#### QEMU

##### qemu和vhost-user前后端协商过程

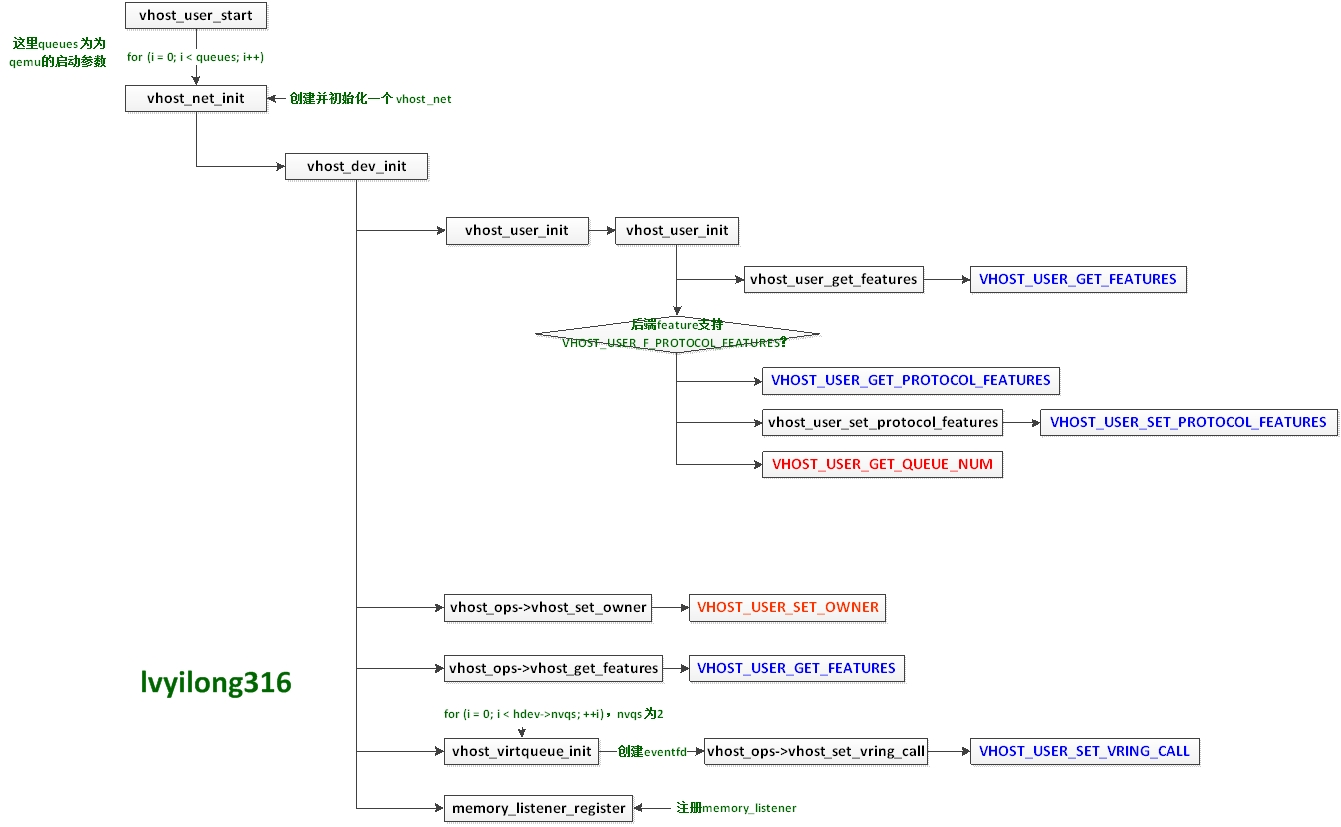
这篇文章主要从qemu的角度分析虚拟机启动前后端的协商过程。虚拟机当后端使用dpdk vhost-user时整个前后端过程可以分为三个阶段：

* qemu启动阶段，
* 前端驱动加载写VIRTIO\_PCI\_GUEST\_FEATURES寄存器，
* 前端驱动加载完成写VIRTIO\_PCI\_STATUS寄存器。

我们这里主要分析qemu和dpdk vhost\_user的交互逻辑

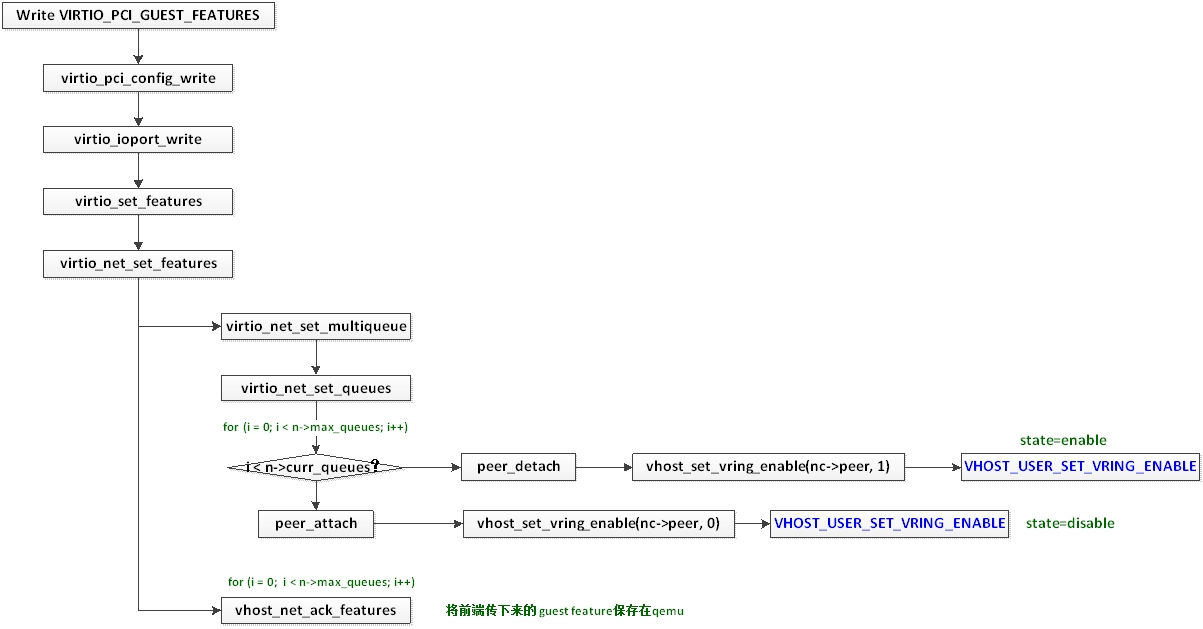
**qemu启动阶段**

qemu启动后，dpdk vhost\_user会和qemu建立vhost socket链接，连接建立成功后qemu会调用net\_vhost\_user\_event函数。



**guest驱动加载**

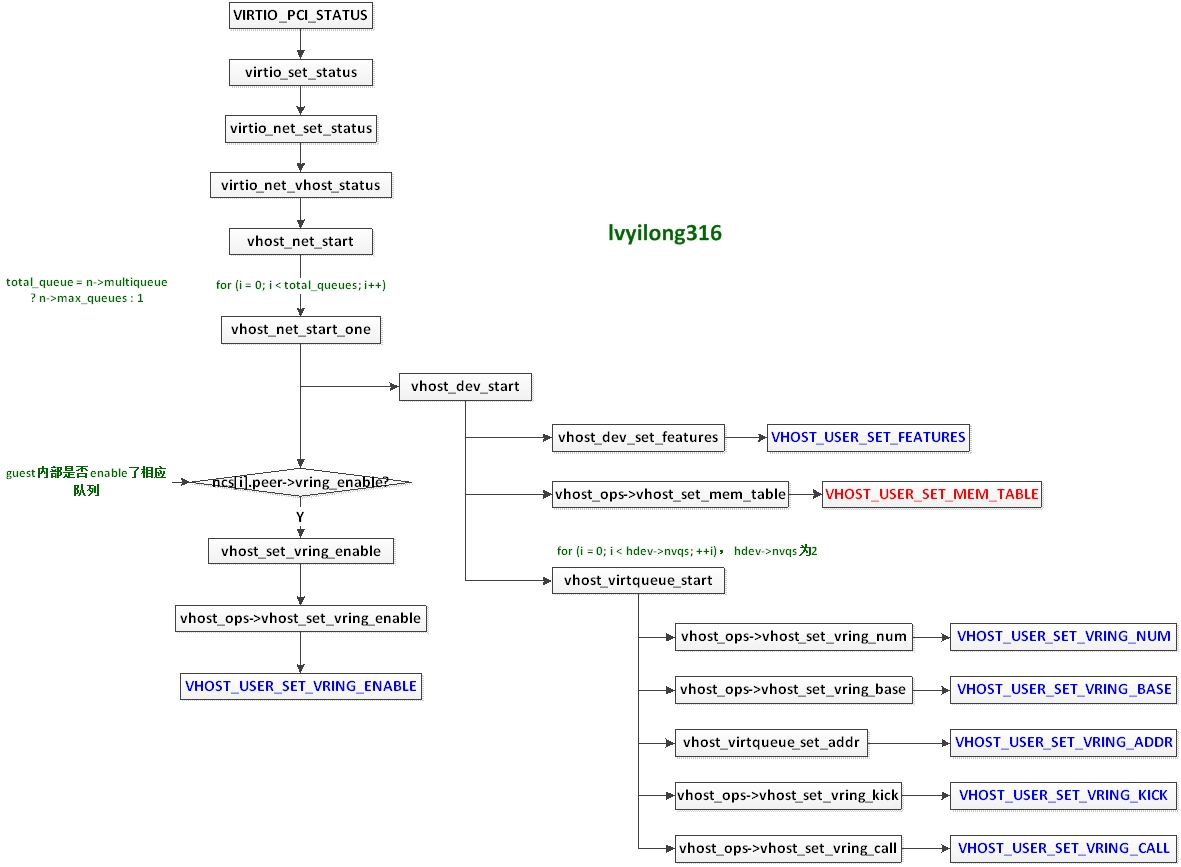
在guest启动后，加载virtio-net驱动，会写寄存器VIRTIO\_PCI\_GUEST\_FEATURES，这个写操作会被kvm捕获传递给qemu。qemu会做如下处理。



其中有两个变量比较关键，一个是max\_queues，这个就是qemu启动时后端指定的队列个数，另一个是curr\_queues，这个是当前前端enable的queue。例如启动时指定20个queue，**但一般guest启动默认只会enable一个queue，所以max\_queues为20，curr\_queues为1。**另外注意vhost\_set\_vring\_enable，最终调用的是vhost\_user\_set\_vring\_enable，这个函数会为当前queue的每个ring发送一次VHOST\_USER\_SET\_VRING\_ENABLE消息，具体在下个阶段分析。所以20个队列会为每个queue都发送两个VHOST\_USER\_SET\_VRING\_ENABLE消息，共40个，但只有小于curr\_queues时，也就是只有enable的queue才会发送state为1的消息（共两个），否则state为0。以20个queue为例，会发生20个VHOST\_USER\_SET\_VRING\_ENABLE消息，但只有第一个state为enable。

**guest驱动加载完成**

当guest中virtio-net加载完成后会写VIRTIO\_PCI\_STATUS寄存器，这个操作同样会被kvm捕获传递给qemu。qemu的相应处理逻辑如下。



其中比较关键的又是两个循环，一个是vhost\_net\_start\_one的调用。这里的循环控制变量total\_queues当guest驱动支持多队列时即为qemu启动的指定的后端队列个数，当guest不支持多队列特性的时候即为1。另一处循环就是vhost\_virtqueue\_start的调用，这个循环和第一阶段的内部循环类似，nvqs为2，即每个queue拥有的ring的个数。hdev即为vhost\_dev结构。

最后要注意的就是只有guest中enable的queue才会调用vhost\_ops->vhost\_set\_vring\_enable，也就是对于开机默认只enable 1个对列的情况只会调用一次。而vhost\_ops->vhost\_set\_vring\_enable实际上就是vhost\_user\_set\_vring\_enable。可以看到vhost\_user\_set\_vring\_enable内部是多当前queue的每个ring调用一次VHOST\_USER\_SET\_VRING\_ENABLE，所以对于一个队列enable的情况这里会发送两个VHOST\_USER\_SET\_VRING\_ENABLE。

到此为止，guest启动前后端的协商过程就完成了。如果是后端dpdk重启，vhost\_user重连过程和以上启动过程类似，区别是没有第二个阶段，因为这个时候guest内部驱动已经加载完成。

##### 待整理的点

QEMU是一个独立的虚拟化解决方案，并不依赖KVM（它本身自己可以做CPU和内存的模拟，只不过效率较低），而KVM是另一套虚拟化解决方案，对CPU进行虚拟效率较高（采用了硬件辅助虚拟化），但本身不提供其他设备的虚拟化，借用了QEMU的代码进行了定制，所以KVM方案一定要依赖QEMU

即使后来RedHat后来开发了libvirt，也只能简单的认为是个虚拟机管理工具，仍然需要通过用户空间QEMU来与KVM进行交互