Status Update of PVUSB

Noboru Iwamatsu
n_iwamatsu@jp.fujitsu.com
FUJITSU LABORATORIES LTD.

Xen Summit at Asia Nov. 18-19, 2009
Contents

- Introduction
  - Background and history

- Tutorial: using PVUSB
  - PVUSB overview
  - Usage of xm commands
  - Demo Video
  - List of tested devices

- Updates of PVUSB (since first posted)
  - Main changes
  - Performance consideration
  - New device probe mechanism with xm/xend assist

- Conclusion
Background:

- In client virtualization, special Service-VMs work in the background of User-VM, and provide various functions.
- Both User-VM and Service-VMs require using the USB devices at the same time by the same USB host controller.
Brief History

Before PVUSB

The options for using USB device:
- Qemu-dm UHCI emulation (USB1.1 only, slow)
- PCI pass-through (the whole host controller is assigned to a single domain)

History of PVUSB

Aug. 2008  Started development
Nov. 2008  Just started working!
           Xen Summit @Tokyo 2008
Feb. 2009  Achieved good performance!
           Xen Summit @Oracle 2009
Mar. 2009  Patches posted (drivers only, 1st release)
           PVUSB was merged into linux-2.6.18-xen.hg tree!
May. 2009  Xen 3.4.0 released (includes PVUSB)
Oct. 2009  Update patches posted (2nd release)
Nov. 2009  Experimental xm/xend integration (to be posted)
PVUSB overview: transferring urbs

- usbfront (Host controller driver) sends the enqueued urb to usback.
- usback (USB Device driver) re-creates and transfers urb to the actual device.

**DOM0 kernel-space**

- USB Core
- Host controller drivers
- usback (USB Device driver)

**DOMU kernel-space**

- USB Core
- USB Device drivers
- usbfront (Host controller driver)

**Xen hypervisor**

- IO RING

**urb (USB Request Block structure):**
In Linux kernel, all USB drivers communicate with urb. (described in include/linux/usb.h).
PVUSB overview: hotplug notification

- PVUSB associates physical device path to virtual root port.
- PVUSB supports:
  - Virtual hotplug: manually attach/detach the device already connected.
  - Real hotplug: automatically attach/detach the device with actual hotplug.
xm: Create/Destroy Virtual Host Controller

- `xm usb-hc-create <Domain> <USBSpecVer> <NumberOfPorts>`

  Create a domain’s new virtual host controller.

  `<Domain>` : domain name
  `<USBSpecVer>` : 2: USB2.0, 1: USB1.1
  `<NumberOfPorts>` : number of root ports (1 to 16)

  Example: (Create USB2.0 host controller that has 8 ports for guest_A)

  ```
  # xm usb-hc-create guest_A 2 8
  ```

- `xm usb-hc-destroy <Domain> <DevId>`

  Destroy a domain’s virtual USB host controller.

  `<Domain>` : domain name
  `<DevId>` : Device Id of virtual host controller

  Example: (Destroy guest_A’s USB host controller the DevId is 0)

  ```
  # xm usb-hc-destroy guest_A 0
  ```
xm: Attach/Detach the USB devices to guests

- **xm usb-attach** `<Domain> <DevId> <PortNumber> <BusId>`

  Attach a new USB physical bus to domain’s virtual port.

  `<Domain>` : domain name  
  `<DevId>` : Device Id of virtual host controller  
  `<PortNumber>` : Port number of the virtual host controller  
  `<BusId>` : Physical USB device path

  Example:

  ```bash
  # xm usb-attach guest_A 0 3 1-2
  ```

  - If a USB device is already on that bus, the device is seized by usbback and connected to the guest’s port.

- **xm usb-detach** `<Domain> <DevId> <PortNumber>`

  Detach a USB physical bus from domain’s virtual port.

  Example:

  ```bash
  # xm usb-detach guest_A 0 3
  ```

  - If a USB device is already connected to the guest’s port, the device is disconnected from the guest.
xm: List assignable USB devices

- **xm usb-list-assignable-devices**

  **List all the assignable usb devices.**

  **Example:**

  ```
  # xm usb-list-assignable-devices
  1-1 : ID 0411:0182 BUFFALO SHD-PE
  2-4.3.3 : ID 04bb:093a I-O DATA DEVICE,INC. I-O DATA ETX2-US2
  2-4.4 : ID 0411:0098 BUFFALO USB Flash Disk
  2-5 : ID 17e9:0129 DisplayLink REX-USBDVI2
  3-2 : ID 04bb:0a0e Prolific Technology Inc. USB-Serial Controller
  6-1.1 : ID 0430:0009 FujitsuTakamisawaComponent USBkeyboard
  6-1.2 : ID 046d:c404 Logitech Trackball
  ```

- All the devices excluding usbhubs and already assigned ones are shown like lsusb output.
- You can easily attach the device with ‘xm usb-attach’ command!
xm: List assigned USB devices

- `xm usb-list <Domain>`

List domain’s attachment state of all virtual ports.

`<Domain>` : domain name

Example:

```
# xm usb-list guest_A
Idx  BE  state  usb-ver  BE-path
 0    0    4      USB2.0  /local/domain/0/backend/vusb/1/0
port 1: 1-1
port 2: 1-2 [ID 0411:0182 BUFFALO SHD-PE]
port 3: 1-3
port 4: 1-4
port 5: 2-4.3.3 [ID 04bb:093a I-O DATA DEVICE,INC. I-O DATA ETX2-US2]
port 6:
port 7:
port 8:
```
Demo (Video)
domain configuration file syntax

- `xm create` command supports creating virtual USB host controller and port-rules at boot time of the guest.

- Options of PVUSB for a domain configuration file (live in `/etc/xen` by default) is followings:
  - `vusb`: enable PVUSB
  - `usbver`: USB Spec Version (1|2 default:2)
  - `numports`: number of root ports
  - `port_1` to `port_16`: physical busid to be assigned

Example setting:

```
vusb = ['usbver=2, numports=8, port_1=1-1, port_2=1-2']
```
## Tested Devices (USB2.0)

<table>
<thead>
<tr>
<th>Device Type</th>
<th>Name</th>
<th>Manufacturer</th>
<th>Transfer Type</th>
<th>Driver</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash drive</td>
<td>RUF2-R2GS</td>
<td>Buffalo</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>Flash drive</td>
<td>RUF-C1G/U2</td>
<td>Buffalo</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>Flash drive</td>
<td>DataTraveler DTI/1G</td>
<td>Kingston</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>HDD</td>
<td>HDCN-U500</td>
<td>IO DATA</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>SSD</td>
<td>SHD-PE32G</td>
<td>Buffalo</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>DVD Drive</td>
<td>DVSM-PL58U2/S</td>
<td>Buffalo</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>Memory Card Reader/Writer</td>
<td>BSCRA38U2</td>
<td>Buffalo</td>
<td>Bulk</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>Ethernet (100Base-TX)</td>
<td>ETX2-US2</td>
<td>IO DATA</td>
<td>Bulk/Intr</td>
<td>pegasus</td>
<td>✔️</td>
</tr>
<tr>
<td>Ethernet (1000Base-T)</td>
<td>ETG2-US2</td>
<td>IO DATA</td>
<td>Bulk/Intr</td>
<td>asix</td>
<td>✗</td>
</tr>
<tr>
<td>VGA-adaptor</td>
<td>REX-USBDVI</td>
<td>RATOC</td>
<td>Bulk/Intr</td>
<td>libdlo-0.1.2</td>
<td>✔️</td>
</tr>
<tr>
<td>VGA-adaptor</td>
<td>REX-USBDVI2</td>
<td>RATOC</td>
<td>Bulk/Intr</td>
<td>libdlo-0.1.2</td>
<td>✔️</td>
</tr>
</tbody>
</table>

- ✔️ Works
- 🔄 Works with issues
- ✗ Not work
<table>
<thead>
<tr>
<th>Device Type</th>
<th>Name</th>
<th>Manufacturer</th>
<th>Transfer Type</th>
<th>Driver</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>FMV-KB333</td>
<td>Fujitsu</td>
<td>Intr</td>
<td>usbhid</td>
<td>✔️</td>
</tr>
<tr>
<td>Keyboard</td>
<td>FKB-108-EU</td>
<td>FILCO</td>
<td>Intr</td>
<td>usbhid</td>
<td>✔️</td>
</tr>
<tr>
<td>Mouse</td>
<td>Cordless Notebook Mouse</td>
<td>Logitech</td>
<td>Intr</td>
<td>usbhid</td>
<td>✔️</td>
</tr>
<tr>
<td>Mouse</td>
<td>Optical Mouse USB</td>
<td>Logitech</td>
<td>Intr</td>
<td>usbhid</td>
<td>✔️</td>
</tr>
<tr>
<td>FDD</td>
<td>USB-FDU</td>
<td>Y-E DATA</td>
<td>Bulk/Intr</td>
<td>usb-storage</td>
<td>✔️</td>
</tr>
<tr>
<td>Serial</td>
<td>USB-RSAQ5</td>
<td>IO DATA</td>
<td>Bulk/Intr</td>
<td>pl2303</td>
<td>✔️</td>
</tr>
<tr>
<td>Webcam</td>
<td>WebCam 3 USB</td>
<td>Creative Labs</td>
<td>Isoc</td>
<td>ov511</td>
<td>✔️</td>
</tr>
<tr>
<td>Webcam</td>
<td>WebCam NX pro</td>
<td>Creative Labs</td>
<td>Isoc</td>
<td>gspca</td>
<td>✔️</td>
</tr>
<tr>
<td>Sound</td>
<td>MM-SP2UBK</td>
<td>SANWA SUPPLY</td>
<td>Isoc</td>
<td>snd_usb_audio</td>
<td>✔️</td>
</tr>
</tbody>
</table>

✔️ Works
⚠️ Works with issues
❌ Not work
Main changes since first release

- Changes in the drivers (all posted and merged):
  
  **Design changes:**
  - Acceptable urb buffer size was increased from 40kB to 64kB for Windows guests. *(This affects the performance.)*
  - Introduced new RING for hotplug notification.
  - Port-rule setting interface was moved to xenstore.
  - Xenbus state flow changed.

  **Added features:**
  - Host controller version (USB2.0 or USB1.1) became selectable.
  - Urb dequeueing function completed.

- New management tools:
  - Experimental xm/xend integrations
  - New hotplug mechanism with xend assist
Urbs and buffers are mapped to RING-request and sent to backend.

Size of RING and RING-request are fixed. So, acceptable urb buffer size is limited.

Urb buffer size and RING_SIZE are trade-off. When the acceptable buffer size was increased from 40kB to 64kB, RING_SIZE was decreased from 32 to 16.

How RING_SIZE change will affect the performance?
Performance impact of RING_SIZE change(1)

USB Mass Storage Read/Write Throughput (MB/s)

- USB Flash Drive: Buffalo RFU2-R2GS
- USB HDD: IODATA HDCN-U500
- USB SSD: SHD-PE32G

Tested with:
- Xen: 3.4.2-rc3, x86_64
- Kernel: linux-2.6.18-xen (c/s 941), x86_64
- CPU: Core 2 Qued Q9450 2.66GHz

PVUSB (RING_SIZE=16)
- Read: 13.7
- Write: 29.6

PVUSB (RING_SIZE=32)
- Read: 15.1
- Write: 26.2

Dom0 native
- Read: 22.9
- Write: 25.8

USB Mass Storage Read/Write Throughput (MB/s)
‘RING_SIZE=16’ significantly degrades the performance

<table>
<thead>
<tr>
<th>RING_SIZE</th>
<th>Performance compared to native</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>16</td>
<td>46.3% - 47.5%</td>
</tr>
<tr>
<td>32</td>
<td>94.6% - 95.5%</td>
</tr>
</tbody>
</table>

In ‘RING_SIZE=16’ condition, RING frequently falls into RING_FULL!

- After reading 512MB block, RING_FULL has happened 65423 times while transferring 148331 urbs.

```
# cat /sys/class/usb_host/usb_host1/statistics
bus xen, device vusb-0
Xen USB2.0 Virtual Host Controller
xenhcd, hcd state 1
complete 148331 unlink 0 ring_full 65423
```
Performance impact of RING_SIZE change(3)

- 64kB buffer is required for Windows guest, though, Linux guest usually use much less buffer.
  - USB storage driver of linux always use 4kB buffer.

- To improve the performance, multi-page RING and efficient use of RING are needed.

- We will solve this problem by using netchannel2’s new RING protocol, however, if you don’t like the default value “64kB”, you can modify it by yourself!

- Edit the value in drivers/include/xen/io/usbif.h:

  ```
  #define USBIF_MAX_SEGMENTS_PER_REQUEST (16)
  ```

- (acceptable buffer size) = 4kB * USBIF_MAX_SEGMENTS_PER_REQUEST

- Relation between RING_SIZE and USBIF_MAX_SEGMENTS_PER_REQUEST:

<table>
<thead>
<tr>
<th>USBIF_MAX_SEGMENTS_PER_REQUEST</th>
<th>1</th>
<th>2 - 5</th>
<th>6 - 13</th>
<th>14 - 29</th>
<th>30 -</th>
</tr>
</thead>
<tbody>
<tr>
<td>RING_SIZE</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>
 Probe/Disconnect problems in usbback driver

Probes function in USB device driver

- USB device driver registers a list of devices that this driver can accept, and probe function is called when the device is added.
- Usbback must handle any devices, so usbback registers the wild-card list. According to this, probe function is called every time when a device is added.

```c
/* Device Id table in drivers/xen/usbback/usbstub.c */
static struct usb_device_id usbstub_table[] = {
    { .driver_info = 1 }, /* wildcard, see usb_match_id() */
    
};
```

Problems cannot be resolved by driver alone:

- Probe function is called once when a device is added. Devices already used by other drivers cannot be claimed.
- Disconnect function is called once when a device is removed. To detach a device from guest, you have to detach the device actually.
- Wild-card sometimes fails. If two or more drivers claim the same device, which is given priority depends on USB core.
Assigning USB devices with user-space assist

- **bind/unbind drivers from devices manually through sysfs**
  - To unbind a device from driver, simply write the bus id of the device interface:
    ```
    # echo -n "1-1:1.0" > /sys/bus/usb/drivers/usb-storage/unbind
    ```
  - To bind a device to a driver, simply write the bus id of the device interface:
    ```
    # echo -n "1-1:1.0" > /sys/bus/usb/drivers/usbback/bind
    ```

- **Implementation of \texttt{xm usb-attach} / \texttt{xm usb-detach}**

  ![Diagram of xm usb-attach flowchart]

  ![Diagram of xm usb-detach flowchart]
- Udev.rule for watching USB device and reporting to xend added.
- When a device that should be claimed to usbback is grabbed by other driver, xend unbinds from it and re-binds it to usbback.
TODO & Future Work

**TODO**

- **xm/xend code:**
  - Add Xen-API support
  - Cleanups and bugfixes
- **paravirt_ops support**
- **Performance improvement**
  - Netchannel2 support
  - Urb re-using in usbback
- **USB2.0 webcams support**

**Future Work**

- **USB3.0 support**
  - Hoping Xen become the first to meet USB3.0 in the virtualization world.
Conclusion

- PVUSB reached at the level any user can use:
  - User-friendly interface (maybe)
  - Hotplug working correctly

- PVUSB will provide:
  - Flexible and high-performance USB device assignment.
  - Various devices work in guest domains.
    - Keyboard, Mouse, FDD, Serial, Flash drive, HDD, SSD, DVD-ROM drive, Ethernet, Sound, Webcam, VGA-adapter
THE POSSIBILITIES ARE INFINITE