




ORACLE[®]

Virtualization & IB

March 3rd, 2014

Michel Riviere & Pierre Orzechowski



The following is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

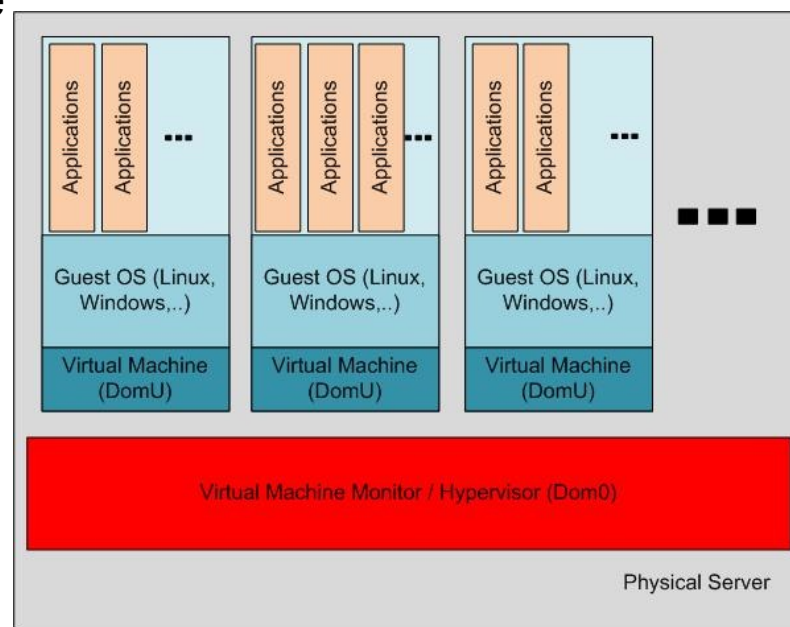


Agenda

- Why Infiniband Virtualization ?
- I/O Virtualization models
 - Software based sharing
 - Hardware based I/O virtualization with SR-IOV
- Case study with OVM/Xen and Oracle Sun Fire systems
 - Implementation
 - Network configuration
 - VM configuration
 - Issues pending and early performance results
- Conclusions and QA

Motivation behind Virtualization

- A physical server can be virtualized in multiple logical servers
- Each logical server or VM (Virtual Machine) run on top of VMM (Virtual Machine Monitor) or Hypervisor
- Each VM can run its own OS and share or own it's own physical resources (CPU, Memory, I/O..)
- VMM controls the resources VM's resources and provides services like VM manage LiveMigration of VM, dynamic resources allocation and load balancing
- This is useful for cloud environment, consolidation and over all improve the physical server utilization and total cost of ownership.





Infiniband Virtualization Model

Software based I/O virtualization

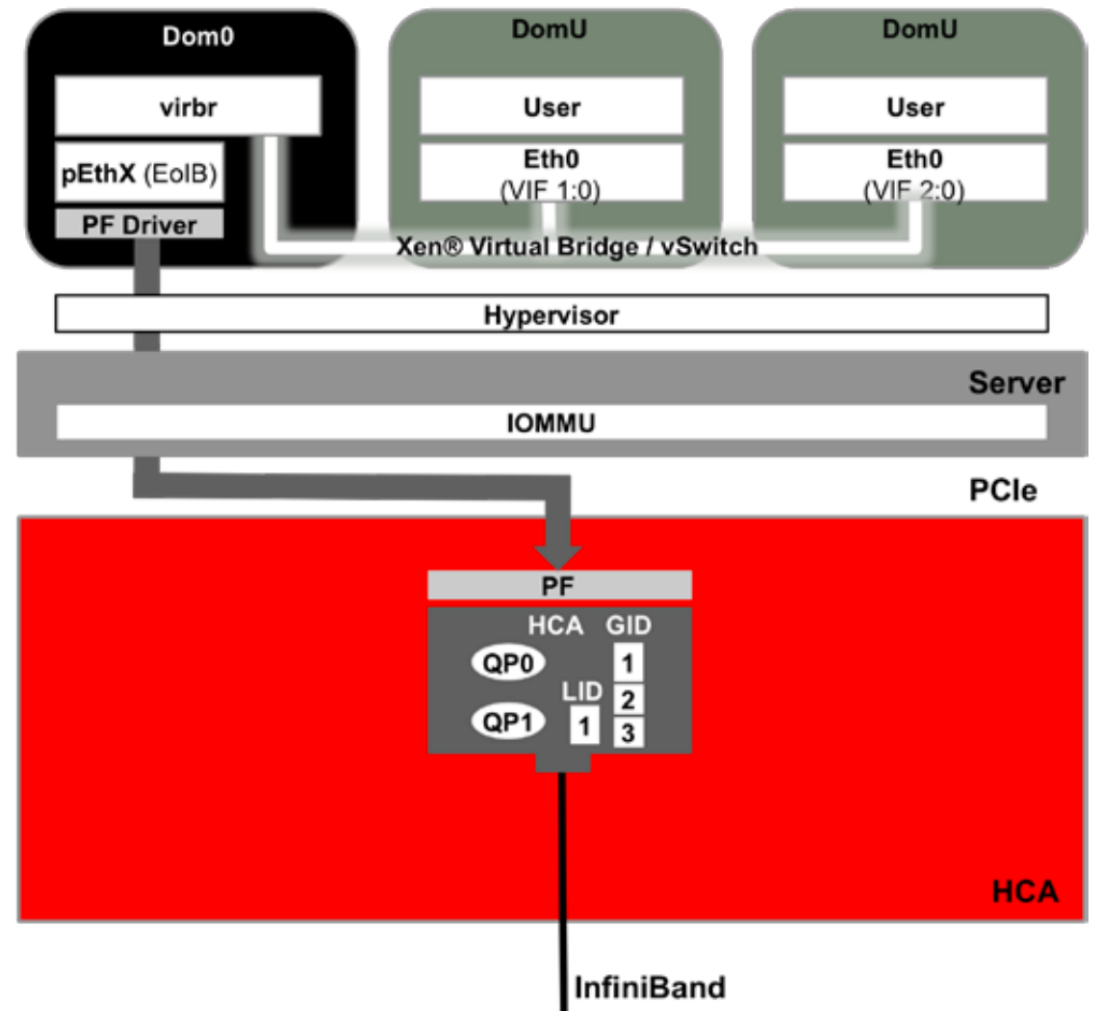
- Very easy to set in motion, transparent to existing VM environments, only solution to rip all all benefits from Virtualization (e.g. Load Balancing, LiveMigration, PV VM,...)
- Performance impact on I/O latency and bandwidth

Hardware-based I/O virtualization (bypass)

- I/O performance is significantly improved
- CPU usage is reduced (better server utilization)
- Virtualization features not always available (LiveMigration, PV VM, PoD..)

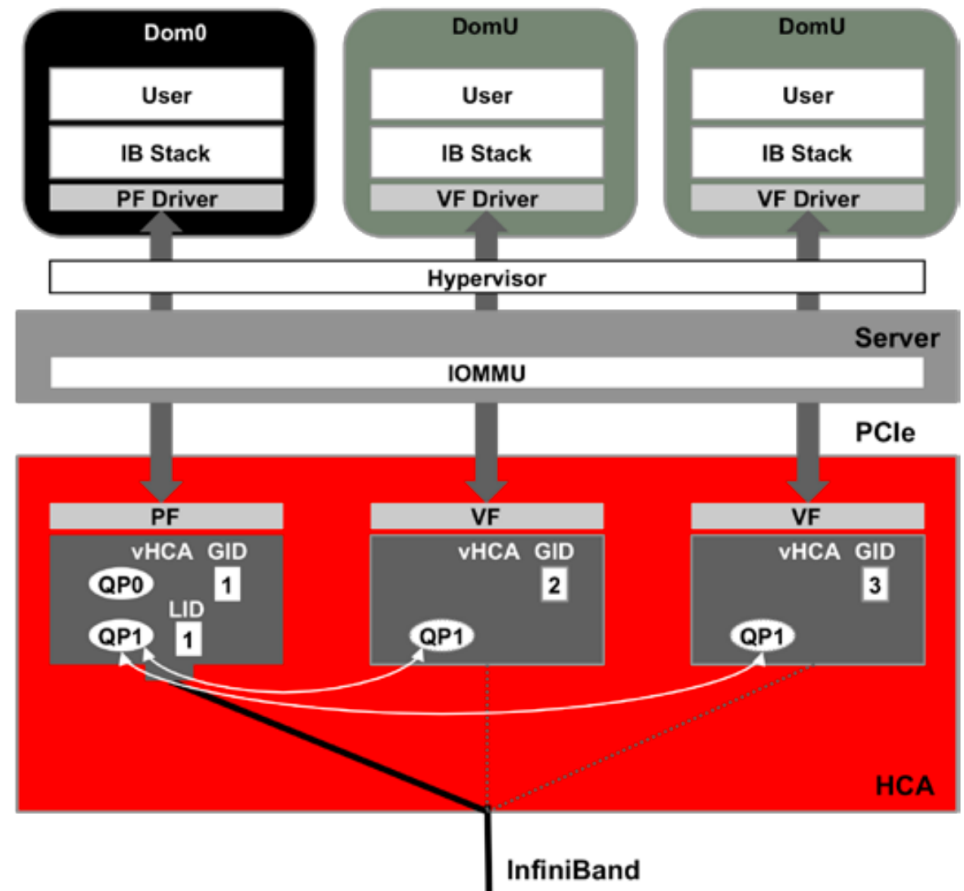
Software based I/O virtualization in Xen

- Virtual Bridging allow VM to be fabric agnostic
- VM relies on an emulated, generic Ethernet NIC for I/O
- Emulated NIC communicates with Infiniband HCA through a virtual Bridge in Dom0
- Infiniband driver stack installed only in Dom0
- Application in DomU can not leverage native Infiniband features or RDMA-enabled protocols.



SR-IOV Virtualization with Xen

- SR-IOV exposes VF enabling Infiniband access for VM
- PF is hosted in Dom0 and is responsible for dynamic allocation of IB resources (PKeys, QP, CQ, memory region,...) to the VFs
- PF owns QP0 and virtualize QP1 to make it available to all VMs
- Each VM has the Infiniband stack instantiated and can leverage RDMA-enabled protocol for enhanced performance.



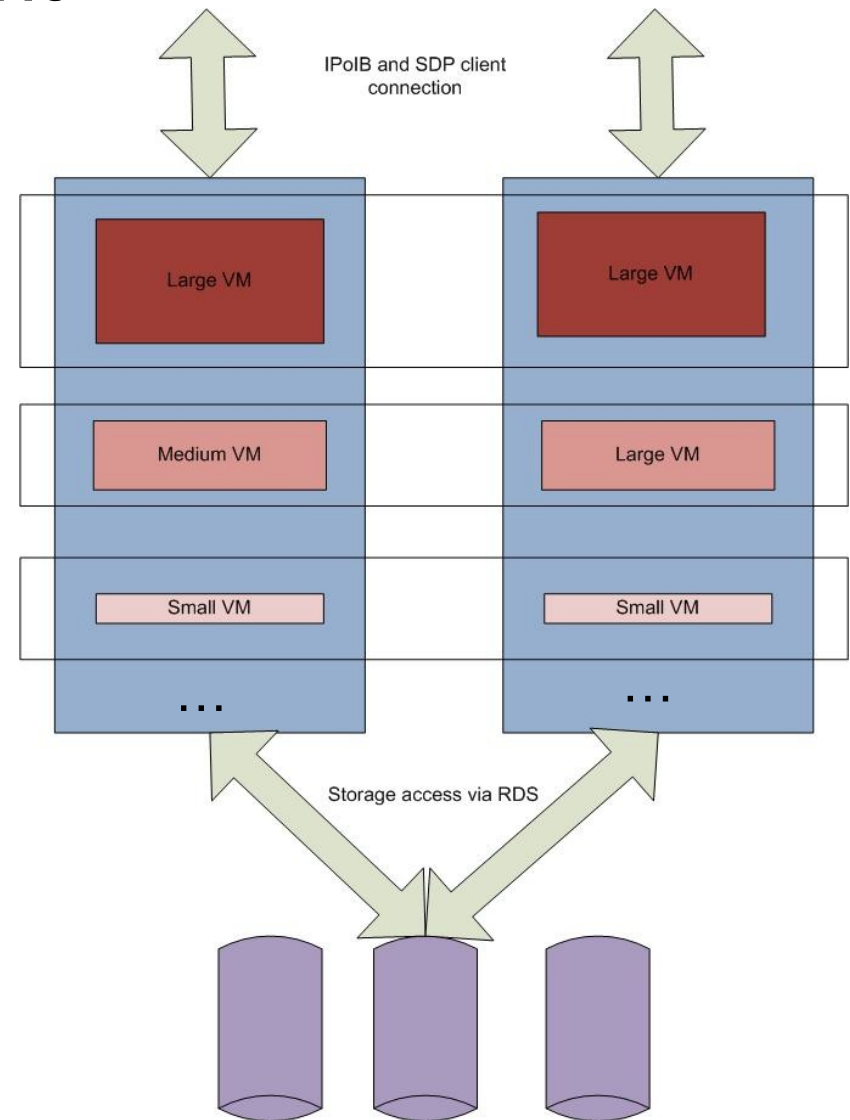


SR-IOV Infiniband constraints

- A VF configuration space provides access to registers to perform I/O only (e.g. Access only DMA channels and related registers).
- The H/W related configuration changes can only be performed via the PF. VF driver needs to interact with PF driver to perform VF's operations. PF driver is responsible to ensure a VF does not impact other VFs or PF in any way.
- This restrict the management IB commands available from the VMs (SM commands only available from Dom0).
- The view of the Infiniband network is different from the privileged VM (Dom0) and the other VMs (DomUs).
- Some Virtualization features are not supported (save/restore, Live Migration)

Case study environment

- Create Database clusters between 2 (8 sockets) servers
- Each server host 8 VMs
- Each cluster contains 2 VMs
- 3 types of VM:
 - Large: 20 cores, 4VFs, 512GB
 - Medium: 10 cores, 2VFs, 256GB
 - Small: 5 cores, 1VF, 128GB
- Clients connections use SDP and IPoIB
- Storage access relies on RDS





Case study goals

- Use Active-Active and Active-Passive IB bonding in each VM
- Network QoS to provide automatic and transparent prioritization of latency sensitive messages
- Infiniband Partitioning for isolation between the virtual DB clusters
- Validate client connections to DB (SDP, IPoIB) and storage (RDS)
- Validate the full stack for both stability and performance.

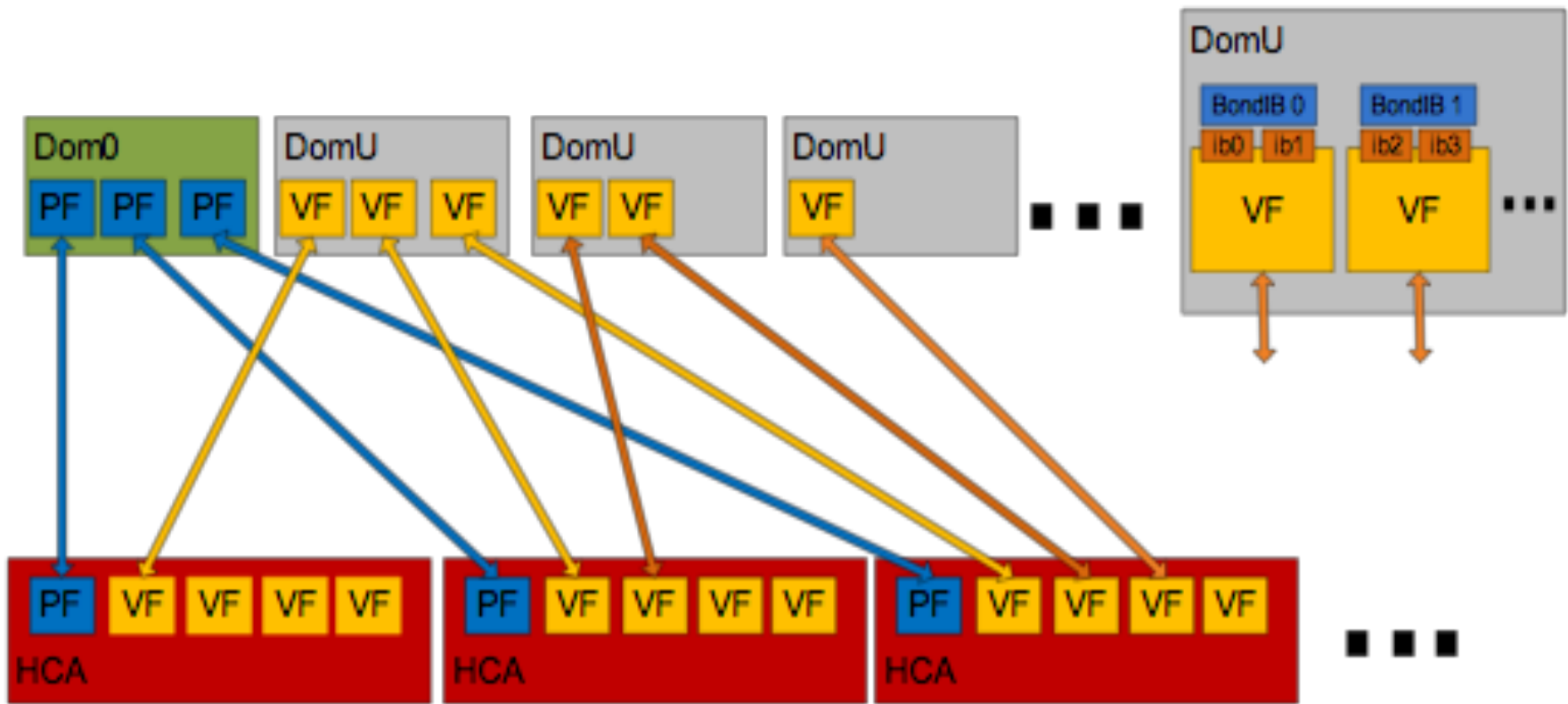


Test Bed / Stack

- Server Hardware Exadata X2-8
- 8 x Ten-Core Intel® Xeon® E7-8870Processors (2.40 GHz)
 - 2 TB Memory
- Chipset support for VT-d and VT-x
- Bios support for SR-IOV, MMIO 64bits
- HCA HW (ConnectX2 MT26428)
 - 8 x InfiniBand QDR (40Gb/s) Ports (4 HCAs)
- Linux distribution in the Guest
 - Oracle Linux 6 (Kernel UEK2 2.6.39)
- OFED (1.5.5-2)
- Oracle VM Server 3.2 (Xen 4.3.1, Linux kernel UEK2 2.6.39)
- Protocol utilized: IPoIB, RDS, SDP

To enable virtualization all pieces of the puzzle need to support virtualization and Infiniband

Server VM IB Setup



- Dom0 manages all PFs
- DomU can have multiple VFs assigned:
 - VFs come from different HCAs
 - VF IB ports are bound together for HA

Network configuration

- IB Bonding support
 - Linux bonding (active-passive) with CX2 HCAs
 - Active Bonding (active-active) with CX3 HCAs
 - Performance benefit from PCIe gen3 systems
 - Enabled via RDS module parameter
 - VF VGUID generated by the PF driver
 - based on the Dom0 PF unique GUID
 - For RDMA with FMR – Dom0 runs both `mlx4_xen_fmr_master` & `slave` drivers while DomUs only run the `slave` driver
- Overall network configuration matches bare metal and follows the same naming convention (`bondib0`, `ib0`, `ib0.pkey`)

PCI Pass-Through configuration

- Determine the BDF (Bus Device Function) of the VF you want to pass through:

```
#lspci
```

```
....
```

```
08:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX VPI PCIe 2.0 5GT/s - IB QDR / 10GigE] (rev b0)
```

```
08:00.1 InfiniBand: Mellanox Technologies MT25400 Family [ConnectX-2 Virtual Function] (rev b0)
```

```
...
```

- Assign the device to pciback instead of its normal driver in Dom0

```
#modprobe xen-pciback
```

```
#echo 0000:08:00.1 > /sys/bus/pci/devices/0000:08:00.1/driver/unbind
```

```
#echo 0000:08:00.1 > /sys/bus/pci/drivers/pciback/new_slot
```

```
#echo 0000:08:00.1 > /sys/bus/pci/drivers/pciback/bind
```

- Verify that the VF is ready to be attached to the VM

```
#xm pci-list-assignable-devices
```

```
....
```

```
0000:08:00.1
```

```
....
```

- Add the the list of VF you want to assigned to the VM in `vm.cfg`

```
pci=['08:00.1']
```



PKeys configuration with SR-IOV

- Standard PKeys configuration remains the same in Dom0 and DomU:

```
#ls /etc/sysconfig/network-scripts/
```

```
ifcfg-ib0
```

```
ifcfg-ib0.c458
```

```
ifcfg-ib0.8002
```

```
...
```

- List available PKeys configured for given PF in Dom0

```
## cat /sys/class/infiniband/mlx4_0/iov/ports/1/pkeys/[0-127]
```

```
0x7fff
```

```
0x8002
```

```
0xc458
```

```
...
```

- Assign PKeys to each VF in Dom0

```
#echo none > /sys/class/infiniband/mlx4_0/iov/0000:08:00.1/ports/1/pkey_idx/[0-127]
```

```
#echo 0 > /sys/class/infiniband/mlx4_0/iov/0000:08:00.1/ports/1/pkey_idx/0
```

```
#echo 2 > /sys/class/infiniband/mlx4_0/iov/0000:08:00.1/ports/1/pkey_idx/2
```



Xen Guest VM configuration

- HVM VM configuration only (no PV VM)
- VM CPU and Memory allocation
 - Memory allocation should be fixed at VM creation time
 - Memory ballooning is disabled because of PCI Pass-through
 - vCPUs pinning to physical CPUs should be done also at creation time
 - Floating vCPU cause performance issues
 - Ensure memory allocation come from local memory domain
- VFs assigned to each VM need to be explicit set in the VM configuration file
 - VF hot-plug is not supported



Pending issues and limitations

- Dynamic memory add/remove not supported
- PCI Pass-Through will prevent Xen operation like save/restore and Live Migration
- Non Transparent Live Migration using ULP that support connection failure/re-establishment like RDS planned in the future with the current HCA generation
- VM needs to be started with pause/un-pause options
- IB management commands restricted in DomU
- Different views of the IB network from Dom0 and DomU
- Stability
 - Fixes required across the stack (FW, BIOS, Kernel, Xen/OVM).
 - More issues found on large systems (e.g. 8 sockets)



Performance consideration

- Overall VF-to-VF bandwidth close to PF-to-PF (5% to 15% impact)
- Less interrupt vectors per VF (3) in each VM in comparison to bare metal (24) – this has an impact as we scale up connections.
- We maintain 2 pools of FMRs (8k and 1M) which are resized and rebalanced dynamically to avoid running periodically out of FMRs
- NUMA impact is important
 - Pin VM vCPUs to physical CPUs
 - NUMA topology not available in VMs
 - Hyper threading visibility not available to VM kernel.
- Scalability
 - Number of VMs capped by the number of VFs
 - Over provisioning causes performance impact
 - Best option is to dedicate vCPUs to each VMs



Conclusion and QA

- IB SR-IOV virtualization support improves performance in comparison to software based approach
- The full stack needs to support SR-IOV
- Xen SR-IOV support present but requires significant planing.
- Validated support for ULPs we need (IPoIB, RDS, SDP), bonding, QoS and PKeys in Guest VM.
- During our evaluation journey many bugs were found across the stack and fixed (~20bugs).
- Future work will focus on performance and scalability