Software and hardware support for Network Virtualization

Knut Omang
Ifi/Oracle
19 Oct, 2015

Motivation

- Goal: Introduction to challenges in providing fast networking to virtual machines

Prerequisites:

- What is virtualization?
- Understand interplay between
  - software ideas/application abstractions
  - hardware evolution
  - compatibility requirements!
- Understand some of the recent hardware additions to CPUs and chipsets
- Understanding some of the underlying APIs we virtualize upon!

Overview

- Introduction to virtualization (Virtual machines)
- Aspects of network virtualization:
  - Virtual network infrastructure, interfaces, adapters
- Network interface attach points (PCI, PCIe)
- Software emulation of a network interface
- Paravirtualized network interfaces
- Hardware support for sharing a network adapter (SR/IOV)
- Use cases, challenges, risks and tradeoffs

Virtualization

- Present an abstraction to the application (guest OS, user program..)
- About resource sharing, resource utilization
  - Not new: ex. process, virtual memory - just taking it further..
  - Virtual memory, virtual disk head, virtual CPU, virtual computer..
- As in virtual machines:
  - Host operating system (often called hypervisor) sees whole computer
  - Guest operating system only sees a partition of the real computer
  - Protection and transparency
  - Flexible use of machine resources

Virtualization of resources

- Motivated from the programming side (software)
- Implementation in software faces problems:
  - performance
  - security
- Hardware: How can we support it better?
- Think about basic OS abstractions..
- Ongoing driver for hardware development
- Applies to network side as well

Virtualization →isolation

Popek and Goldberg,1974:

- *Sensitive instructions*: Instructions that for protection reasons must be executed in kernel mode
- *Privileged instructions*: Instructions that causes a trap

A machine is *virtualizable* iff the set of sensitive instructions is a subset of the set of privileged instructions.

Virtualization before ca.1995

IBM CP/CMS -> VM/370, 1979

- Hardware support: Traps sensitive instructions
- Architecture still in use for IBM “mainframes”

- Largely ignored by others
  - Taken up by Sun and HP in 1990's
  - x86-world? Difficult because:
    - Some sensitive instructions ignored in user mode!
    - Some sensitive instructions allowed from user mode!

Virtualization in the (limited) x86

Problems:

- Performance:
  - I/O
  - Page faults
  - Interrupts (when?)
  - Host resource usage
- Avoidig 'leaking' instructions
  - Pentium allows instruction that makes it possible to determine if it is executed in kernel mode
  - Might confuse OS..

Virtualization in the (limited) x86

Solutions:

- Interpretation (emulating the instruction set)
  - Performance penalty of factor 5-10
  - Benefit: May emulate any type of CPU
- “Full” virtualization
  - Privileged instructions in guest OS'es rewritten by virtualization software (binary translation)
  - Stanford DISCO --> VmWare workstation
    - Did not require source code of OS!
- Paravirtualization
  - Replacing parts of the guest operating system with custom code for virtualization

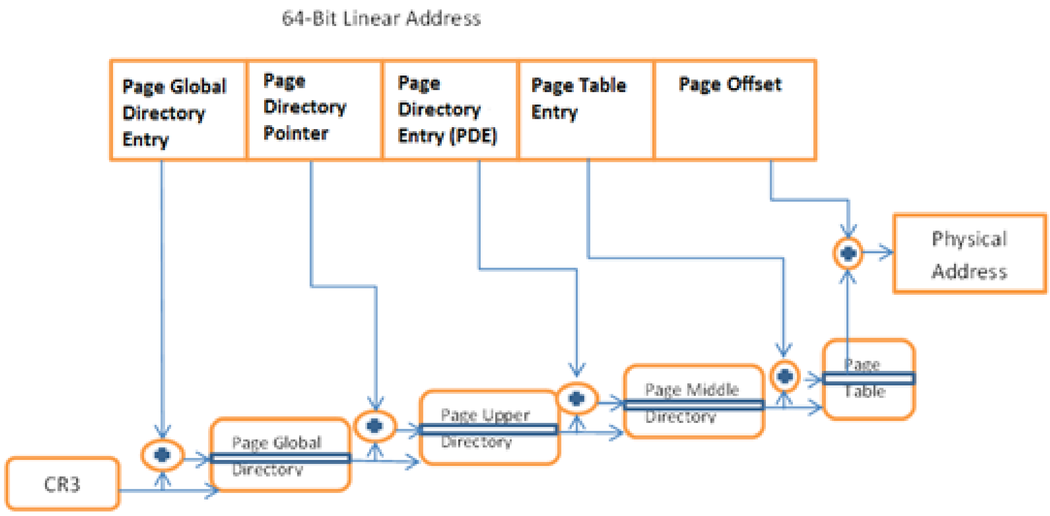
Xen PV (Xen Paravirtualization)

- Uses x86 privilege levels differently:
  - Rings: 0, 1, 2, 3 (highest to lowest privilege)
  - Normally OS executes in ring 0 and applications execute in ring 3
  - With Xen
    - 0 – Hypervisor
    - 1 – Guest OS
    - 2 – unused
    - 3 – Applications
  - Guest OS modified for privileged instructions
- Still used for dom0 (privileged guest mode) in Xen
- VMWare ESX: similar approach

Initial hw support for virtualization on x86_64

- VT-x(Intel) and SVM(AMD):
  - Inspired by VM/370
  - Set of operations that trap
    - controlled by bitmap managed by host OS/hypervisor
- Present in most (all?) newer 64 bit versions of AMD/Intel processors
  - must sometimes be enabled in BIOS
- Delivers isolation according to Popek & Goldberg
- Effectively privileged mode, guest privileged mode and user mode..
- On linux: `cat /proc/cpuinfo | egrep 'svm|vmx'`

Intel x86_64 page tables

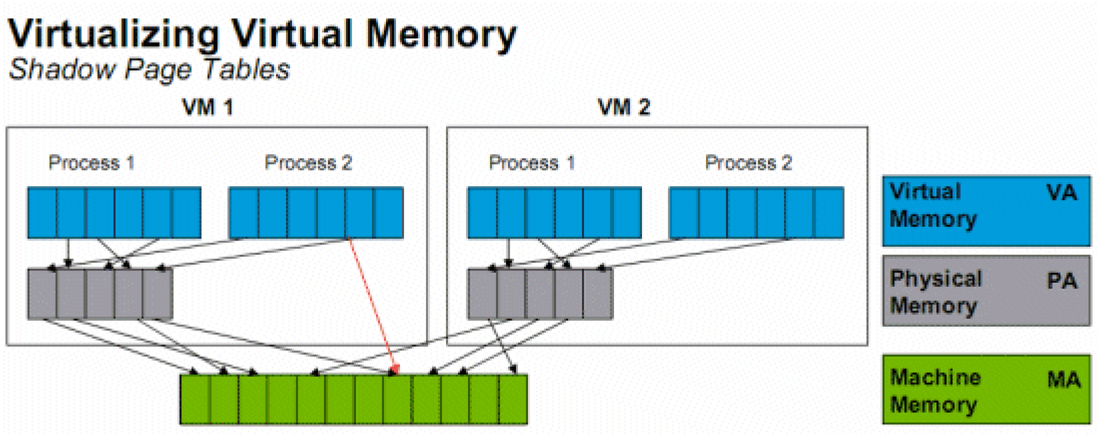


Shadow page tables

- Guest page table is GVA ->GPA
- Hypervisor maintains shadow page tables
  - Trap all changes to guest page tables
  - Sync shadow page table: GVA → HPA
- Very expensive to keep these tables in sync:
  - lots of traps!
  - memory overhead of extra page tables!

ORACLE

Shadow page tables



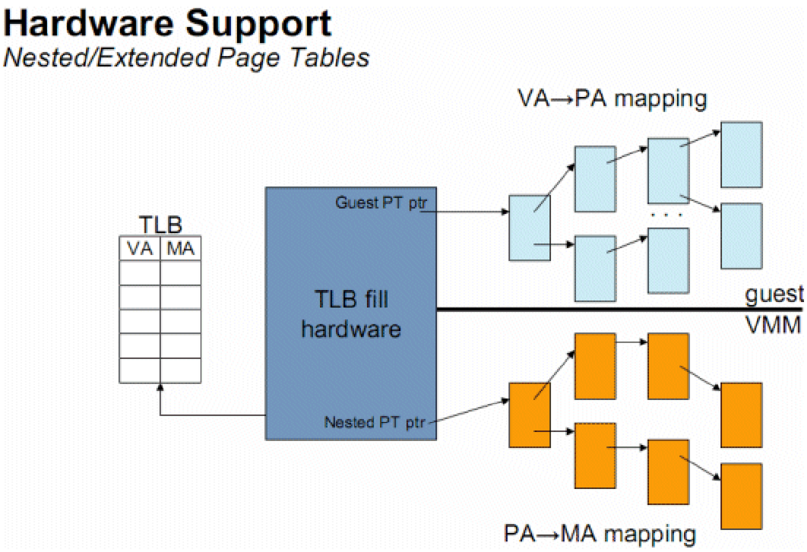
http://www.anandtech.com/show/2480/10

ORACLE

Extended/nested page tables

- Problem: Very expensive virtual machine context switches!
- OSes expect to "own" address space
  - Need extra level of page tables
- Two virtual machines could have the same guest physical addresses
  - but these "physical pages" were pointing to different host physical pages
  - All state about pages must be flushed upon machine switch (VM exit)
- Hardware solution:
  - Intel: Extended page tables (ept + vpid)
  - AMD: Nested page tables (npt)
  - Introduces hardware support to distinguish guest physical addresses from different machines
  - Extra page table: GPA → HPA

Extended page tables



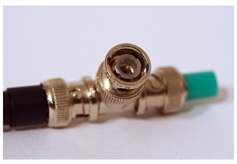
Network virtualization: 1) Virtual infrastructure

Ethernet as example:

- Hardware: broadcast → point to point
- HUBs to switches
- VLAN: Sharing physical links
- Wireless/mobile..
- Speed ↔ technology
- Pure software: virtual switches and links

Ethernet…








Ethernet

- CSMA/CD (Carrier Sense Multiple Access w/Collision Detection)
- Half-duplex, serial, single wire pair
- Designed for optimistic, unreliable broadcast w/repeaters
- Today:
  - somewhat related usage on Wifi
  - But: Mostly reliable point-to-point w/switches (parallel) full-duplex
- Speed: 10,100,1000Mb/s, 10,40,100Gb/s,...?
- Software aspects of Ethernet as success factor:
  - Extensible, flexible protocol
  - high penetration...

Virtual Ethernet?

VLAN (IEEE 802.1Q):

- Extra VLAN ID field in Ethernet packet
- Allows several logical networks to use same medium
- "Smart" switches and routers define who will see which logical streams of packets
- Benefits: Flexibility, saves expensive wire resources, network capacity
- Drawbacks? Complexity, security issues, requires switch support

Ethernet and machine virtualization?

- Virtual ethernet switches
  - to be able to route packets to a virtual machine
- Virtual packet forwarding
  - logical transport links between VMs within a host
- Host virtual ethernet
  - tap, bonding devices
- Virtual ethernet devices in virtual machines
  - emulated (OS sees a “real” device)
  - paravirtualized (custom interface with hypervisor)
- In common: All operates on Ethernet packets
  - benefits? drawbacks?

Network interface attach points (Linux)

Ethernet not the only network interface, abstraction layer:

- serial ports
- parallel ports
- tun (IP packets)
- IPC sockets
- USB
- Firewire
- Bluetooth
- RDMA devices (later)

Virtual ethernet support within a Linux host

- Virtual switches (bridges)
  - brctl
- Virtual ethernet interfaces (tap devices)
  - tunctl
- Packet filtering
  - ebtables
- Tunneling

Virtual IP network support on Linux

- Tun devices
  - tunctl tunN | -n dev
- Packet filtering
  - iptables, ip route + NAT
- Queuing and manipulation of packet queues
  - tc (traffic control)

The system I/O "bus"

- Before the PC: Proprietary, incompatible → expensive!
- IBM PC: AT bus - tried MicroChannel Architecture
- ISA (Industry Standard Architecture) bus!
  - "clone" manufacturer effort
  - parallel broadcast medium, 8 or 16 bit at a time
  - Hardware design: slot design, pinout standardized w/extensions
- 486: ISA bus too slow for video req → VESA local bus: 32 bit isa
- Pentium: PCI + ISA for bw comp

PCI (Peripheral Component Interconnect)

- DMA (Direct Memory Access) support for devices
- New, more compact physical design
- Standardized, extensible software interface!
- 3 Address space types:
  - Config space
  - I/O ports (ISA compat++)
  - Memory mapped I/O (MMIO)
- Config space has standardized layout, standardized semantics

ISA vs PCI

Why do we care about details of an obsolete I/O bus?

- Common software implementations of virtualization emulates a PCI based system architecture
- Most OSes automatically recognize and are able to tell something about PCI devices
- Some OSes would probably not even boot if no PCI host bridge was detected!
- It is basically a good API to base a virtual device on!
- PCI Express is an extension of PCI
  - from a software perspective! (remember old and new Ethernet? ;-))

PCI config space

31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Lat. Timer	Cache Line S.	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Cap. Pointer	34h
Reserved				38h
Max Lat.	Min Gnt.	Interrupt Pin	Interrupt Line	3Ch

```
[root@o4kvm205 ~]# lspci -xxx -s 00:03.0
00:03.0 Ethernet controller: Intel Corporation 82540E
00: 86 80 0e 10 07 01 00 00 03 00 00 02 00 00 00 00
10: 00 00 bc fe 01 c0 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 f4 1a 00 11
30: 00 00 b8 fe 00 00 00 00 00 00 00 00 0b 01 00 00
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

[root@o4kvm205 ~]# 
```

PCI config space, Pentium emulation

```
[root@o4kvm205 ~]# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Device 1234:1111 (rev 02)
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)
00:04.0 Unclassified device [0002]: Red Hat, Inc Virtio filesystem
00:05.0 SCSI storage controller: Red Hat, Inc Virtio block device
[root@o4kvm205 ~]# lspci -vvv -s 00:03.0
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)
        Subsystem: Red Hat, Inc QEMU Virtual Machine
        Physical Slot: 3
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR+ FastB2B- DisINTx-
        Status: Cap- 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin A routed to IRQ 11
        Region 0: Memory at febc0000 (32-bit, non-prefetchable) [size=128K]
        Region 1: I/O ports at c000 [size=64]
        Expansion ROM at feb80000 [disabled] [size=256K]
        Kernel driver in use: e1000
        Kernel modules: e1000

[root@o4kvm205 ~]# 
```

PCI BAR (Base Address Register) spaces

- OS writes 0xffffffff to determine size/validity - size is 2^n
- Only the bits needed to set an aligned address are writable
- OS can deduce size - writes back desired PCI address.
- Device listens to PCI requests to range from BAR address + BAR size.
- Up to 6 32 bits regions
- Up to 3 64 bits regions
  - Registers used in pair
- BAR type information in read only lower bit