CompSC: Live Migration with Pass-through Devices

ZHENHAO PAN &, YAOZU DONG *, YU CHEN &, LEI ZHANG &, ZHIJIAO ZHANG &,
&Tsinghua University, *Intel Asia-Pacific Research and Development Ltd.

VEE 2012
Outline

- Introduction
- CompSC solution
- Experiments
- Conclusion
Introduction

• Background
  o Live migration
  o Pass-through device
  o SR-IOV spec

• Experimental result
  o Liv migration with SR-IOV NIC
    o 282.66% more throughput
    o 42.9% less downtime
Introduction

- **SR-IOV Specification**

- **Start with a single function device**
  - HW under the control of privileged SW
  - Includes an SR-IOV Extended Capability
  - Physical Function (PF)

- **Replicate the resources needed by a VM**
  - MMIO for direct communication
  - RID to tag DMA traffic
  - Minimal configuration space
  - Virtual Function (VF)

- **Introduces PCI Manager (PCIM)**
  - Conceptual SW entity
  - Completes the configuration model
  - Translates VF into a full function
  - Configures SR-IOV resources
Related work

- **Bonding driver** [Linux Ethernet Bonding Driver HOWTO]
- Failover/Load balance
- **NPIA (Network Plug-in Architecture)**
Related work

- **VMDq (Virtual Machine Device Queue)**
  - Multiple queue pairs for partitioning

![Diagram showing VMDq and related components]

- Why not store/restore device states directly?
Outline

• Introduction
• **CompSC solution**
• Experiments
• Conclusion
CompSC Approaches

• Requirement challenges
  – The state (such as registers) of the device needs to be efficiently read and written to support device state replication;
  – The dirty memory written by the device Direct Memory Access (DMA) needs to be efficient and tracked for lazy memory state transmission.
CompSC Approaches

- interface of Hardware – OS
  - Registers
  - DMA
  - Interrupts

![Diagram showing CPU, Memory, IO W/R, DMA, Device Controller, Bus Interface, Hardware Controller, Addressable Memory and/or Queues, Registers, Memory Mapped Region, and Interrupt Request connections.]
Compsc Approaches

- Requirement of I/O Register migration:
  - Most parts: Read/write, No side effect
  - Some special: RO/WO, RC/WC, etc., with side effect

<table>
<thead>
<tr>
<th>Register type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read-write</td>
<td>If written since reset, the value read reflects the value written.</td>
</tr>
<tr>
<td>read-only</td>
<td>Writes to this reg have no effect.</td>
</tr>
<tr>
<td>write-only</td>
<td>Reading this reg returns no meaningful value.</td>
</tr>
<tr>
<td>read-write-clear</td>
<td>A register can be read and written. However, a write of a 1b clears the corresponding bit.</td>
</tr>
<tr>
<td>write-clear</td>
<td>Writing 1b to register clears an event possibly reported in another register.</td>
</tr>
<tr>
<td>read-clear</td>
<td>A register bit with this attribute is cleared after read. Writes have no effect on the bit value.</td>
</tr>
<tr>
<td>read-write-set</td>
<td>Register that is set to 1b by software, and cleared to 0b by hardware.</td>
</tr>
<tr>
<td>reserved</td>
<td>Reserved field can return any value on read access and must be set to its initial value on write access.</td>
</tr>
</tbody>
</table>
CompSC Approaches

State replay for side effect

• Method
  o Record every hardware access (Recording stage)
  o Replay them on the target device (Replaying stage)

• Optimization 1
  o Record last reg writing when this writing brings no side effect

• Optimization 2
  o Define operation sets (op set), the op sets is Critical Section
CompSC Approaches

State replay – with op set

Op sets in Intel 82576/82599 NIC
- All *initializing* operations
- All *Sending* operations
- All *Receiving* operations
- other remaining op states include only \{uninitialized, up, down\}

In this kind of set up, only the latest operations on each setting register and whether or not the interface is up need to be tracked.
CompSC Approaches

Self-emulation for Read-only, etc. Registers

- Design for statistic registers (read-only/read-clear)
- Require mathematical attributes (monotonicity)
- Example: dropped packets counter
  - \( n \) before migration
  - initialized to 0 when migration
  - \( m \) now (after migration)
  - correct value = \( n + m \)
CompSC Approaches

• Dummy writing for DMA dirty page
  – *DMA dirty page tracking*. To replicate the I/O state, memory pages modified by the device DMA operations must be efficiently tracked for efficient live migration. Unfortunately, DMA dirty page tracking is not supported in the existing I/O MMU.
  – Dummy write the DMAed page after DMA process finished.
CompSC Architecture

Design & Implementation

live migration
1. Pre-Migration stage
2. Reservation stage
3. Iterative Pre-copy stage
4. Stop-and-copy stage
5. Commitment stage
6. Activation stage
## CompSC Implementation

### Xen and SR-IOV NIC drivers

**Implementation complexity** ~2000 LoC

<table>
<thead>
<tr>
<th></th>
<th>Line of code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xen hypervisor</td>
<td>362</td>
</tr>
<tr>
<td>Xen tools</td>
<td>446</td>
</tr>
<tr>
<td>VF driver (common)</td>
<td>153</td>
</tr>
<tr>
<td>IGBVF driver</td>
<td>344</td>
</tr>
<tr>
<td>IGB driver</td>
<td>215</td>
</tr>
<tr>
<td>IXGBE VF driver</td>
<td>303</td>
</tr>
<tr>
<td>IXGBE driver</td>
<td>233</td>
</tr>
</tbody>
</table>
CompSC Implementation

Xen and SR-IOV NIC drivers

• Intel 82576 Gbps NIC & 82599 10Gbps NIC
• PF/VF drivers

• Driver changes on IGBVF/IXGBEVF
  o Rlock every hardware operation
  o Pack igbvf_up/igbvf_down and ixgbevf_up/ixgbevf_down into operation sets
  o Restoration after migration
CompSC Implementation

Xen and SR-IOV NIC drivers
• Shared memory for sync
  o rw-lock and version counter
  o List of registers for I/O register migration
  o List of registers for self-emulation
• Synchronization for Live Migration
  o Acquire w-lock before suspending
  o Increase version counter
  o Release w-lock after migration
  o Invoke driver restoration at first r-lock
CompSC Implementation

Xen and SR-IOV NIC drivers

• Pages dirtied by DMA
  o In x86/x64, memory access by DMA cannot be tracked on page tables by MMU, IOMMU
  o In CompSC, driver performs dummy writes to descriptor/buffer when receive an interrupt
  o May cause packet miss/packet duplication during migration

<table>
<thead>
<tr>
<th></th>
<th>Dup</th>
<th>Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>No workload</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>scp</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SPECweb</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
CompSC Implementation

Xen and SR-IOV NIC drivers

• Descriptor ring
  o Descriptor ring head index is in read-only register
  o Altering head index is hard (hard for state replay)
  o CompSC introduces an offset between the ring in hardware's view and software's view
  o During migration, increase the offset to make sure ring head index on target hardware is 0

![Diagram of ring structure]

Head

Tail

Head

Tail
Outline

- Introduction
- CompSC solution
- Experiments
- Conclusion
Experiments

• Physical Environment
  o Intel Core i5 670 (with VT-x, VT-d, VT-c features)
  o 4GB memory, 1TB hard disk
  o Intel 82576 & Intel 82599 NICs

• Virtual Environment
  o 4 vCPU
  o 3GB memory
  o PF/VF of Intel 82576 or Intel 82599 NIC
Experiments

Evaluation - Throughput

Intel 82599

9.4 Gbps

int: sel f-emulation
Experiments
Evaluation - SPECweb2009

Intel 82599
Experiments

Evaluation - Live migration (Bonding v.s. CompSC), 82576 NIC

Netperf
Experiments

Evaluation - Live migration (PV v.s. CompSC), 82599 NIC

PV

CompSC

Dom0 and the guest were sharing the physical CPU

Migration start

Service up

Service down
Consolution

• Proposed a directly solutions for live migration of pass-through device : CompSC
  – Support Live Migration with SR-IOV NIC

• Future
  – Evaluate NPIA method
  – Support Checkpoint (such as Remus in XEN)
  – Other SR-IOV devices
  – ...
Thank you for your attention!

Questions?