the main idea of paravirtualization → make the target software(虚拟机的操作系统) "aware" of the fact that it is running inside a virtual machine.

paravirtualization has become an important topic in I/O → instead of trying to (inefficiently) emulate some hardware I/O device, we can define a new virtual-only device, and simply provide a driver for it.

Virtio networking

NAPI : https://www.wikiwand.com/en/New_API

花时间读下slides收获会很大

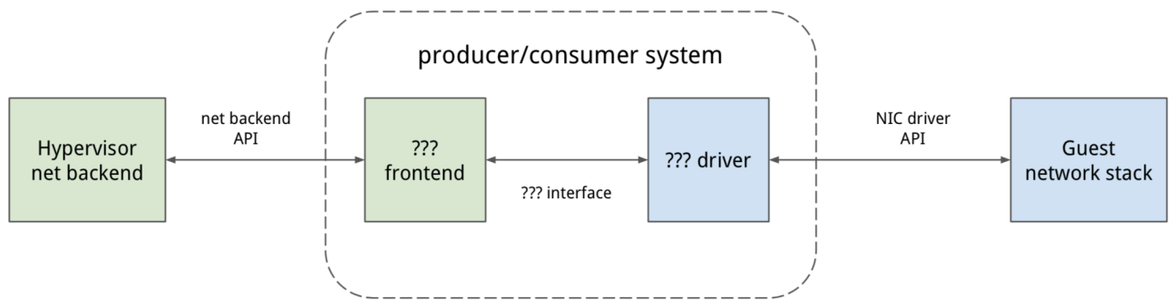
辅助资料：
1. 论文：**virtio: Towards a De-Facto Standard For Virtual I/O Devices**
结合代码阅读效果更佳

2.Slides：Virtio: An I/O virtualization framework for Linux
论文的整理

I/O paravirtualization ideas (3)

A better producer/consumer system should follow these principles:

A. Registers used only by the producer to notify the consumer that a queue is not empty anymore. They should not be used to store producer and consumer state (e.g. ring indices).
B. Interrupts used by consumer to notify producer that a queue is not full anymore.
C. Producer and consumer state should be stored in memory, so that both producer and consumer can read it without VM exit overhead.
D. Producer and consumer should be run in separate threads, so that they can work in parallel. They should both try to do as much processing as possible (polling) before going to sleep again, to amortize notification and wake-up costs.
E. Notifications should not be used when not necessary, i.e. when the other party is actively processing (not sleeping).
F. Busy waiting (uncontrolled polling) is not an acceptable general-purpose solution.

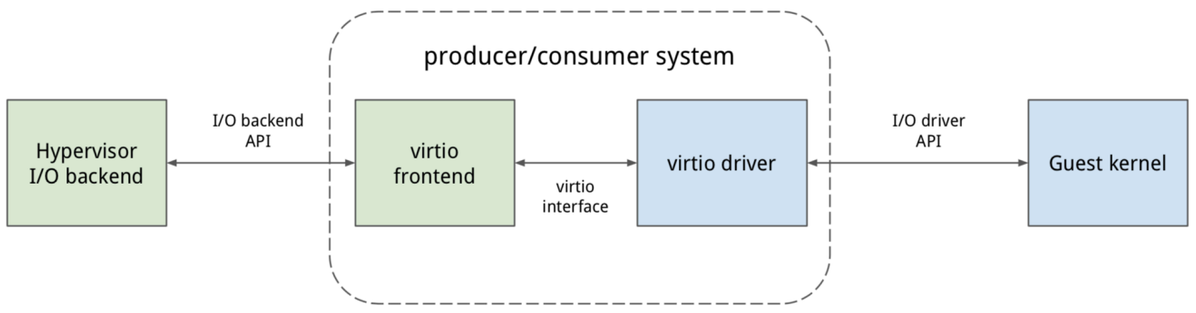


The VirtIO interface (1)

These principles are the foundation of I/O *paravirtualization*, which means that the guest device driver is aware of running in a VM, while the rest of the guest kernel is not.

To get things right, we should note that the same principles can be applied also to other forms of virtualized I/O (block storage, serial ports, …), since all forms of I/O can be seen as producer/consumer systems that exchange messages.

In any case we need to define a reasonable interface that is compatible with them. The VirtIO standard has been introduced to take this role.



The VirtIO interface (2)

VirtIO aims at high performance I/O through device paravirtualization. It's an effort to establish a standard message passing API between drivers and hypervisors. Different drivers and frontends (e.g. a network I/O and block I/O) can use the same API, which implies code reuse of the API implementation.

The task of a VirtIO driver is to convert the OS-specific representation of the message (e.g. a skb object for a Linux network driver) to the VirtIO message format, and the other way around.

The virtio-net frontend performs the same task on the hypervisor side, converting VirtIO messages from/to formats understandable to the backend.