Xen Multi-Function PCI Pass-Through

Simon Horman <simon@valinux.co.jp>

VA Linux Systems Japan K.K.

23rd October 2009

PCI Pass-Through is a method of making a PCI function available exclusively to a guest. This is as opposed to emulation, whereby a virtual device is provided to arbitrate access to a physical device.

Pass-through is typically done with the help of an IOMMU which isolates the resources of the function in question, otherwise the guest may use DMA in order to read from or write to memory which it shouldn't be able to access.

The work discussed here focused on fully-virtualised domains however pass-through may also be used with paravirtualised domains.

Multi-Function and Single-Function

Single-Function PCI devices are those that contain only one function and multi-function devices have between 2 and 8 functions. Functions are numbered from 0–7 and function 0 must always be present.

Xen 3.4.1 allows physical PCI functions to be passed-through as single-function devices. That is, regardless of its physical function number a function will appear as function-zero of a single-function device in the guest that it has been passed-through to. This means that if multiple functions of a device are passed-through they will appear as multiple devices in the guest.

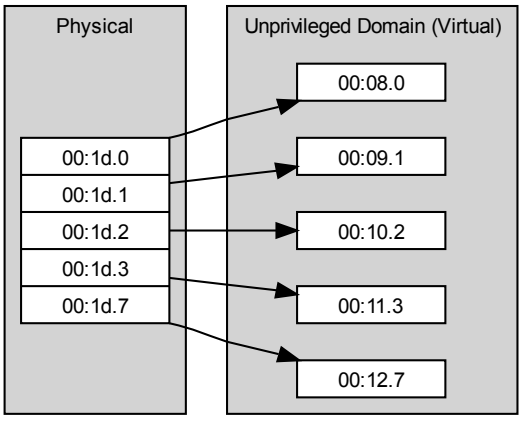

Figure 1: Single-Function Pass-Through

The aim of this work is to allow a group of passed-through PCI functions which belong to the same multi-function physical device to appear as a multi-function device when passed-through to an guest. In other words multi-function pass-through.

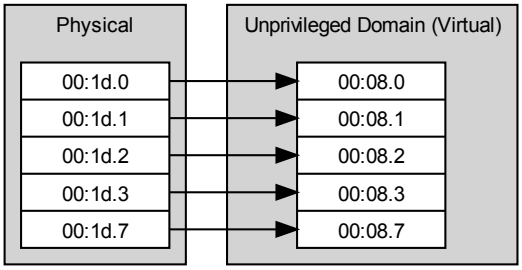

Figure 2: Multi-Function Pass-Through

Pass-Through Architecture

The two key operations relating to pass-through are attachment and detachment. Attachment of a pass-through device may occur when a domain is created. Attachment may also occur in the form of hot-plug after domain-creation. Detachment of a pass-through device may occur after a domain has been created. This is hot-unplug.

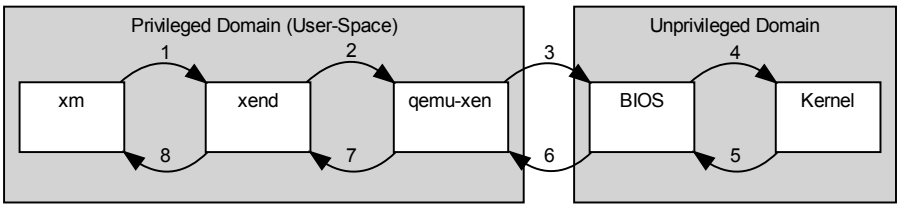

Figure 3: Event Flow During PCI Attach and Detach

Attachment and detachment are complex operations that ultimately trigger an event in the guest's kernel. The steps by by sub-system are as follows:

1. xm is a command-line tool. It accepts commands from the end-user and makes corresponding requests to xend.
2. xend is a management daemon that marshals information between the different sub-systems of a xen system. In this case it checks the pass-through commands sent by xm, reconciles them with the current state of the system, and passes them on to qemu-xend.
3. qemu-xen is used to emulate devices and control pass-through devices. Here it accepts pass-through attach or detach commands from qemu-xend. It reconfigures the xen hypervisor accordingly and triggers a corresponding ACPI event in the virtual BIOS of the target guest.
4. The BIOS in turn triggers an ACPI event in the kernel of the guest.
5. The guest's kernel receives the ACPI event, hot-plugs or unplugs the device, and sends an acknowledgement back to the BIOS.
6. The BIOS passes on an acknowledgement to qemu-xen.

7. qemu-xen updates its internal state and passes on an acknowledgement to xend.

8. xend updates its internal state and that of xenstore, and passes on an acknowledgement to xm.

Implementation Challenges

A number of challenges presented themselves while implementing multi-function pass through. While individually quite simple, their interaction made things more difficult than was originally expected.

User interaction

BDF (Bus Device Function) notation is commonly used to describe PCI devices. It consists of the bus number, colon, device number (also referred to as the slot number), decimal point, function number. For example, `00:02.0` denotes bus 0, device 2, function 0.

In order to allow the virtual slot to be specified this notation was extended by optionally appending at-sign and then the virtual slot number. For example, `00:02.0@7` specifies that the physical function at bus 0, slot 2, function 0 should be placed at virtual slot 7 in the guest.

In order to allow multi-function devices to be specified the specified this notation was further extended by allowing the function field to be a comma-delimited list of function numbers; ranges of functions denoted by the first function, a dash and then the last function; or * for all functions of the device. For example, `00:02.0,3-5@7` denotes that functions 0, 3, 4 and 5 of bus 0 slot 2 should be placed at virtual slot 7.

Device Keys

Most of the infrastructure that deals with pass-through works at the function level and has no way of dealing with the concept that devices may contain multiple devices and that hot-plug and unplug events need to be grouped and ordered such that the event corresponding to function zero comes last. In order to overcome this a per-device key is added to the data representing a function in xend. In this way all the functions of a device can be marshalled and requests can be ordered as necessary.

Mapping Physical-Functions to Virtual-Functions

The following scheme is used to map physical-functions to virtual-functions.

- The numerically lowest physical function is mapped to virtual-function zero.
- All the other functions are identity mapped.

The first step is necessary in order to ensure that the virtual-device includes function 0, which is required. It is also possible to specify the virtual slots explicitly, however if the resulting virtual-device does not include function 0 then the operation will fail.

Conclusion

The current implementation of multi-function pass-through is an incremental improvement, allowing physical multi-function devices to appear as multi-function devices in guests. This functionality is scheduled to be included in Xen 3.5 and is already available in the xen-unstable development tree.

Moving forward it would be interesting to examine combining functions from different physical devices into a virtual multi-function device. However it is unknown what hardware-level problems this may present.