

SR-IOV and repeated enhancement for pass through device support

Yunhong Jiang, Yaozu Dong

1. Abstract

Direct device assignment support with IOMMU DMA buffer remapping is critical for IO virtualization especially for SR-IOV devices where multiple VFs in a single physical device is designed to be assigned to different guest, which could be an ideal IO virtualization solution in future. This talk will show how we support SR-IOV devices and how we enhance pass through device support in Xen.

2. Introduction

Pass-through device support in virtual machine environment is critical for high performance. With the help of IOMMU to remap device DMA buffer, pass-through devices are getting more and more attention and are about to get adopted widely due to the merit of close to native performance while maintaining minimal CPU cycles involved in IO transaction, which is critical to high bandwidth devices such as 10Gb Ethernet card. PCI SIG IO Virtualization technology, aka IOV, especially Single Root IOV aka SR-IOV, is emerging quickly on horizon as a hardware based IO virtualization solution would improve both performance and scalability. In addition to a Physical Function (PF), an SR-IOV device can have hundreds if not thousands of Virtual Functions (VFs) associating with a PF. Each VF appears as an lightweight PCIe device being only difference with the traditional PCIe device being configuration space from software perspective, and can be assigned to guest individually with full speed of data movement without CPU intervention. PCI Manager or PCIM in short converts a lightweight PCIe device of VF to a virtual full feature of PCIe device by emulating VF's configuration space.

Xen supports PCI pass-through device with IOMMU since 3.2.0 with the help of Intel Virtualization Technology for Directed I/O aka VT-d.. However virtualizing a pass through device's PCI configuration space is still a challenge, since different devices may use different approach in hardware design and software access to the configuration space. Currently employed pure virtual configuration space appears to be insufficient to support various different PCIe devices and the SR-IOV VFs such as UHCI controller or some network devices. In meanwhile, as security and reliability concern keeps rising, how to isolate host from the guest's assigned device's failure becomes another critical issue. Further more SR-IOV VFs need to talk with Physical Function (PF) through configuration space such as Function Level Reset (FLR). and it also depends on PCI configuration space virtualization.

In this talk, we start from isolating device failure and host OS PCI device reconfiguration issue, followed by standardizing pass through device's configuration space emulation, and then go to the SR-IOV support in Xen and how we propose to implement a virtual hardware for inter VF and PF driver communication to avoid costly hardware circuit.

3. Virtualizing PCI configuration space

A failure from guest assigned device, such as SERR#, may generate host side unexpected activities, a system error from PCI device may cause an OS NMI or a BIOS SMI depending on the underlying platform, and it may lead to further host system reset on some

platforms. PCI-E AER (Advanced Error Reporting) can provide hardware support to prevent host side unexpected activity, and it is integrated in Linux 2.6.19. If the device SERR in CMD command is pinned to 1 and AER is enabled in host side, then a system error generated in guest assigned device will be logged in AER registers, in the meanwhile an AER interrupt rather than NMI or SMI will be generated in host side, i.e. dom0, to take further action per its internal policy so that the error can be limited to the guest environment only.

As test and usage of pass through devices surges, the current configuration space emulation policy which ignores guest write of configuration registers except CMD register and/or MSI/MSI-X registers, leads to failure for some of the devices. The root cause is that some of the specific configuration does not affect the real device, For example the UHCI controller troubled us for quiet a long time. We think that passing through all these registers is fair enough for devices to work properly, except those known configuration registers which includes BAR and MSI/MSI-X interrupt remapping whose updating in physical side may leads to resource confliction. As time goes on, more and more PCI configuration registers may be virtualized with more and more new feature support such as power management, a PCI capability based filter could be added in future within current framework. Meanwhile a configuration space includes not only standard PCI spec defined capabilities but also vendor specific registers and vendor specific capabilities; these are totally black boxes from the virtualization point of view. A flexible and extensible framework such as a configuration to disable or enable vendor specific configuration space could help Xen to be more flexible.

This talk will also cover other enhancement like:

- 1) **Device hot add on/removal.** Currently Xen uses initial BIOS PCI settings to setup internal device list. However, Dom0 may re-configure the system, and devices may be hot added/removed, especially consider VF creation in SR-IOV environment. In such situation, Xen need to update the device list dynamically.

PCI-E device in virtual PIIX4 platform. A PIIX4 platform has no notion of PCIe capability nor root port support, further more today's Xen HVM virtual platform doesn't expose PCIe extended capability to guest OS which may leads to failures for some devices if they set internal register through PCIe extended capabilities. We suggest to enhance Qemu/Bochs BIOS to expose PCIe extended capabilities so that guest pass through device setting of those registers is visible in physical device side to behavior correctly. We also suggest community to have a serious thinking on potential PCIe support issue in an non PCIe aware platform.

4. Support SR-IOV

SR-IOV device support with IOMMU assistance appears to be attractive enough to guarantee guest with a portion of real hardware without sacrificing CPU cycles. An SR-IOV device embeds both PFs and VFs, a PF is designed to manage hardware resources sharing among multiple VFs including BAR and global enable/disable. In some situations a PF needs to assist VF to complete some per-VF operations such as VF reset or error reporting between VF and PF. And none of these is performance critical. The major task of PCIM is to emulate the VF's configuration space so that a VF appears as a full PCIe device in guest side, and also provide a channel for VF and PF driver to communicate with each other using

mailbox/doorbell like solution. And it could be implemented by hardware or software.

Needless to say, hardware based solution needs additional circuit cost which increases proportionally with the number of VFs it supports, while software based solution can provide greener solution which can be based on either inter VM channel or a virtual hardware. However inter VM communication APIs need an up level abstract from complicated low level primitive ABIs or hypercall such as event channels, grant table and xenbus operation or global Xenstore resources. The abstraction is very difficult to have a standard API among different OSV's Xen based release. For some OSes such as Windows where DDK license is exclusive with GPL licensee on which open source VMM such as Xen and KVM are based. Further more, different VMM vendors implement different APIs such as Xen and hyper-V, which makes inter VM API based solution much hard for VF drivers. A virtual hardware can eliminate all this kind of API issues, and can help VF driver developers provideing a VMM transparent image to run across major VMMs with greatly simplified effort. Of course, this is all based on the standardization of virtual hardware, which means an additional effort to push virtual hardware specification as standard through 3rd party organizations such as PCI SIG.

In addition to PCIM effort, supporting of SR-IOV will also involve many host Linux changes such as enabling VFs in PCI subsystem, as well as some new tools such as a user level VF management tool to enable/disable/migrate of a VF or show SR-IOV status. Besides this, each SR-IOV may require a device type based configuration such as mac and vlan setting in a network SR-IOV device, which needs additional effort to push upstream and we expect to see community help in making that happen earlier.