Prototyping Live Migration With SR-IOV Supported InfiniBand HCAs

Vangelis Tasoulas (vangelis@simula.no)

Date: September 12th 2013
Acknowledgements

- Wei Lin Guay (weilin@simula.no)
- Bjørn Dag Johnsen (Oracle), Ola Torudbakken (Oracle), Chien-Hua Yen (Oracle), Sven-Arne Reinemo (Simula) and Richard Frank (Oracle)
Outline

• Introduction
• Problem
• Live migration of VMs with Ethernet SR-IOV VFs
• Live migration of VMs: Ethernet VFs vs IB VFs
• The challenges of live migration with an IB VF
  – Detaching a VF if an active Queue Pair (QP) exists
  – Reallocation of the IB communication resources
  – Handling the outstanding IB operations
  – Reestablishing a remote connection
• Potential approaches for migration of IB SR-IOV VFs
  – The bottom-up approach – IB verbs layer migration
  – The top-down approach – ULP/application aware
• Summary
Introduction – Why VM Migration?

• VM migration is a powerful feature in virtualization
  – Server consolidation and workload distribution
  – Service availability
  – Enable greener data centers with resource overcommitment

• VM migration can be categorized as “cold” or “hot”
  – Cold migration is a traditional way to migrate a VM. The VM is shutdown and then booted at the destination host
  – Hot or Live migration is to migrate a running VM from a host to another with a minimal downtime and transparent to the running applications
Introduction – Why SR-IOV?

- IOV: I/O Virtualization
  - The way hypervisors utilize hardware resources and serve them to virtual machines.
  - Common techniques can be emulation, paravirtualization, passthrough, SR-IOV

- SR-IOV: Single Root I/O Virtualization
  - “I/O Virtualization (IOV) Specifications, in conjunction with system virtualization technologies, allow multiple operating systems running simultaneously within a single computer to natively share PCI Express® devices. The Single Root I/O Virtualization is focused on single root topologies (i.e. a single computer that supports virtualization technology)” - PCI-SIG Consortium
  - Introducing the idea of Physical Functions (PF) and Virtual Functions (VF)
  - Provide improved performance in virtual machines and less CPU overhead to the hypervisor comparing to other types of IOV
Problem

• Unlike emulated or paravirtualized devices, physical devices cannot be paused to save and restore their hardware states so a consistent device state across live migration is impossible using any kind of device passthrough, including SR-IOV
Live migration of VMs with Ethernet SR-IOV VFs

• The Ethernet VF uses the hot plugging mechanism together with the Linux bonding driver to maintain the network connectivity during the migration [1]
  – The fail-over mechanism is provided by the OS at the TCP layer. E.g. TCP timeout is sufficient

• The CompSC [2] as proposed by Zhenhao Pan et al, suggests an extension to the SR-IOV specification where the internal VF state is cloned and migrated as part of the VM migration

[1] Edwin Zhai et al – Live Migration with Pass-through Device for Linux VM - OLS08
Live migration of VMs: Ethernet VFVs vs IB VFVs

• No Linux bonding device available yet for IB native network except bundles with IPoIB (or EoIB)
• Not only need to maintain the hardware state (SR-IOV) but need to keep track of the QP state
• The addressing – LID is assigned by the subnet manager
• With the shared port model*, the QP context cannot be reused after migration. E.g LID, QPN etc.
  – *There are two SR-IOV models for IB HCAs [3]:
    • The shared port model (Mellanox CX2 HCA)
    • The virtual switch model

• Experimental setup
  – Two hosts: Host A and B are connected through IB using one IB switch
  – Each host is an Oracle Sun Fire X4170M2 server
    • Oracle VM server (OVS) 3.0 – Xen based VMM (Hypervisor)
    • The Mellanox ConnectX2 QDR HCA with customized firmware to support SR-IOV
    • The OFED software stack that supports SR-IOV
  – Each VM uses 2 vCPUs, 512MB RAM and one IB VF
    • Running ib_rdma_bw bandwidth test
Live migration with an IB VF

- Migrate VMa from host A to host B
- **Problem 1a**: The VM migration is not allowed if a VF is attached to the VM
  - This is not an IB problem, but a general prerequisite for the PCIe device that needs to be quiesced before hot-swap
- A workaround for **Problem 1a** is to detach the VF from VMa
Live migration with an active IB VF

- The dom0 fails to detach the VF of VMa if an active QP exists (QPa)
- Problem 1b: How to detach a VF in order to migrate a VM if an active QP exists
Detaching a VF with an active QP

- Most applications are calling the IB user verbs
  - The user space application can operate directly on a QP
  - In contrast, the Ethernet operations require to go through the kernel software stack
- All ib_uverbs contexts need to be released before ib_uverbs module can be unloaded
- If a QP is created by an application, detaching a VF returns with an error because the Xen hot-plug script has timed-out before the uverb’s wait_for_complete() completes execution
Detaching a VF with an active QP

- We propose a PID-QPN translation table
- When a QP is created, the PID of the user application is registered in the kernel
- Before ib_uverbs is removed, the kernel signals an event to the user space libmlx4_ib
- The user space libmlx4_ib releases the ib_uverbs contexts
- The kernel uverb’s wait_for_completion() is executed successfully – VF can be detached from VMa
- In order to prevent further operations the communication is halted at the user space libmlx4_ib until a new VF is reattached to VMa
Reallocation of the IB communication resources

- After VMa is migrated from host A to host B and a new VF is reattached to VMa
- The newly attached VF contains a new *vGUID
- The user application continues with an invalid opaque handler that is pointed to Qpa
- In the SR-IOV shared port model, the QP context cannot be reused after migration
- **Problem 2**: How can the user application continues with QPa (QPb) after the migration?

* The vGUID is assigned by the SM to ease the implementation of this prototype. After migration, the associated vGUID to LID mapping may change.
After a new VF with a new vGUID is reattached, the physical resources such as Memory Region (MR), Completion Queue (CQ), QP must be recreated. This includes a user process mapping table to remap the physical resources with QPa’s QPN and to replace the RKEY and LKEY in the posted Work Requests (WR).
User Process Mapping Table

- We propose the translation mechanism as part of the user space libmlx4_ib. Why?
  - The time critical operations do not involve kernel space
  - The same QPN with QPa might exist in Host B. A conflict is avoided if the translation table is only visible per PID.
Handling the outstanding IB operations

• The user application resumes the send operations with QPb on host B

• If there are outstanding operations before migration and there is no recovery mechanism in the ULP, how to retrieve and resume those Send Queue (SQ) operations in host B?

• **Problem 3**: How do we handle the outstanding SQ operations after migration?
Handling the outstanding IB operations

- We propose to use a deterministic state for migration.
- Simulate the SQD-like QP state in software because SQD is not supported by CX2 yet.
  - Work queues are in quiescent state before migration. This is applicable to both sides (QPs) of a connection.
- All the outstanding send operations must be completed (received the CQ) before detaching the VF from host A.
- There are no outstanding operations after the migration.
Reestablishing a remote connection

- Looking from the perspective of the peer QP of the migrated VM
- The original QPa (VMa) is communicating with QPc (VMc)
Reestablishing a remote connection

- Looking from the perspective of the peer QP of the migrated VM
- The original QPa (VMa) is communicating with QPc (VMc)
- With the shared port model,
  - After VMa has migrated to host B, QPb is created to resume the remaining operations.
  - However, QPc is not aware that QPa has been replaced by QPb in host B.
- **Problem 4**: How to maintain the connection with the peer QP after the migration?
Reestablishing a remote connection

- The Communication Manager (CM) must be tolerant to the device removal fault. E.g. do not destroy QPc after the VF is detached from VMa.
- Before VMa releases the uverb, an event is used to notify and suspend QPc. (into the SQD-like state).
  - to avoid QPc transits into the error state by sending to the non-existing QPa.
- Keep track of the Out-Of-Band (OOB) communication (socket address) used by the *CM.
- After a new VF is reattached, QPb is created and a new CM ID is generated to establish a connection with QPc.

* In the current “workaround” in software, we assume all applications are using RDMA_CM to establish the connection and the reconnection mechanism is part of the user space library.
The VMM level migration process

- From the VMM perspective, a three-stage migration process is performed to migrate a VM with an IB VF.
From the VMM perspective, a three-stage migration process is performed to migrate a VM with an IB VF.

- Stage 1: Detach the VF.
The VMM level migration process

- From the VMM perspective, a three-stage migration process is performed to migrate a VM with an IB VF.
  - Stage 1: Detach the VF.
  - Stage 2: Migrate the VM.
The VMM level migration process

- From the VMM perspective, a three-stage migration process is performed to migrate a VM with an IB VF.
  - Stage 1: Detach the VF.
  - Stage 2: Migrate the VM.
  - Stage 3: Attach a new VF.
The VMM level migration process

- From the VMM perspective, a three-stage migration process is performed to migrate a VM with an IB VF.
  - Stage 1: Detach the VF.
  - Stage 2: Migrate the VM.
  - Stage 3: Attach a new VF.

- Without the Linux bonding driver, the three-stage migration process leads to a long service down-time.

- **Problem:** How to reduce the service down-time?
Late-detach Migration

- The Xen migration script is modified to allow migration without detaching the VF during the warm-up stage.
- Dom0 on the migrating source: detach the VF just right before the VM is suspended (stop-and-copy stage).
- Dom0 on the migrating destination: do not initialize the VF during the early restoration, but attach a new VF at the final stage of the restoration.
The bottom-up approach

- The bottom-up approach resolves the above (previous) mentioned challenges.
- The “workarounds” are implemented in both the kernel space and the user space library (libmlx4_ib) to reconfigure the underlying hardware resources during migration.
- The migration model remains the same, it is still based on the three-stage migration process.
The bottom-up approach
The bottom-up approach
The *top-down* approach

- The top-down approach assumes that the complexity of the underlying IB verbs is a black box and the fail-over mechanism is provided by the ULP.
- Reliable Datagram Socket (RDS) as the ULP.
  - RDS is tolerant to fault including the device removal fault.
- The live migration is still based on the three-stage migration process.
  - When a VF is detached, RDS drops the connection.
  - The VM is migrated to the new host.
  - A new VF is reattached and RDS reconnects the connection.
- The top-down approach is not generic because it depends on a dedicated ULP to support live migration.
The top-down approach

Parameters of the rds-stress test: -q 4K -a 4K -t 7
Summary

- The bottom-up approach provides a generic solution.
  - The current prototype has a service downtime of 2.7s.
- How to further improve the service downtime and reconfiguration?
  - The *vSwitch* model is a better architecture.
    - The QP namespace is isolated perVF.
    - A better model from networking and routing perspective.
    - How to resolve the scalability issue with the *vSwitch* model?
      - Bloats LID space which is only 16 bits
    - A new state to define suspend?
Questions?