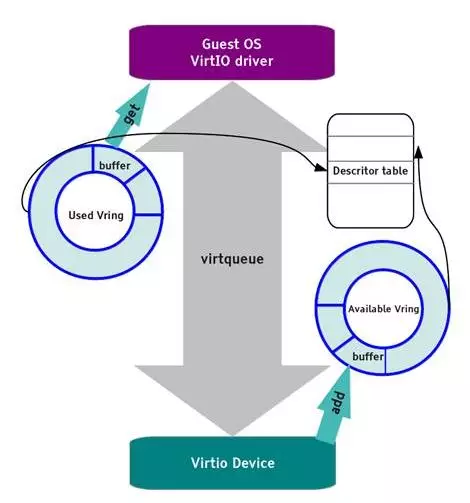
**1.2 virtio 的架构**

从总体上看，virtio 可以分为四层，包括前端 guest 中各种驱动程序模块，后端 Hypervisor （实现在Qemu上）上的处理程序模块，中间用于前后端通信的 virtio 层和 virtio-ring 层，virtio 这一层实现的是虚拟队列接口，算是前后端通信的桥梁，而 virtio-ring 则是该桥梁的具体实现，它实现了两个环形缓冲区，分别用于保存前端驱动程序和后端处理程序执行的信息。

严格来说，virtio 和 virtio-ring 可以看做是一层，virtio-ring 实现了 virtio 的具体通信机制和数据流程。或者这么理解可能更好，virtio 层属于控制层，负责前后端之间的通知机制（kick，notify）和控制流程，而 virtio-vring 则负责具体数据流转发。

**1.3 virtio 数据流交互机制**

vring 主要通过两个环形缓冲区来完成数据流的转发，如下图所示。

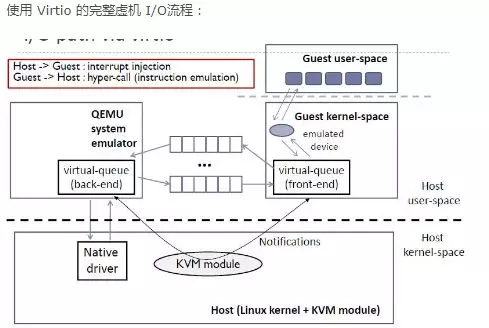


vring 包含三个部分，描述符数组 desc，可用的 available ring 和使用过的 used ring。

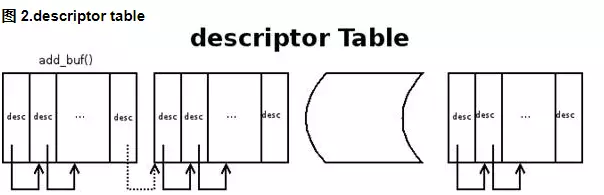
desc 用于存储一些关联的描述符，每个描述符记录一个对 buffer 的描述，available ring 则用于 guest 端表示当前有哪些描述符是可用的，而 used ring 则表示 host 端哪些描述符已经被使用。

Virtio 使用 virtqueue 来实现 I/O 机制，每个 virtqueue 就是一个承载大量数据的队列，具体使用多少个队列取决于需求，例如，virtio 网络驱动程序（virtio-net）使用两个队列（一个用于接受，另一个用于发送），而 virtio 块驱动程序（virtio-blk）仅使用一个队列。

具体的，假设 guest 要向 host 发送数据，首先，guest 通过函数 virtqueue\_add\_buf 将存有数据的 buffer 添加到 virtqueue 中，然后调用 virtqueue\_kick 函数，virtqueue\_kick 调用 virtqueue\_notify 函数，通过写入寄存器的方式来通知到 host。host 调用 virtqueue\_get\_buf 来获取 virtqueue 中收到的数据。



存放数据的 buffer 是一种分散-聚集的数组，由 desc 结构来承载，如下是一种常用的 desc 的结构：



当 guest 向 virtqueue 中写数据时，实际上是向 desc 结构指向的 buffer 中填充数据，完了会更新 available ring，然后再通知 host。

当 host 收到接收数据的通知时，首先从 desc 指向的 buffer 中找到 available ring 中添加的 buffer，映射内存，同时更新 used ring，并通知 guest 接收数据完毕。

**1.4 总结：**

virtio 是 guest 与 host 之间通信的润滑剂，提供了一套通用框架和标准接口或协议来完成两者之间的交互过程，极大地解决了各种驱动程序和不同虚拟化解决方案之间的适配问题。

virtio 抽象了一套 vring 接口来完成 guest 和 host 之间的数据收发过程，结构新颖，接口清晰。

**Virtio原理简介**

## 1. 前言

　　实现IO虚拟化主要有三种方式：全虚拟化、半虚拟化和透传。全虚拟化Guest OS不会感知到自己是虚拟机，也无需修改Guest OS，但是它的效率比较低。半虚拟化Guest OS知道自己是虚拟机，通过Frontend/Backend驱动模拟实现IO虚拟化。透传就是直接分配物理设备给VM用。Virtio是一种半虚拟化的设备抽象接口规范，在Qemu和KVM中得到了广泛使用，本文将简单介绍Virtio的基本原理。

### **Virtio SPEC**

* Virtual I/O Device (VIRTIO) Version 1.0
  + http://docs.oasis-open.org/virtio/virtio/v1.0/cs04/virtio-v1.0-cs04.html
* Virtio PCI Card Specification Version 0.9.5
  + http://ozlabs.org/~rusty/virtio-spec/virtio-0.9.5.pdf

## 2. 正文

Virtio在Guest中实现了前端驱动，在Host中实现了后端驱动，前后端之间通过Virtqueue(虚拟队列)交换数据，Host中会使用后端驱动程序模拟一个PCI设备，因此也称前端驱动为Driver，后端驱动为Device。Guest在Host OS上表示为一个Qemu的进程，Guest OS的pa实际上也属于Host OS的地址空间，因此Virtio采用的Virtqueue的方式来避免了Guest和Host间数据的复制。下面先介绍一下Virtio，更为详细的描述请阅读前言中提供的Virtio SPEC文档，那里是最准确而详细的描述，在此仅简单介绍一下后文需要用到的概念。

### **2.1 Virtio规范简介**

#### **2.1.1 Basic Facilities of a Virtio Device**

每个Virtio设备包括以下部分

* Device status field　　**设备状态字段**
* Feature bits　　 **特征位**
* Device Configuration space　　**设备配置空间**
* One or more virtqueues　　 **一个或多个virtqueues**

#### **2.1.2 virtqueues**

在virtio设备上进行批量数据传输的机制被称为virtqueue，每个设备可以拥有零个或多个virtqueue，每个virtqueue由三部分组成：

* Descriptor Table
* Available Ring
* Used Ring

每部分在客户机内存中是物理连续的，并且有不同的对齐要求，virtqueue的每个部分的内存对齐和大小要求如下表，其中Queue Size对应virtqueue中的最大buffer数，始终为2的n次幂，以特定于总线的方式指定：

| **Virtqueue Part** | **Alignment** | **Size** |
| --- | --- | --- |
| Descriptor Table | 16 | 16\*(Queue Size) |
| Available Ring | 2 | 6 + 2\*(Queue Size) |
| Used Ring | 4 | 6 + 8\*(Queue Size) |

Descriptor Table指的是Driver用于Device的缓冲区，由Queue Size个Descriptor组成。Descriptor中存有GPA的字段addr，长度字段len，可以链接next Descriptor的next指针等(形成描述符链)。如果协商了VIRTIO\_F\_INDIRECT\_DESC feature则可以使用Indirect Descriptors来增加ring的容量。

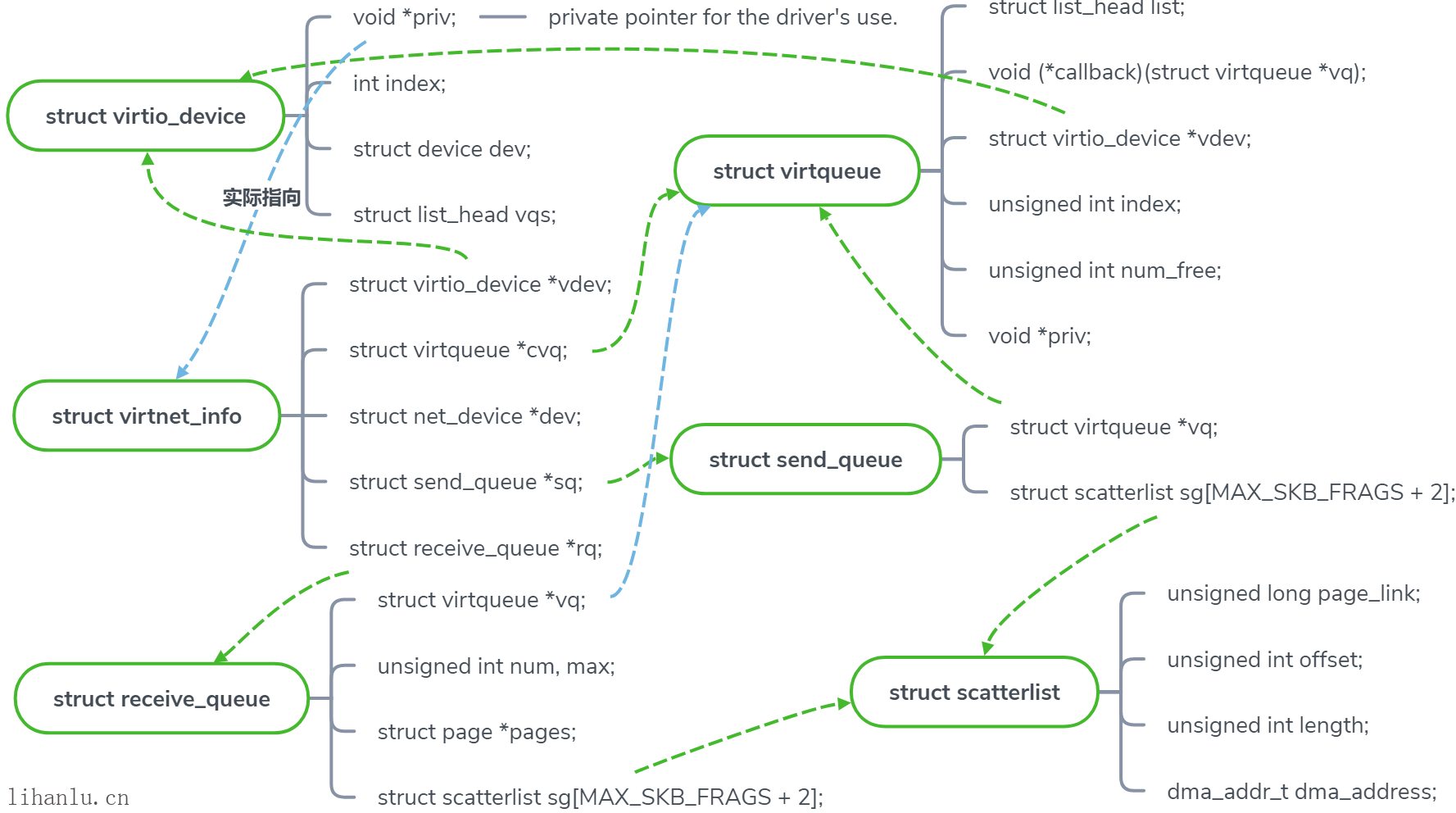
Available Ring中的每个条目是一个是描述符链的头部，它仅由Driver写入并由Device读取，Device获取Descriptor后，Descriptor对应的缓冲区可能是可读的也可能是可写的，可读的用于Driver发送数据，可写的用于接收数据。

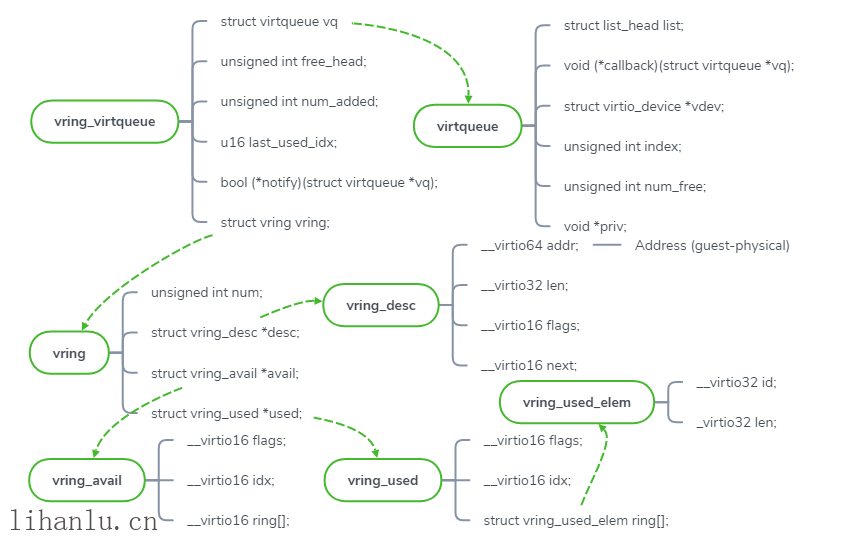
Used Ring的介绍直接贴SPEC文档中的描述上来比翻译过来更容易理解。The used ring is where the device returns buffers once it is done with them: it is only written to by the device,and read by the driver.简单来说Used Ring的作用就是Device使用完Descriptor之后，将Descriptor放入这里，通知Driver回收。

### **2.2 相关数据结构**

一些数据结构只列举了部分关键信息

如图所示，**virtnet\_info**作为virtio网络设备的私有数据将**virtio\_device**和**net\_device**链接在一起。对于virtio网络设备来说，它至少有两个**virtqueue**(如果协商了**VIRTIO\_NET\_F\_MQ**则可以创建多个队列,详见VirtIO SPEC中5.1.2 )，一个用于TX(**send\_queue**)，另一个用于RX(**receive\_queue**)，TX和RX队列中都包含了**virtqueue**和**scatterlist[]**。**virtio\_device、virtnet\_info、receive\_queue、send\_queue、virtqueue**给定任一结构体均可得到其它结构体信息。

[](https://lihanlu.bj.bcebos.com/virtio_frontend_struct.png@!lihanlu)

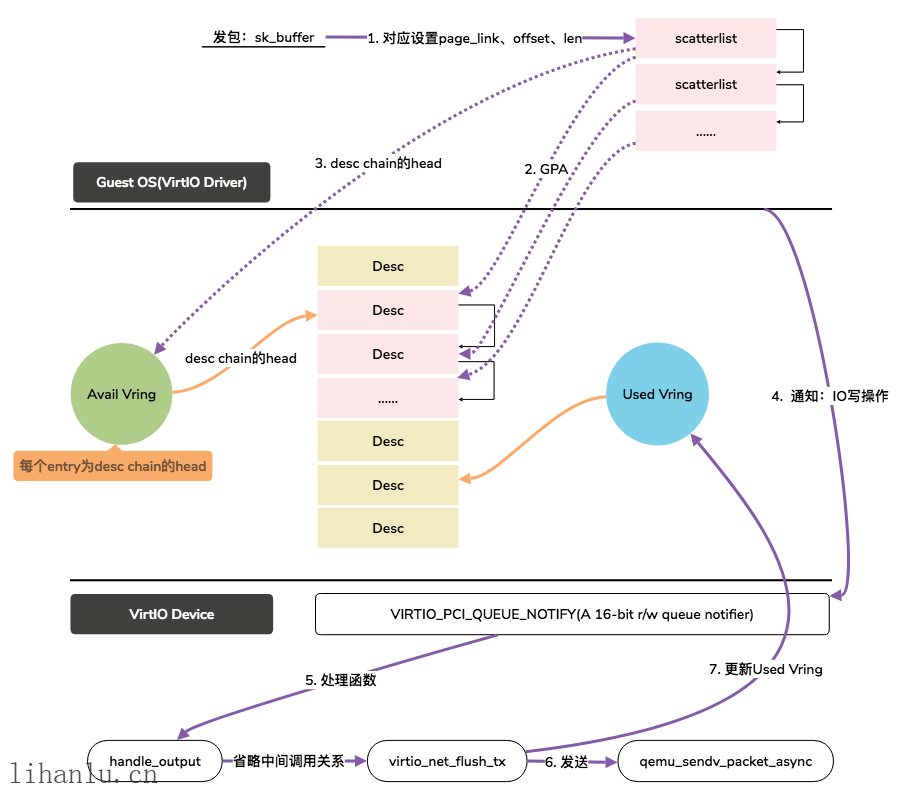
可以把**virtqueue**理解为一个接口类，而**vring\_virtqueue**作为这个接口的一个实现，**vring\_virtqueue**通过成员vq可以与上述其它struct建立联系。virtio的环形缓冲区机制是由vring来承载的，vring由三部分组成：Descriptor表(vring\_desc)，Available ring(vring\_avail)和Used ring(vring\_used)。  
[](https://lihanlu.bj.bcebos.com/vring.png@!lihanlu)

### **2.3 Vring机制简介**

在virtio设备上进行批量数据传输的机制被称为**virtqueue**。每个设备可以拥有零个或多个**virtqueue**，当Driver想要向设备发送数据时，它会填充Descriptor Table中的一项（或将几项链接在一起），并将描述符索引写入**Available Ring**中，然后它通知Device，当Device完成后，它将描述符索引写入**Used Ring**中并发送中断。(详见SPEC 2.4)

下图Virtio网络设备发包过程为例讲解上述机制，Driver将sk\_buffer填充进scatterlist table中(只是设置地址没有数据搬移)，然后通过计算得到GPA并将GPA写入Descriptor Table中，同时将Desc chain的head记录到**Available Ring**中，然后通过PIO的方式通知Device，Device发包并更新**Used Ring**。

后续会有一篇文章专门分析发包过程

[](https://lihanlu.bj.bcebos.com/virtiovring.png@!lihanlu)