Virtualizing I/O

Yiying Zhang
Quiz next Mon: cover everything until (including) today’s lecture

- Open everything (you can also use google), but no communication to anyone else

- Quiz will be on Canvas, and you’ll need to join Zoom with camera turned on during the quiz (so that we can proctor)

- Quiz will be 15 minutes (only few small questions, e.g., T/F, MC, short answers)

- Project proposal due on 1/24

- Finalize project topic choice (and update Google form if needed)

- Breakout room for project discussion at the end of the lecture
Leftover: Memory Sharing

- Motivation
  - Multiple VMs running same OS, apps
  - Collapse redundant copies of code, data, zeros

- Transparent page sharing
  - Map multiple PPNs to single MPN (copy-on-write)
  - Pioneered by Disco, but required guest OS hooks

- New twist: content-based sharing
  - General-purpose, no guest OS changes
  - Background activity saves memory over time
Leftover: Page Sharing: Scan Candidate PPN
Leftover: Page Sharing: Successful Match

VM 1

VM 2

VM 3

Machine Memory

shared frame

Hash: ...06af
Refs: 2
MPN: 123b

hash table
Outline

• Device virtualization techniques
• Storage virtualization
• Network virtualization

Acknowledgment: some slides from Scott Devine’s lectures at Columbia, Teemu Koponen’s NSDI’14 presentation, and Ben Pfaff’s Network Virtualization lecture
I/O Virtualization

- Goal
  - Multiplexing device across guest VMs
- Challenges
  - Each guest OS has its own device driver
  - How can one device be controlled by multiple drivers?
  - What if one guest OS tries to format its disk?
Possible Solutions of I/O Virtualization

- Direct access: VM exclusively owns a device
- Device emulation: VMM emulates device in software
- Para-virtualization: split driver into guest part and host part
- Hardware assisted: hardware devices offer isolated “virtual interfaces”
Sol-1: Direct Access Device Virtualization
Sol-1: Direct Access Device Virtualization
Sol-1: Direct Access Device Virtualization
Direct Access Device Virtualization

- Positives
  - Fast, since the VM uses device just as native machine
  - Simplify monitor: limited device drivers needed

- Negatives
  - Hardware interface visible to guest (bad for migration)
  - Interposition is hard by definition (no way to trap & emulate)
  - Now you need much more devices! (imagine 100 VMs)
Sol-2: Emulating Devices

- Emulate a device
  - Implement device logic in pure software
  - Can even emulate non-existing devices
Example: Emulating Serial Port
Emulated Devices

• Positives
  • Platform stability (good for migration)
  • Allows interposition
  • No special hardware support is needed

• Negatives
  • Can be slow (it’s software emulated)
Sol-3: Para-Virtualized Devices

- VMM offers new types of device
- The guest OS runs a new driver (*front-end driver*)
- VMM runs a *back-end* driver for each front-end
- VMM finally runs the real device driver to drive the device
virtio: Linux’s paravirtualized I/O solution

- **Front-end Driver**
  - A kernel module in the guest OS
  - Accepts I/O requests from the user process
  - Transfer I/O requests to back-end driver

- **Back-end Driver**
  - Accepts I/O requests from front-end driver
  - Perform I/O operation via physical device

- **Virtqueue**
  - A memory region accessible from both guest and host OS
  - An interface implemented as vring
Sol-4: Hardware Support for I/O Virtualization

- A virtualization-enabled device can be configured to appear in the PCI configuration space as multiple virtual functions (VFs)

- The VMM assigns one or more VFs to a VM by mapping the actual configuration space of the VFs to the configuration space presented to the VM by the VMM
SR-IOV (Single Root I/O Virtualization)

- Hardware support for guest to access device without going through VMM
- Physical Function (PF)
  - A standard PCIe function
  - Can be associated with multiple VFs
- Virtual Function (VF)
  - A lightweight PCIe function (a unique PCIe xact source)
  - Each VF is isolated from other VFs
  - Has dedicated access to certain hardware resources
  - Share some other resources
- IOVM and PF driver: set up VFs and provide full features to each VF
## Virtualization Technologies Summary

<table>
<thead>
<tr>
<th>Virtualization</th>
<th>Software Solution</th>
<th>Hardware Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>• Trap &amp; Emulate</td>
<td>• VT-x</td>
</tr>
<tr>
<td></td>
<td>• Instruction interpretation</td>
<td>• Root / non-root mode</td>
</tr>
<tr>
<td></td>
<td>• Binary translation</td>
<td>• VMCS</td>
</tr>
<tr>
<td></td>
<td>• Para-virtualization</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>• Shadow page table</td>
<td>• EPT</td>
</tr>
<tr>
<td></td>
<td>• Para-virtualization</td>
<td></td>
</tr>
<tr>
<td>Device</td>
<td>• Direct I/O</td>
<td>• SR-IOV</td>
</tr>
<tr>
<td></td>
<td>• Device emulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Para-virtualization</td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Device virtualization techniques
• Storage virtualization
• Network virtualization
Virtual Disk

- A virtual disk is just a file in host file system
- Hypervisor maps disk blocks to file offsets
  - Flat file (fix sized virtual disk)
  - Indexed file (virtual disk can grow on demand)
Outline

• Device virtualization techniques

• Storage virtualization

• Network virtualization
VMware Workstation’s Network Subsystem

Figure 3: VMware’s network subsystem provides virtual Ethernet adapters, hubs and bridges. A hub can be either be bridged to a physical Ethernet adapter, or connected to a virtual network interface in the host OS. The virtual bridge and hub are implemented via a VMNet driver that is loaded into the host OS.
Data Center Network Design with VMs

- Core Switch
  - Aggregation Switch
    - "Top of Rack" Switch
      - virtual switch (= vswitch)
      - Machine 1
        - VM VM VM
      - Machine 2
        - VM VM VM
      - ... (other ToRs)
    - Machine 40
      - VM VM VM

One rack of machines

up to 128 VMs each
Virtualizing Network Functionalities

<table>
<thead>
<tr>
<th></th>
<th>VLAN</th>
<th>NAT</th>
<th>MPLS</th>
<th>VRF</th>
<th>Network Elements as VMs</th>
<th>FlowVisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet</td>
<td></td>
<td>IP address space</td>
<td>Path</td>
<td>L3 FIB</td>
<td>Elements</td>
<td>ASIC</td>
</tr>
</tbody>
</table>

Plenty of primitives but **no** network virtualization per se.
Problems?

• A physical topology is hard to support different virtual topologies

• Virtualized workloads stay in the physical network address space (L2)
  
  • Slow provisioning

  • Limited mobility

  • Limited VM placement

  • Hardware dependent

  • Operationally intensive

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
Goals (Paraphrase with Machine Virtualization)
Network Virtualization

Packet Abstraction

Mgmt

Mgmt

Mgmt

Logical Network
Logical Network
Logical Network

Network Hypervisor

Control Abstraction

Packet Abstraction + Control Abstraction = Network Hypervisor
Management, Control, and Data Planes
Virtual Network Encapsulation (Data Plane Solution)
Distributed Network Services
Open vSwitch (OVS)

- Software-based virtual multi-layer switch, open source
- ovsdb for storing configurations
- Uses a flow cache in kernel for fast processing
- First packet goes to the user-space ovs-vswitchd
Open vSwitch: Design Details

Hypervisor physical machine

Virtual machines

Host operating system

Hypervisor

Cache hierarchy:
<1us: Kernel module
<1ms: ovs-vswitchd
<10ms: controller

Controller