Paper Reading:
Virtualizing I/O devices on VMWare Workstation's hosted virtual machine monitor

J. Sugerman, G. Venkitachalam, B.-H. Lim
presented by Bjoern Doebel

2007-04-04
Overview

- VMWare workstation overview
- Virtualizing I/O
- I/O performance
- Optimizations
- Further Ideas
VMWare overview

Host world
- Host OS
  - VMDriver
- Applications

VM world
- VMM
  - Virtual Machine
- Guest OS
  - Applications

Physical Machine
I/O: Network interfaces

- Virtual Machine
  - Virtual NIC
  - Virtual Network Hub (Host-Only)
    - Virtual NIC
    - VMNet Driver
  - VMM

- Virtual Machine
  - Virtual NIC
  - Virtual Network Hub (Bridged)
    - Virtual Bridge
    - VMNet Driver
  - VMM

- Physical Ethernet
  - Physical NIC

- Host OS, VMDriver & VMAppl
I/O: Network send

Applications -> VMApp

VMNet -> VMDriver

Host OS

Guest OS

Virtual Machine

VMM

Applications

Ethernet line

HW send

System call to VMNet

Send via host driver

OUT to I/O port

World switch

Return to VMApp
• Receiving packet even more complicated
• Overhead occurs for every packet!
• Packet send involves 11 (!) separate IN/OUT instructions that are propagated to the real NIC
  – 26.8 % of all overhead
• One interrupt on every send or receive
  – VMM not able to handle IRQs itself
  – IRQ handlers typically execute further IN/OUT instructions
• VMApp uses `select()` to wait for events on network devices (incoming / outgoing)
Performance optimizations

- Handle I/O port accesses in VMM
  - only 1/3 needs to be propagated to host
  - most effective optimization, because many world switches are avoided

- Send combining
  - queue packets in VMM, send several at once to save some more world switches

- Use shared mem between VMApp and VMNet driver to save expensive select() calls
Performance optimizations (2)

Transmit Bandwidth

Data Size

MBits/second

PC-733, PC-350
Optimized VM/PC-733
Version 2.0 VM/PC-733
Optimized VM/PC-350
Version 2.0 VM/PC-350
Further optimizations

- Reduce CPU virtualization overhead
  - modify virtualized PIC for better performance
- Modify guest OS
  - avoid PT switches from/to idle task
  - removes some of the 8.5% overhead spent for PT virtualization
- Optimized guest device driver
  - idealized interface with fewer IN/OUT instructions per send/receive
  - requires specialized driver for each guest (however many guests support loadable modules...
Further optimizations (2)

- Modify Host OS
  - adapt Linux' sk_buff handling
- Bypass Host OS
  - only host handles IRQs
  - leads to lots of world switches
  - can we drive the device directly from the VMM?
    - need additional management to multiplex
device between VMMSs and host
    - need own drivers for VMM
  - called “hypervisor direct I/O” in VMWare ESX Server
Discussion

• “Further optimizations” are basically **paravirtualization**. Will future virtualization techniques benefit from best of both worlds?

• Can we encourage OS developers to make their OSes more virtualization-friendly? Tradeoff between increased complexity and increased performance?

• Is improved hardware the only real solution? (TCP Offload Engines, Remote DMA, Intel I/OAT, **Passthrough I/O** using I/O MMUs and Partitionable I/O devices)