Running KVM for Dynamic Infrastructure Creation

Jay Brenneman – rjbrenn@us.ibm.com
August 12, 2015

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

AIX*	FlashSystem	Storwize*	Tivoli*
DB2*	IBM*	Spectrum Scale*	WebSphere*
DS8000*	IBM (logo)*	System p*	XIV*
ECKD	MQSeries*	System x*	z/VM*
		System z*	Z Systems*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.
TEALEAF is a registered trademark of Tealeaf, an IBM Company.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
Worklight is a trademark or registered trademark of Worklight, an IBM Company.
UNIX is a registered trademark of The Open Group in the United States and other countries.

* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html ("AUT"). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Agenda

KVM Review

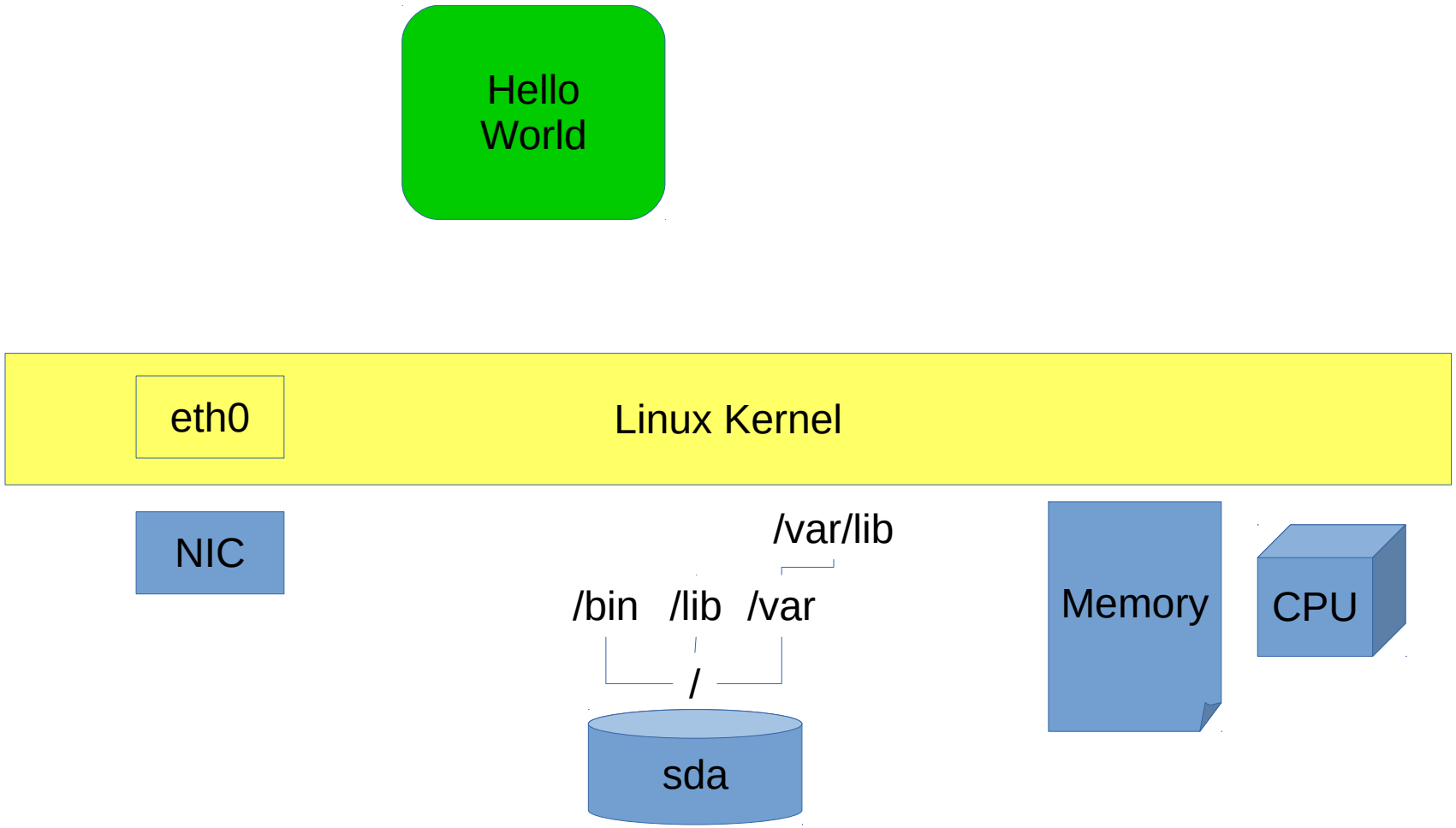
Virtualizing Networks with KVM

Virtualizing Disk with KVM

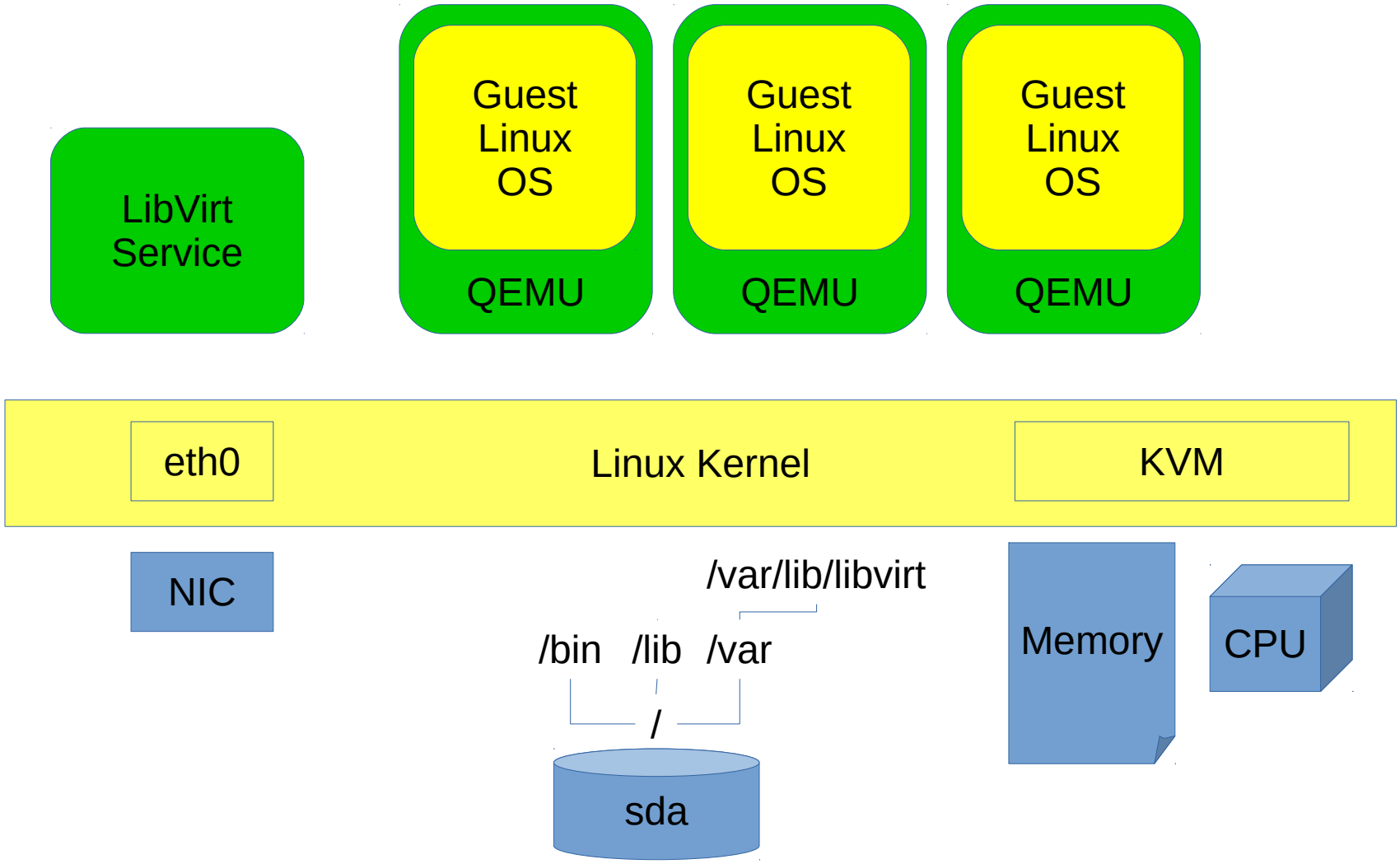
KVM Review

- Hopefully you attended Mark Post's 'KVM for z Systems' session ?
- QEMU Virtualizes the Processor and Memory resources of a Server using the KVM resources provided by a Linux Kernel assisted by whatever hardware instructions the processor supports
- QEMU Virtualizes the IO of a server using a mix of paravirtualization and good ole trap and translate.
- From an administrators point of view – a virtual machine is contained within the qemu process running on the KVM host

KVM Review


Hello
World
eth0
Linux Kernel
NIC
/var/lib
/bin
/lib
/var
/
sda
Memory
CPU

KVM Review


LibVirt Service
Guest Linux OS
QEMU
Guest Linux OS
QEMU
Guest Linux OS
QEMU
eth0
Linux Kernel
KVM
NIC
/var/lib/libvirt
/bin
/lib
/var
/
sda
Memory
CPU

Agenda

KVM Review

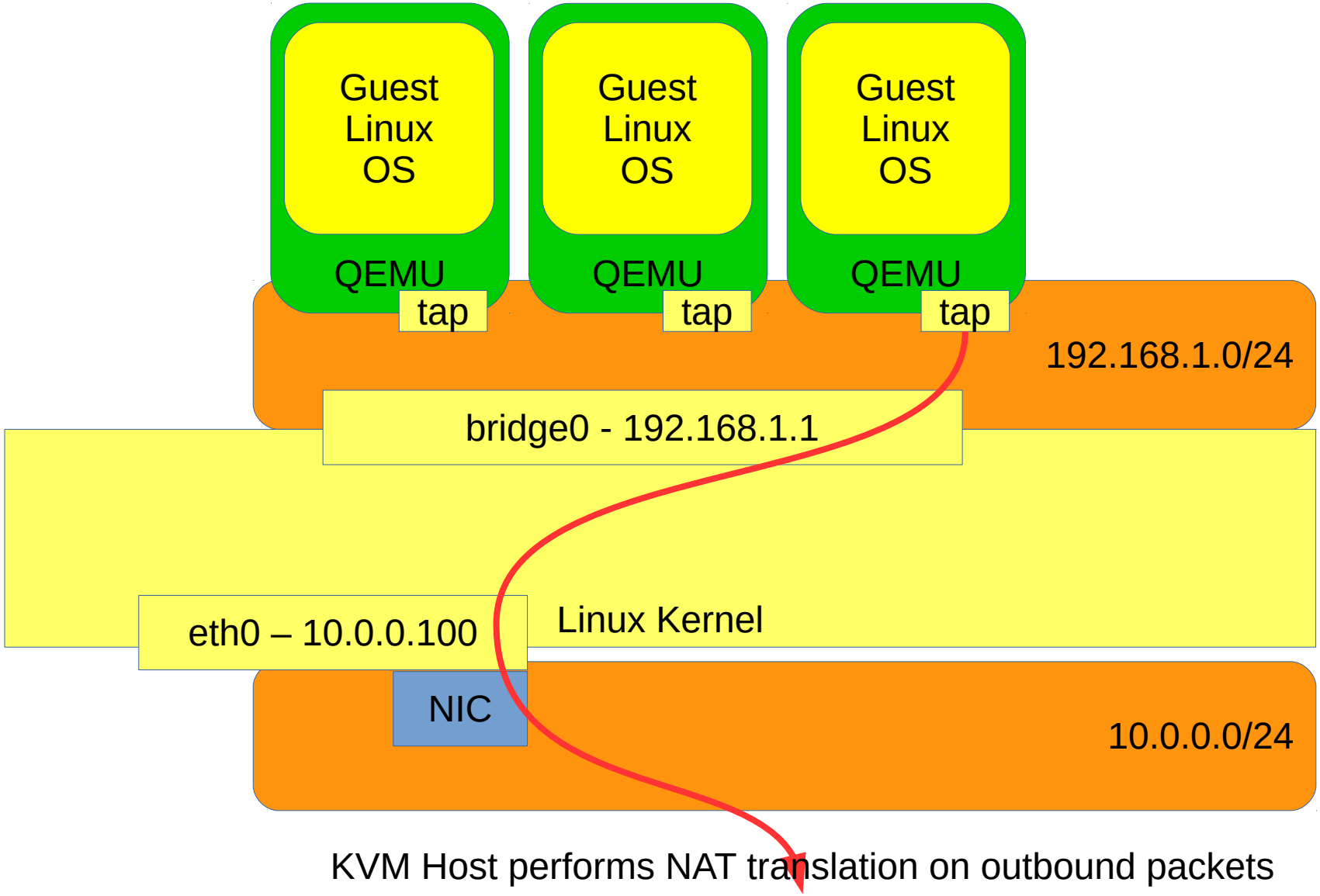
Virtualizing Networks with KVM

Virtualizing Disk with KVM

Virtualizing Networks with KVM

- KVM offers several methods to virtualize the host's network connectivity, each with their own plusses and minuses
  - Ethernet Routing with NAT
  - Ethernet Routing without NAT
  - Ethernet Bridging
  - MacVTap
  - Open vSwitch

KVM – Ethernet Routing with NAT



KVM – Ethernet Routing with NAT

- +++
  - Relatively straightforward to set up
  - Does not require additional IP addresses from the network
  - MAC addresses local to KVM host
- ---
  - Inbound connections require additional setup for each mapped port on each guest
  - KVM host is performing packet forwarding
  - Otherwise, there is no inbound connectivity
- Provides a good starting point for gaining familiarity with KVM

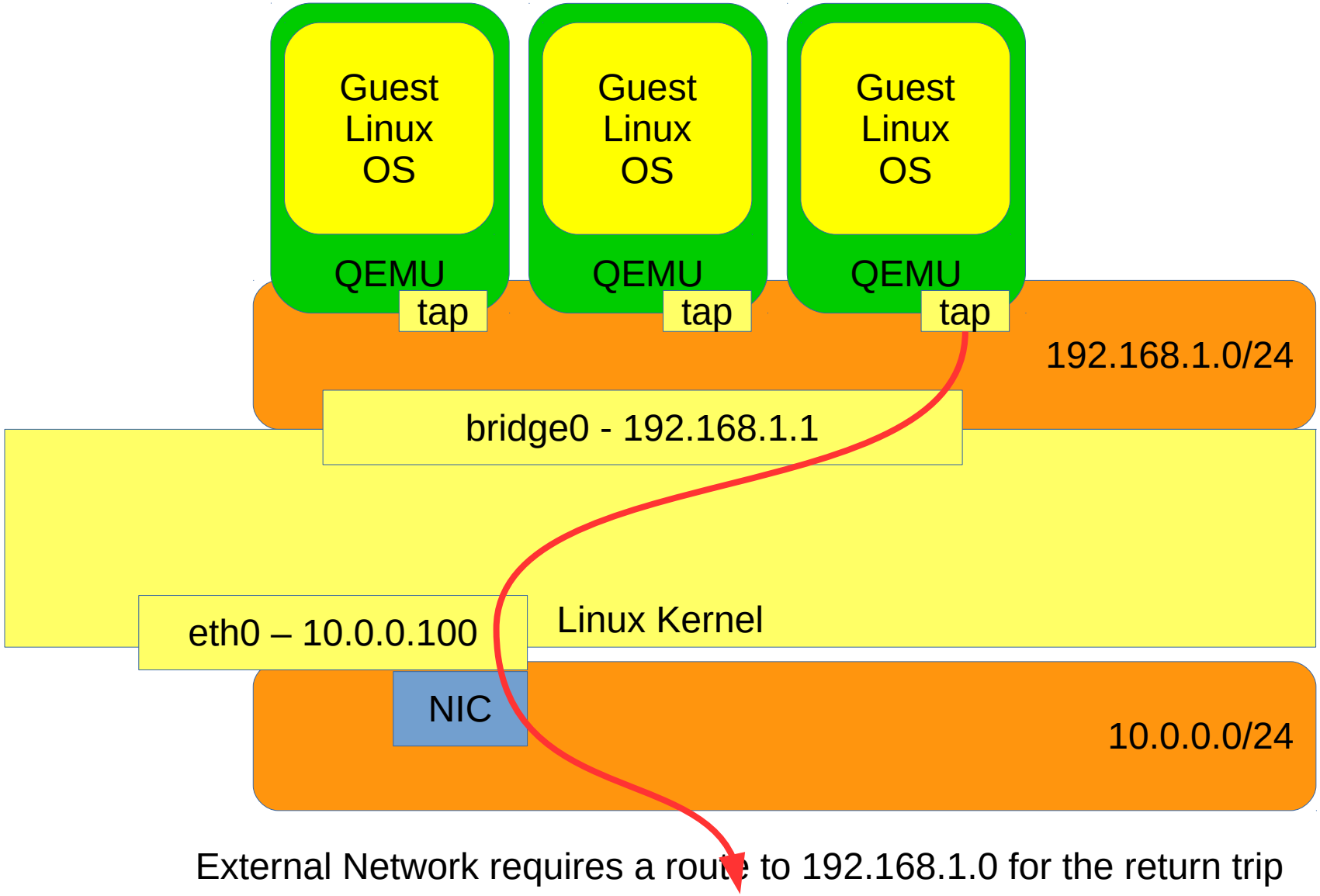
```
<network>
        <name>default</name>
        <bridge name="bridge0" />
        <forward mode="nat"/>
        <ip address="192.168.1.1"
            Netmask="255.255.255.0">
            <dhcp>
                <range start="192.168.1.2"
                       end="192.168.1.254" />
            </dhcp>
        </ip>
</network>
```

Host definition

```
<interface type='network'>
      <mac address='52:54:00:c0:d5:cf'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000'
                bus='0x00' slot='0x06'
                 function='0x0'/>
</interface>
```

Guest definition

KVM – Ethernet Routing without NAT



KVM – Ethernet Routing without NAT

- +++
  - Relatively straightforward to set up
  - Does not require additional IP addresses from the network
  - MAC addresses local to KVM host
- ---
  - Inbound connections require a route for the private network in the external network
  - KVM host is performing packet forwarding

- Provides a good starting point for gaining familiarity with KVM

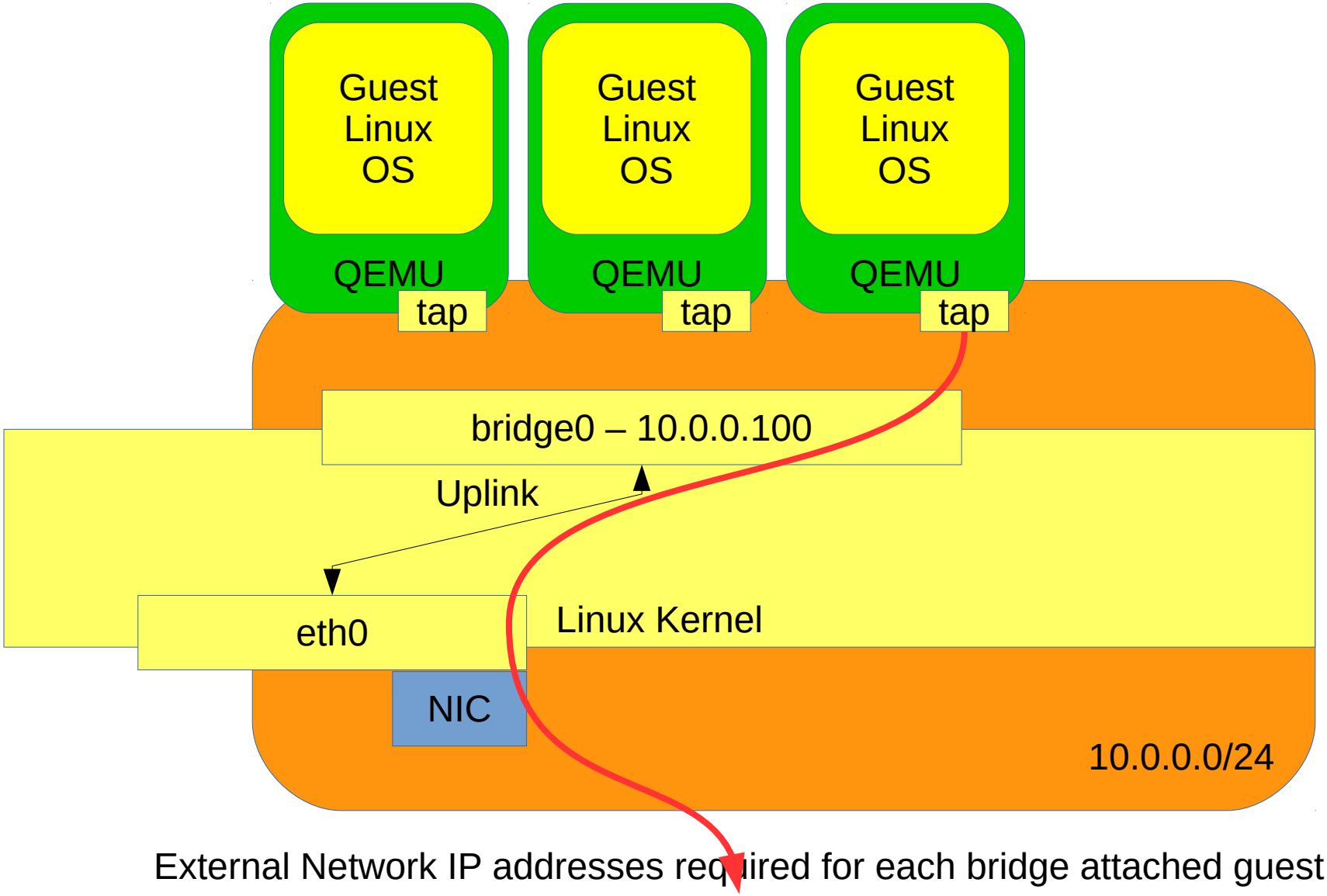
```
<network>
        <name>default</name>
        <bridge name="bridge0" />
        <forward mode="route" dev="eth0"/>
        <ip address="192.168.1.1"
            Netmask="255.255.255.0">
            <dhcp>
                <range start="192.168.1.2"
                       end="192.168.1.254" />
            </dhcp>
        </ip>
</network>
```

Host definition

```
<interface type='network'>
      <mac address='52:54:00:c0:d5:cf'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000'
                bus='0x00' slot='0x06'
                 function='0x0'/>
</interface>
```

Guest definition

KVM – Ethernet Bridging



External Network IP addresses required for each bridge attached guest

KVM – Ethernet Bridging

- +++
  - Relatively straightforward to set up
  - Very common KVM networking for production environments
  - No external routing changes required
- ---
  - KVM host is performing packet forwarding
  - MAC address management required since MACs travel to the external network

- More or less the defacto network standard for KVM until recently

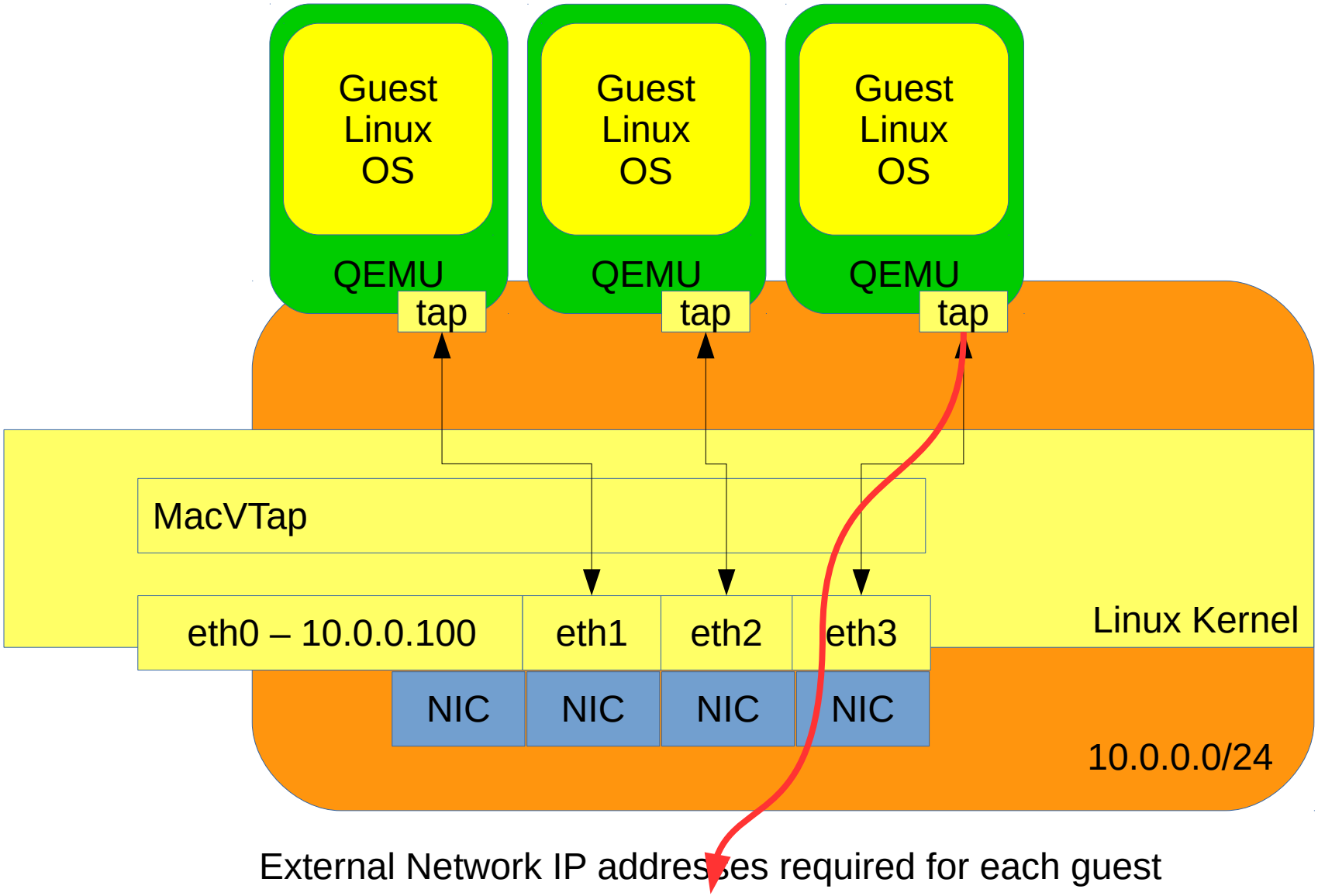
```
<network>
        <name>host-bridge</name>
        <bridge name="bridge0" />
        <forward mode="bridge"/>
</network>
```

Host definition

```
<interface type='network'>
      <mac address='52:54:00:c0:d5:cf'/>
      <source network='host-bridge'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000'
               bus='0x00' slot='0x06'
                function='0x0'/>
</interface>
```

Guest definition

KVM – MacVTap


Guest Linux OS
Guest Linux OS
Guest Linux OS
QEMU
QEMU
QEMU
tap
tap
tap
MacVTap
eth0 – 10.0.0.100
eth1
eth2
eth3
Linux Kernel
NIC
NIC
NIC
NIC
10.0.0.0/24
External Network IP addresses required for each guest

KVM – MacVTap

- +++
  - Higher performance than Bridging or Routing or NAT
  - Compatible with the Network teams management tools and processes
    - Supports VEPA, 802.1Qbg
  - No external routing changes required
- ---
  - MAC address management required since MACs travel to the external network
  - New Technology is new
- Fast, but requires kernels 2.6.34 or newer (not in RHEL 6)

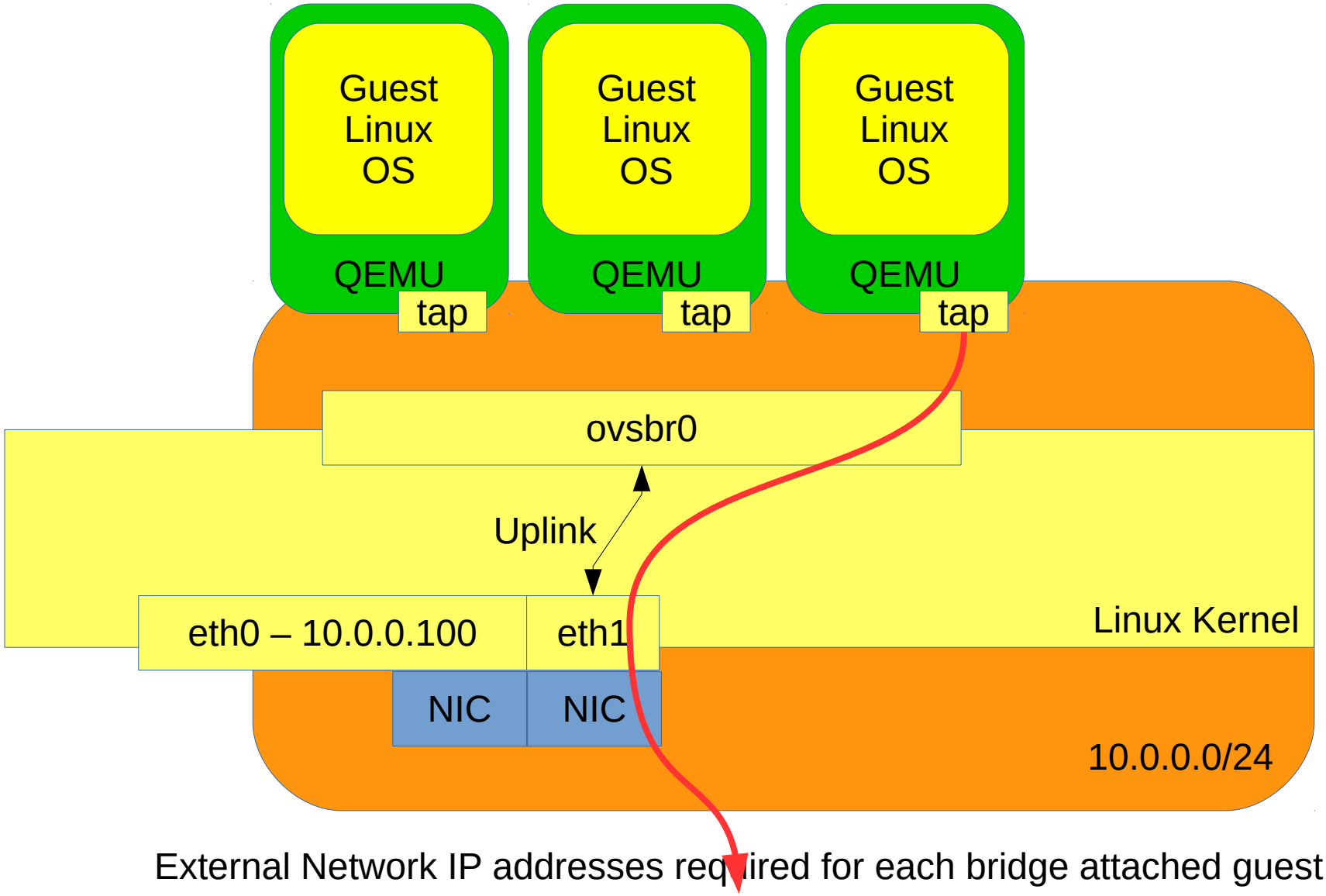
```
<network>
        <name>macvtap-passthru</name>
        <forward mode="bridge">
             <interface dev=”eth1”/>
             <interface dev=”eth2”/>
             <interface dev=”eth3”/>
        </forward>
</network>
```

Host definition

```
<interface type='network'>
      <mac address='52:54:00:c0:d5:cf'/>
      <source network='macvtap-passthru'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000'
               bus='0x00' slot='0x06'
                function='0x0'/>
</interface>
```

Guest definition

KVM – Open vSwitch



External Network IP addresses required for each bridge attached guest

KVM – Open vSwitch

- +++
  - Relatively straightforward to set up
  - Compatible with the Network teams management tools and processes
    - Supports VEPA, 802.1Qbg, VLAN, and more!
  - No external routing changes required
- ---
  - KVM host is performing packet forwarding
  - MAC address management required since MACs travel to the external network
- Requires Open vSwitch package compatible with running Kernel

```
<network>
        <name>ovs-bridge</name>
        <bridge name="ovsbr0" />
        <forward mode="bridge"/>
        <virtualport type=”openvswitch”/>
</network>
```

Host definition

```
<interface type='network'>
      <mac address='52:54:00:c0:d5:cf'/>
      <source network='ovs-bridge'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000'
               bus='0x00' slot='0x06'
                function='0x0'/>
</interface>
```

Guest definition

Agenda

KVM Review

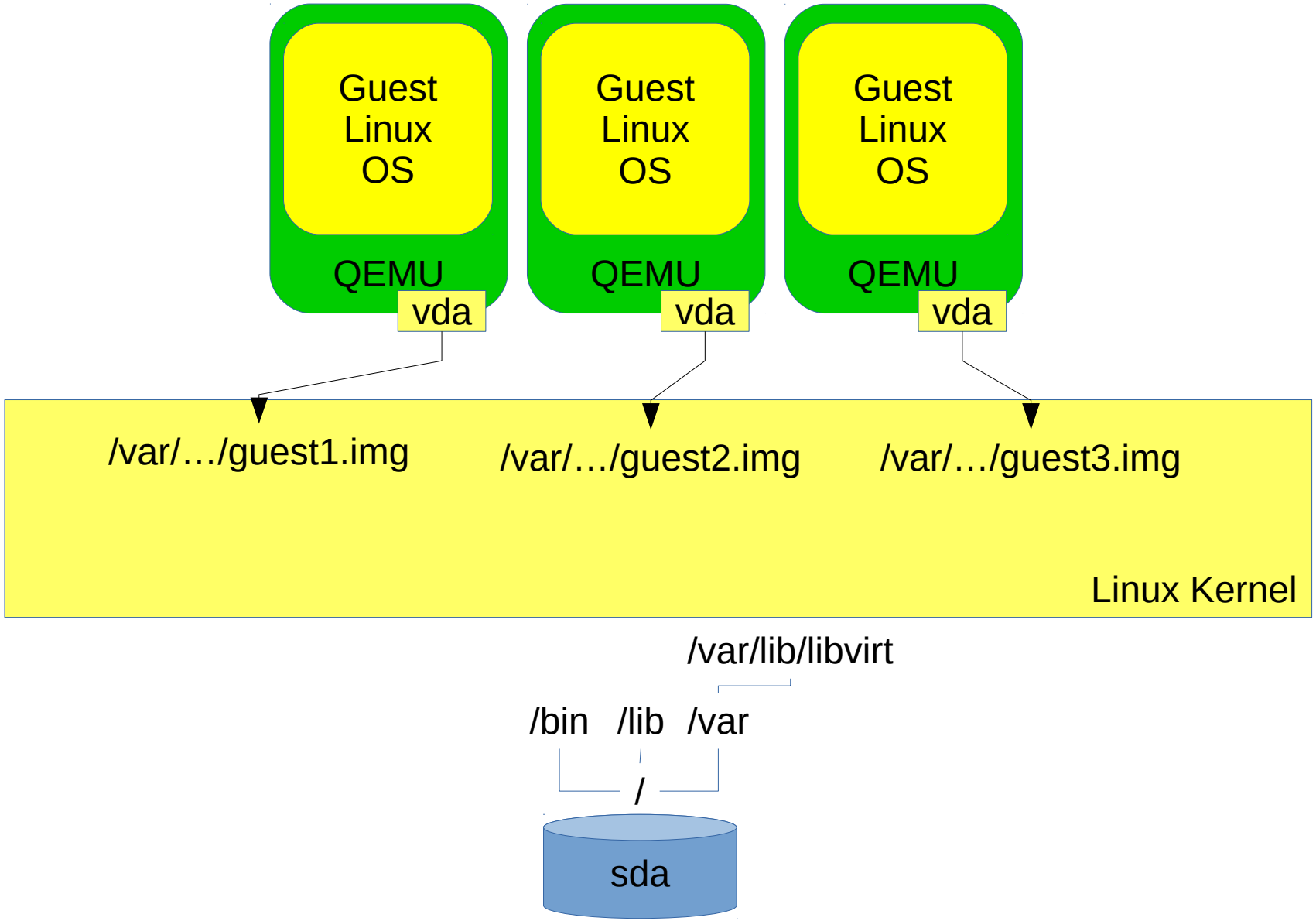
Virtualizing Networks with KVM

Virtualizing Disk with KVM

Virtualizing Disk with KVM

- KVM offers several methods to virtualize the host's disk, each with their own plusses and minuses
  - Files
    - Raw
    - QCOW2
  - Block Devices
    - Entire devices
    - Partitions
    - Logical Volumes
  - Directories & Network services

KVM – Virtualizing Disk with Raw Files



KVM – Virtualizing Disk with Raw Files

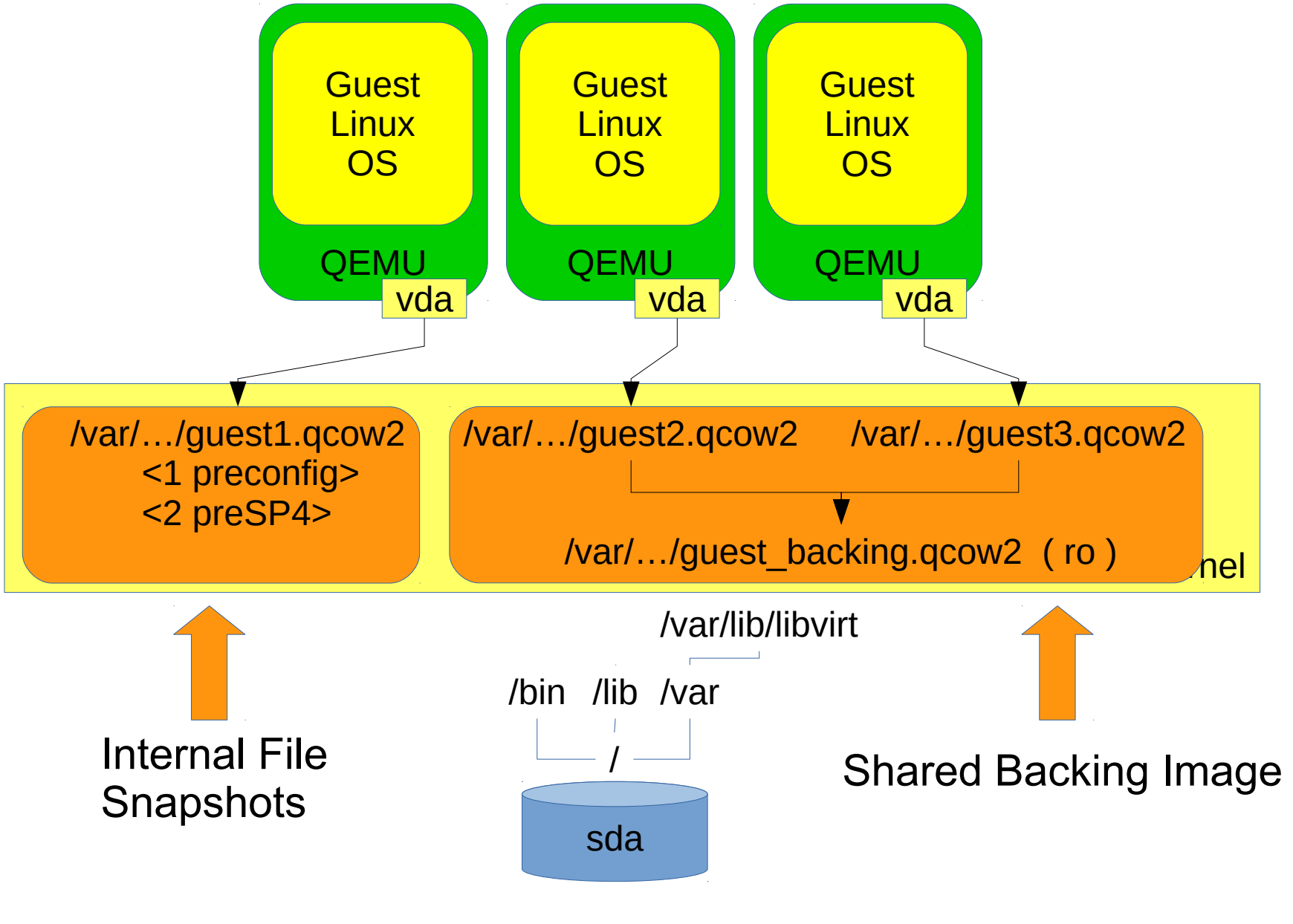
- +++
  - Relatively straightforward to set up and work with
  - No need to ask the storage group to clone disks for you
  - Less overhead than QCOW2
- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - Not as fast as block devices

- Provides a good starting point for gaining familiarity with KVM

```
<disk type='file' device='disk'>
      <driver name='qemu' type='raw'/>
      <source file='/var/lib/libvirt/images/sl12sp0.img'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition

KVM – Virtualizing Disk with QCOW2 Files



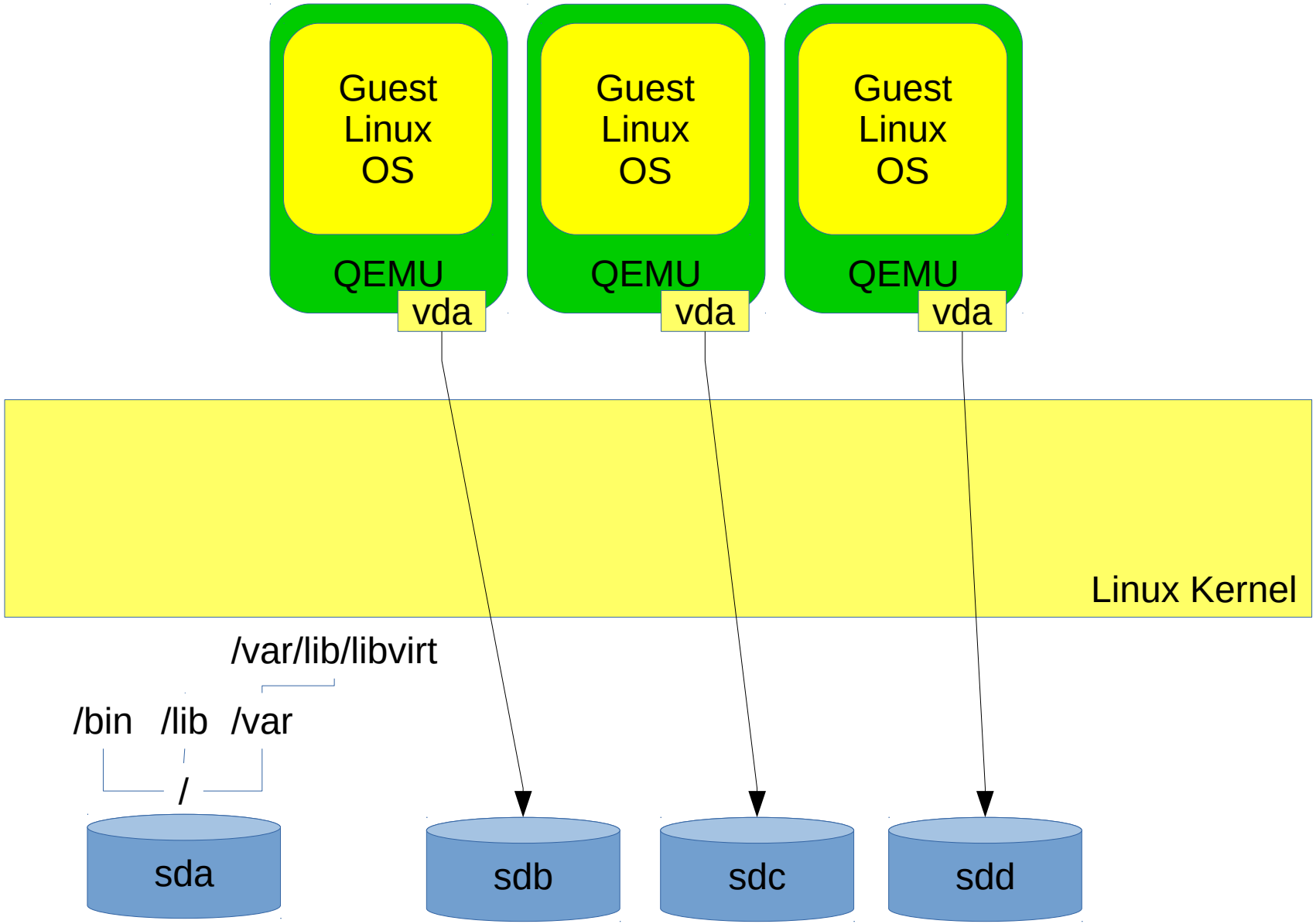
KVM – Virtualizing Disk with QCOW2 Files

- +++
  - Thin Provisioning
  - Snapshots and Base Images
  - No need to ask the storage group to clone disks for you
- ---
  - Overhead
  - Base Images introduce the possibility for data loss if the base is written to
  - Not nearly as fast as block devices
- Provides extensive possibilities for cloning and extending OS images and moving them to production

```
<disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/var/lib/libvirt/images/sl12sp0.qcow2'/>
      <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition

KVM – Virtualizing Disk with entire Block Devices



KVM – Virtualizing Disk with entire Block Devices

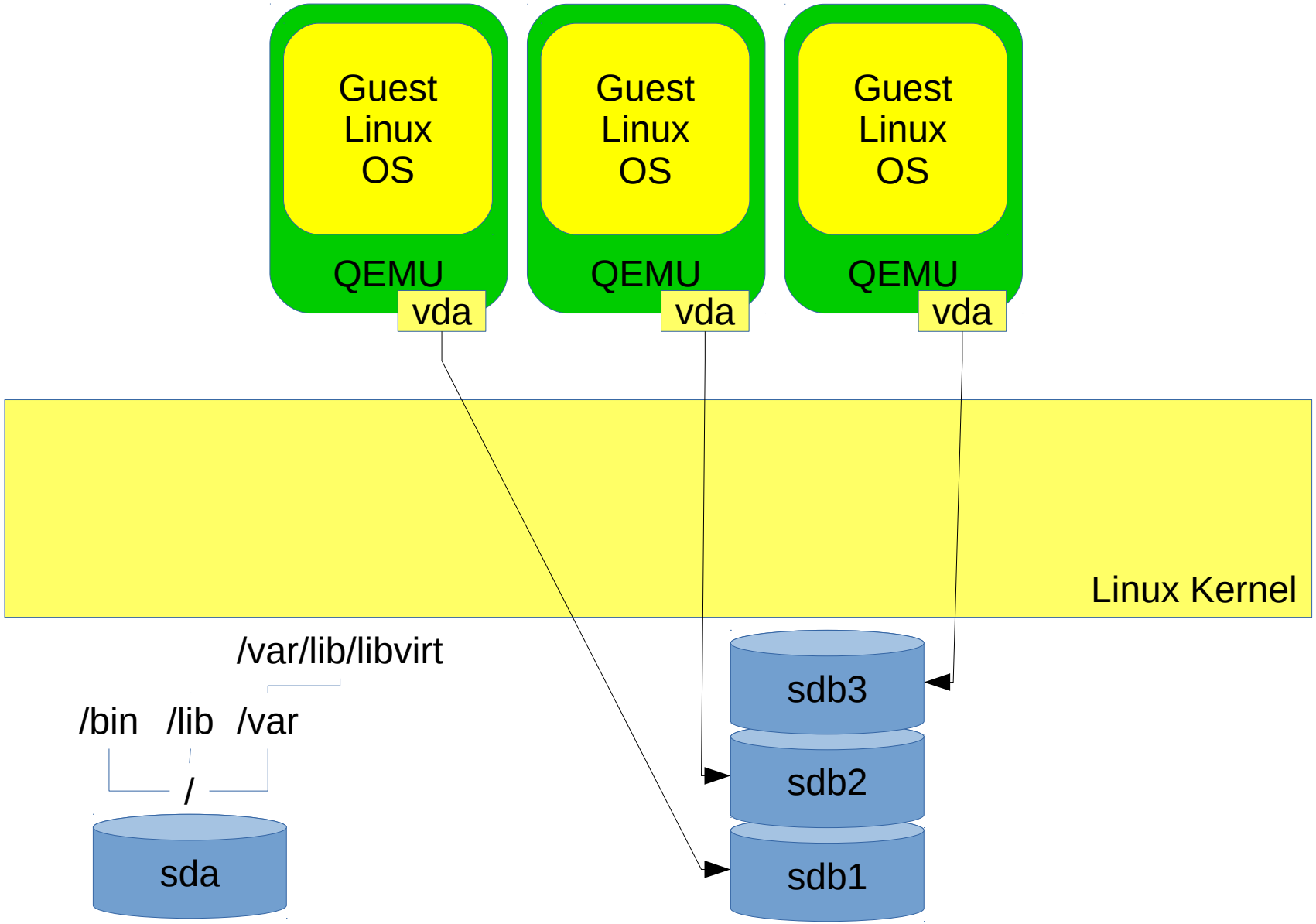
- +++
  - Fast
  - Flashcopy et all do exactly what we want them to do
  - Compatible with existing DR processes
- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - Requires interaction with Storage admins

- Put Databases here

```
<disk type='block' device='disk'>
      <driver name='qemu' type='raw'/>
      <source dev='/dev/sdb'/>
      <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition

KVM – Virtualizing Disk with Partitions



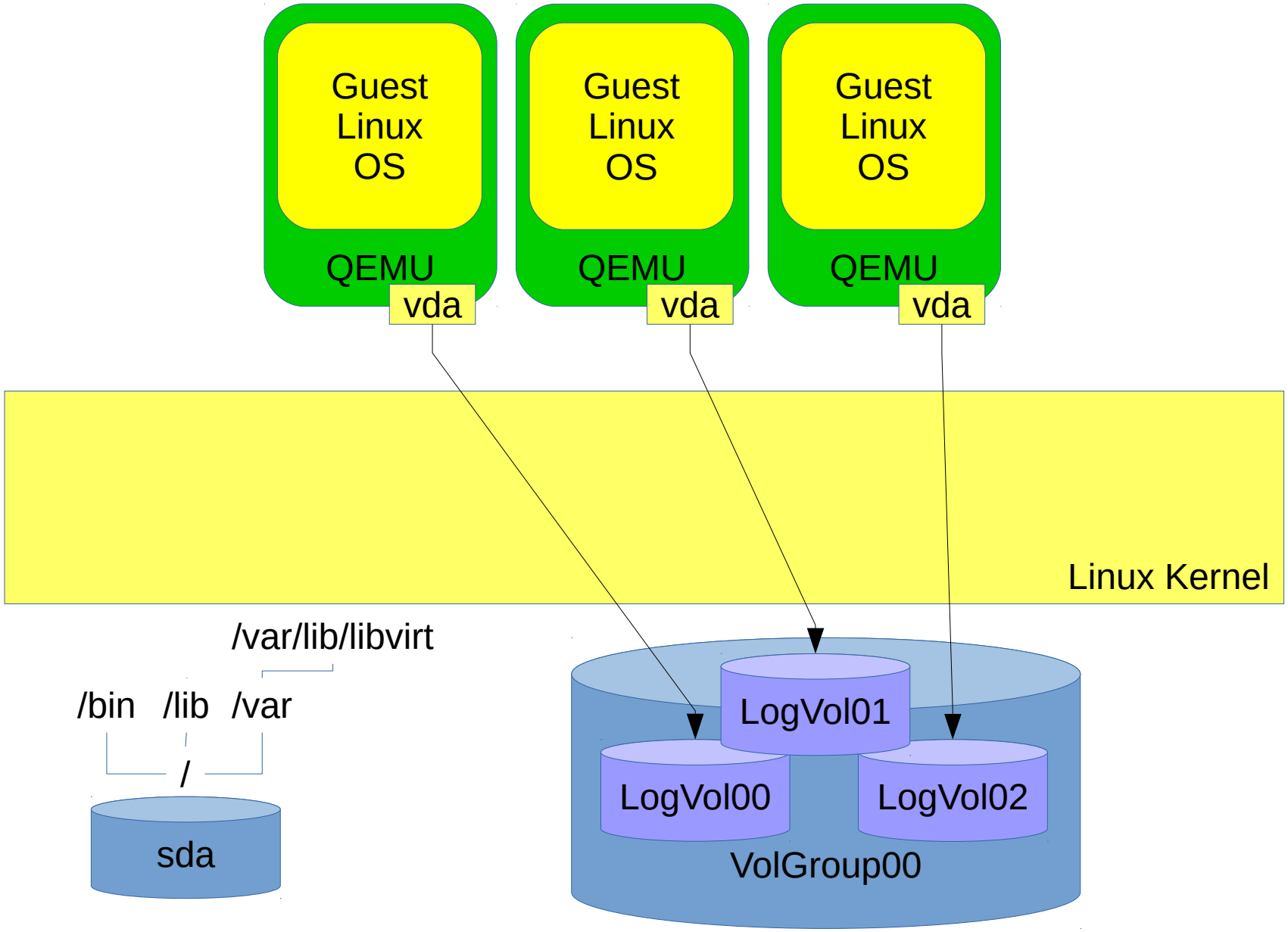
KVM – Virtualizing Disk with Partitions

- +++
  - Fast
  - Flashcopy et all do mostly what we want them to do
  - Compatible with existing DR processes
- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - Requires interaction with Storage admins
  - Partition table maintenance can be tricky and require a boot to refresh
- Make efficient use of very large storage volumes

```
<disk type='block' device='disk'>
      <driver name='qemu' type='raw'/>
      <source dev='/dev/sdb1'/>
      <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition

KVM – Virtualizing Disk with Logical Volumes



KVM – Virtualizing Disk with Logical Volumes

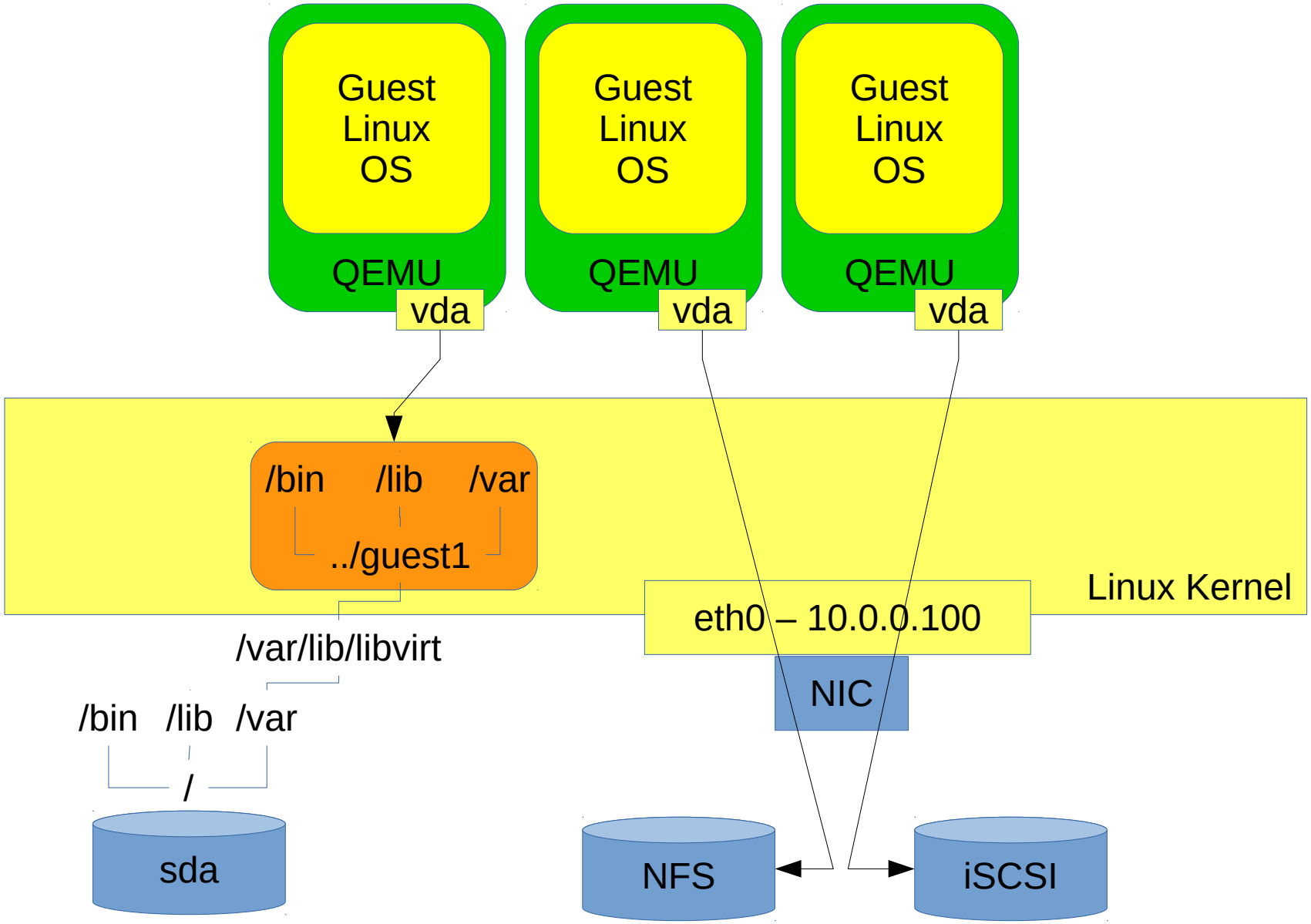
- +++
  - Fast
  - Linux LVM has its own snapshot function
  - Very Flexible and Dynamic, managable from within libvirt
- ---
  - Thick provisioning (lots of 0x0000 on disk)
  - LVM snapshots are kinda slow and drive lots of host IO

- Most flexible and performant disk management approach

```
<disk type='block' device='disk'>
      <driver name='qemu' type='raw'/>
      <source dev='/dev/VolGroup00/LogVol00'/>
      <target dev='vda' bus='virtio'/>
</disk>
```

Guest definition

KVM – Virtualizing Disk with Directories and Services



KVM – Virtualizing Disk with Directories and Services

- +++
  - Flexible
  - Diskless Guests
- ---
  - ??

```
<disk type='network' device='lun'>
      <driver name='qemu' type='raw'/>
      <source protocol='iscsi' name='iqn.2013-07.com.example:iscsi-nopool/1'>
        <host name='example.com' port='3260'/>
      </source>
      <auth username='myuser'>
        <secret type='iscsi' usage='libvirtiscsi'/>
      </auth>
      <target dev='sdb' bus='scsi'/>
</disk>

<filesystem type='mount' accessmode='passthrough'>
      <driver type='path' wrpolicy='immediate'/>
      <source dir='/export/to/guest'/>
      <target dir='/import/from/host'/>
      <readonly/>
</filesystem>
```

Guest definition

Agenda

KVM Review

Virtualizing Networks with KVM

Virtualizing Disk with KVM

Questions?