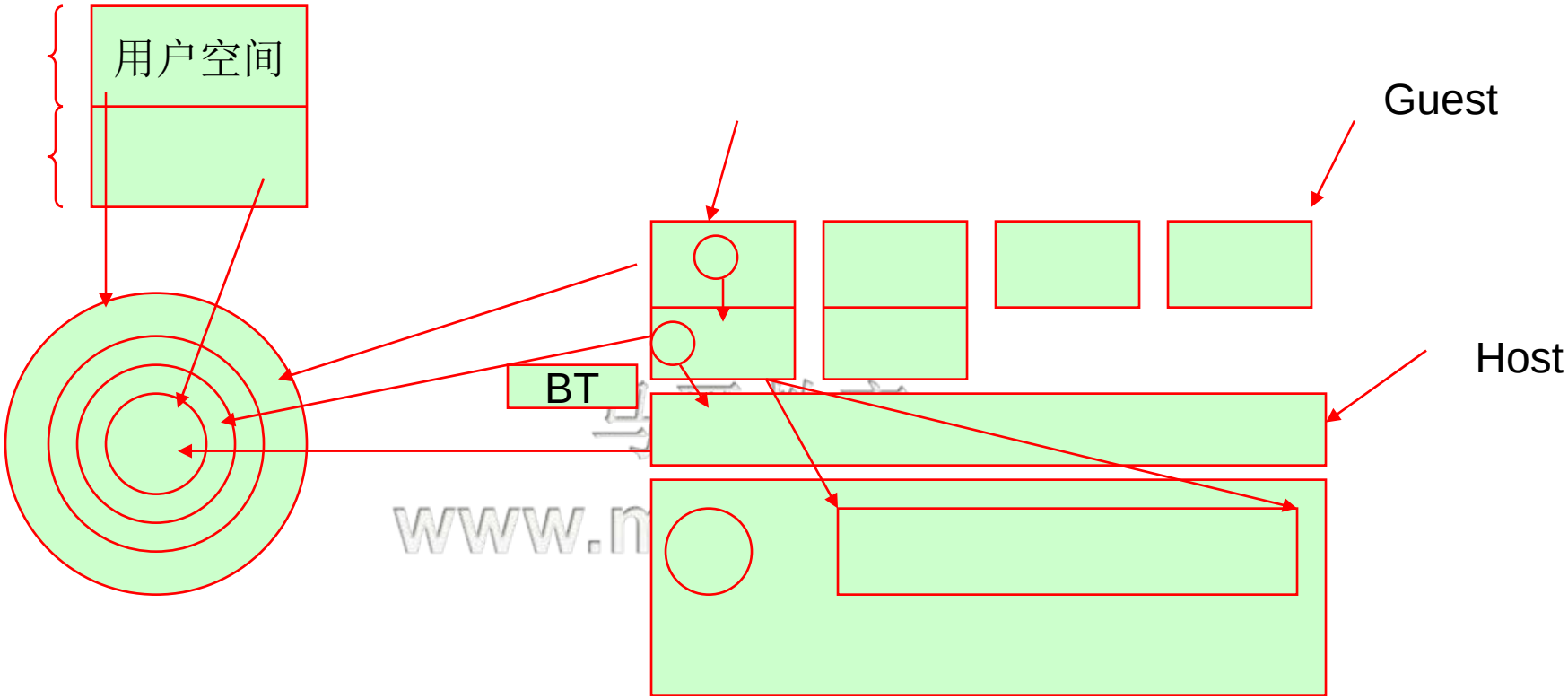
马哥教育

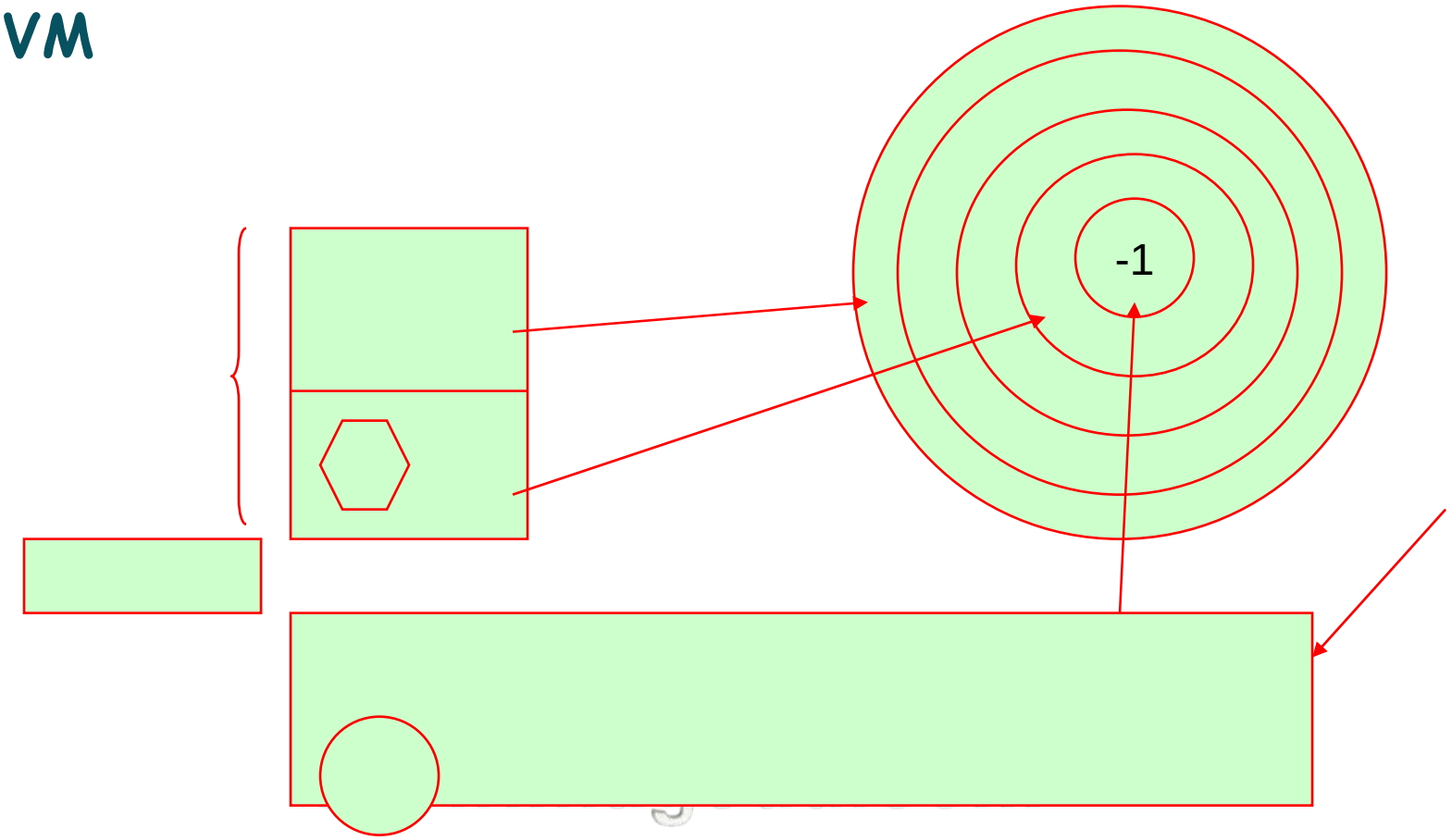
Virtualization

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

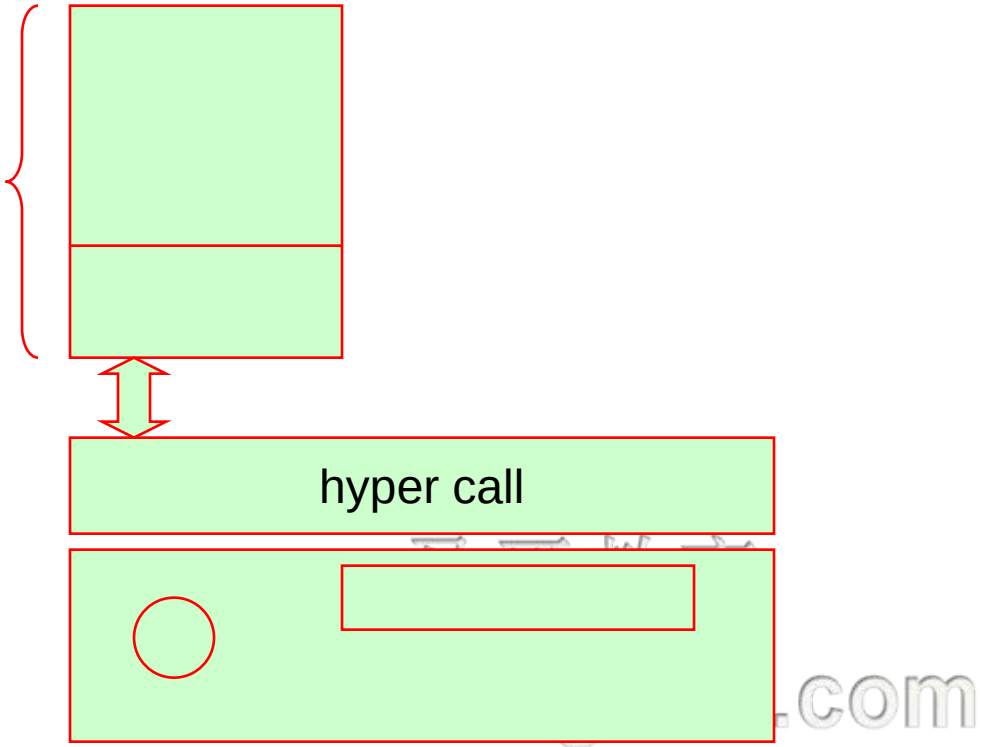
- CPU, Memory, I/O
  - keyboard, monitor

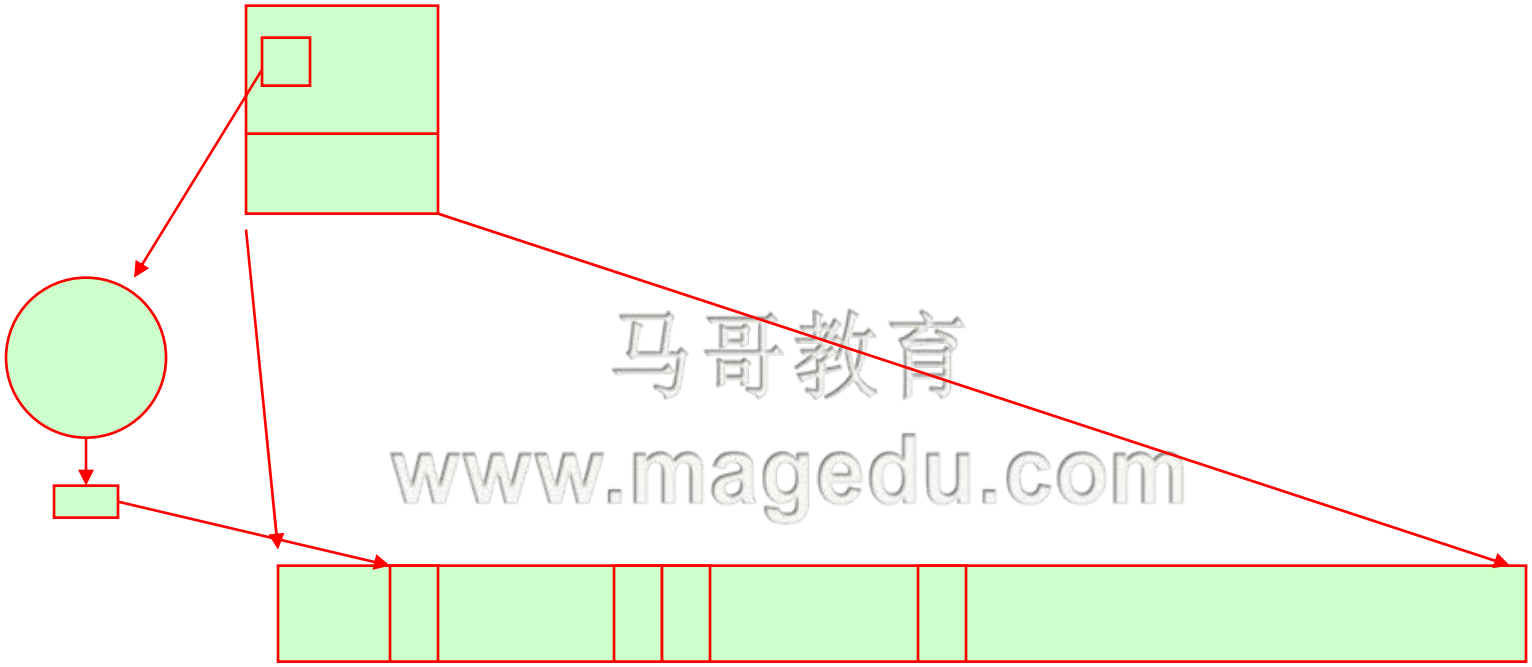


HVM



vm monitor = hypervisor



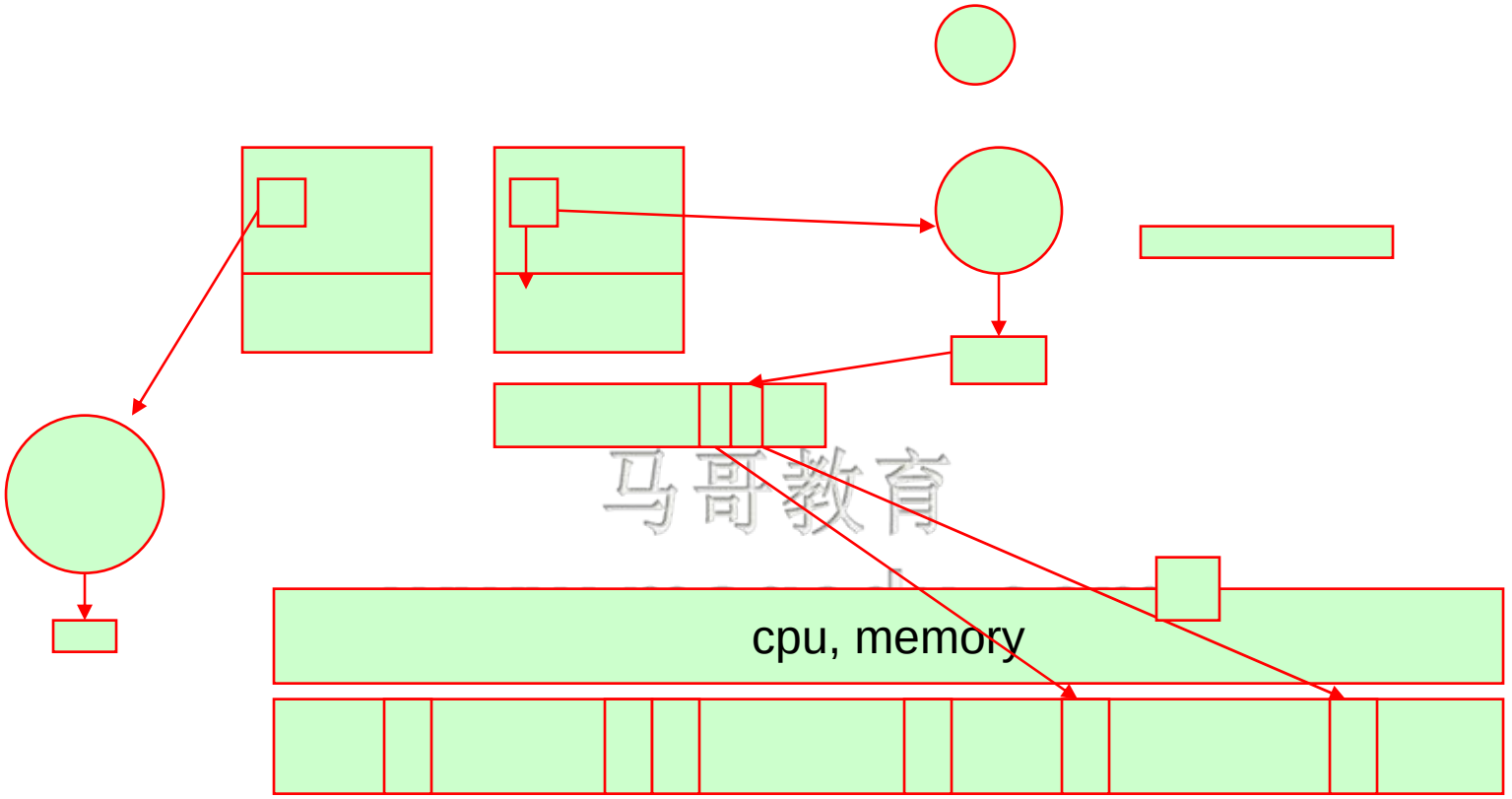


shadow page table

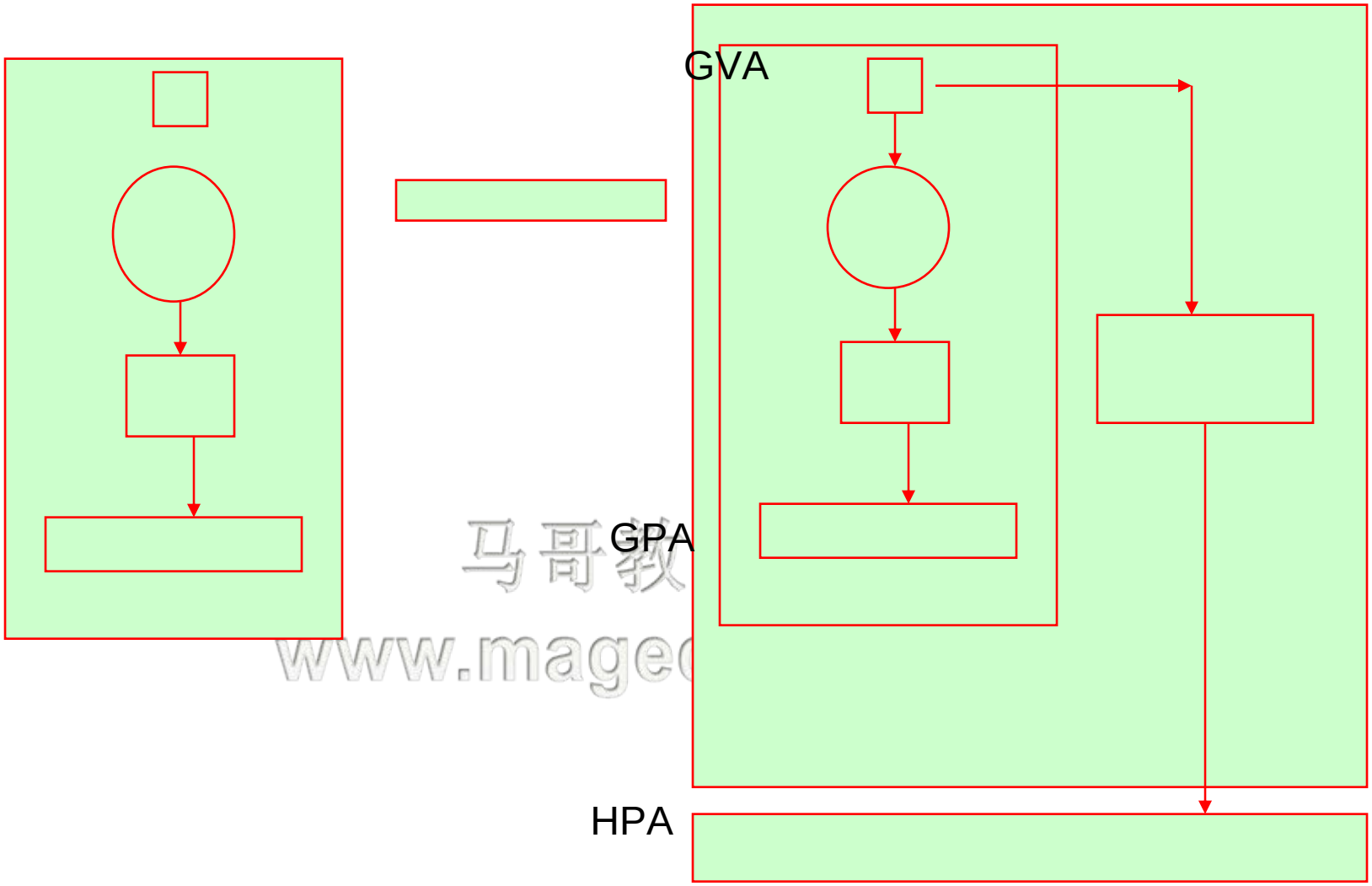
0-512

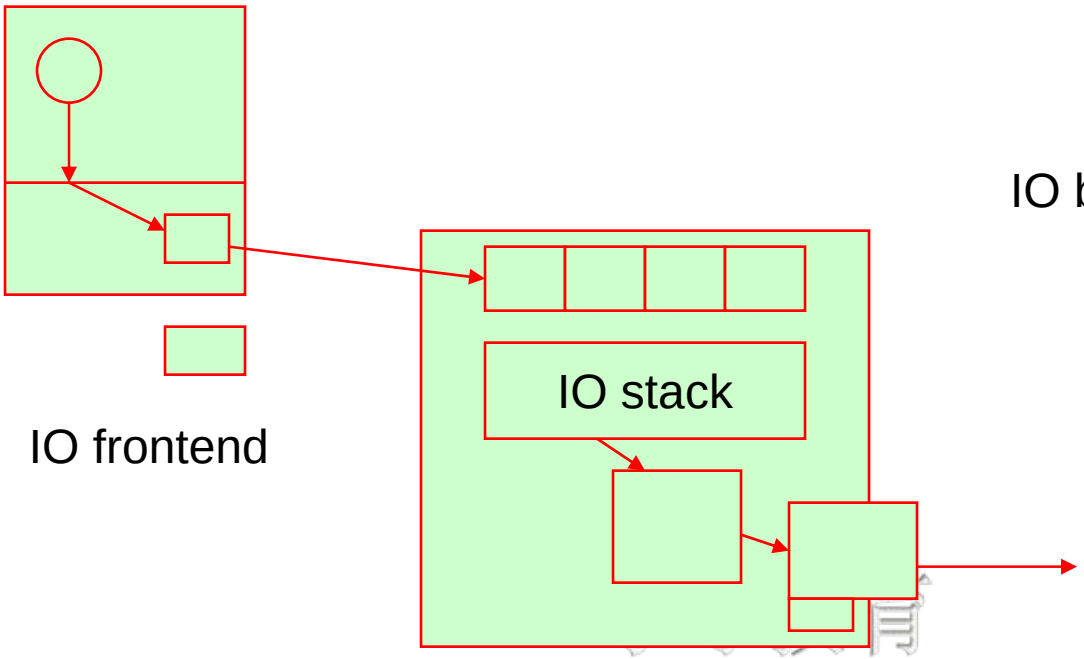
0-512

a → 3page

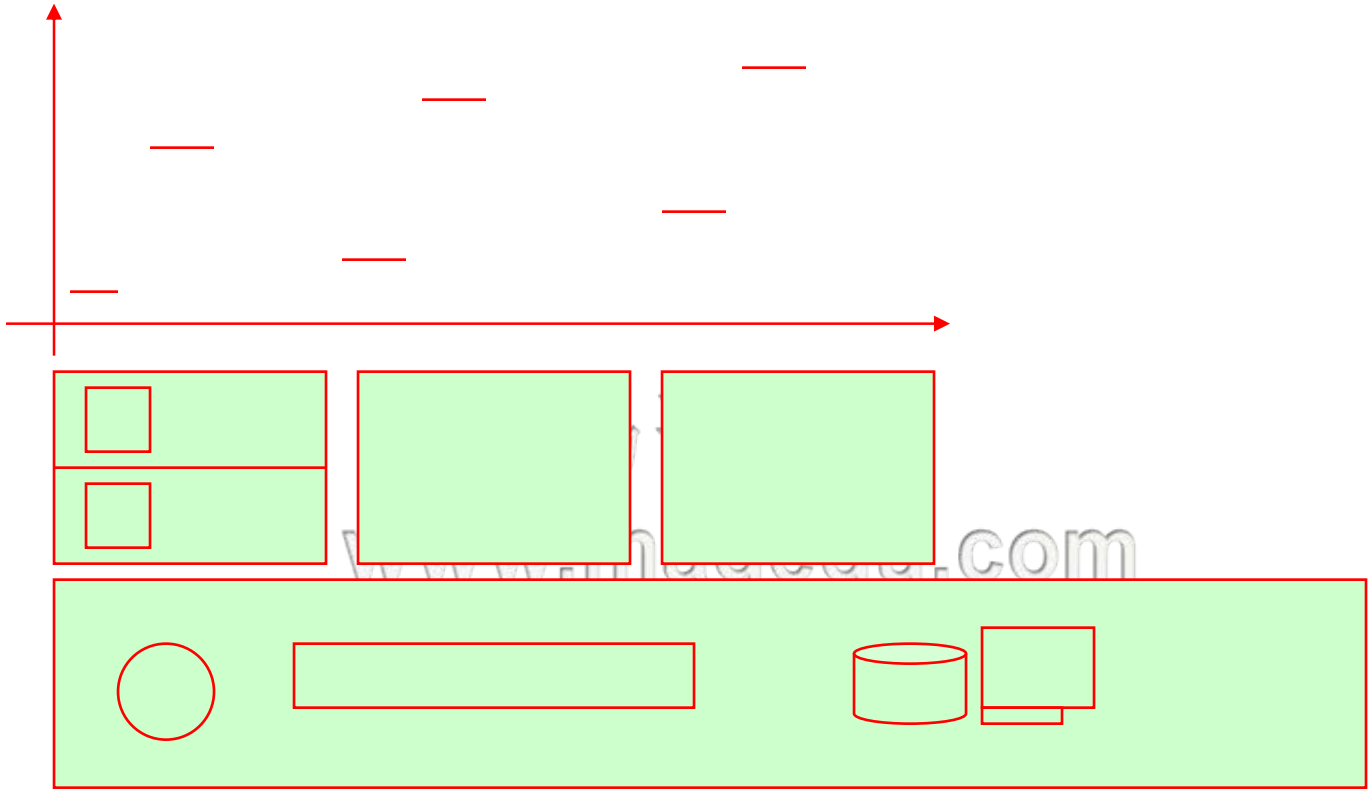


VA → PA

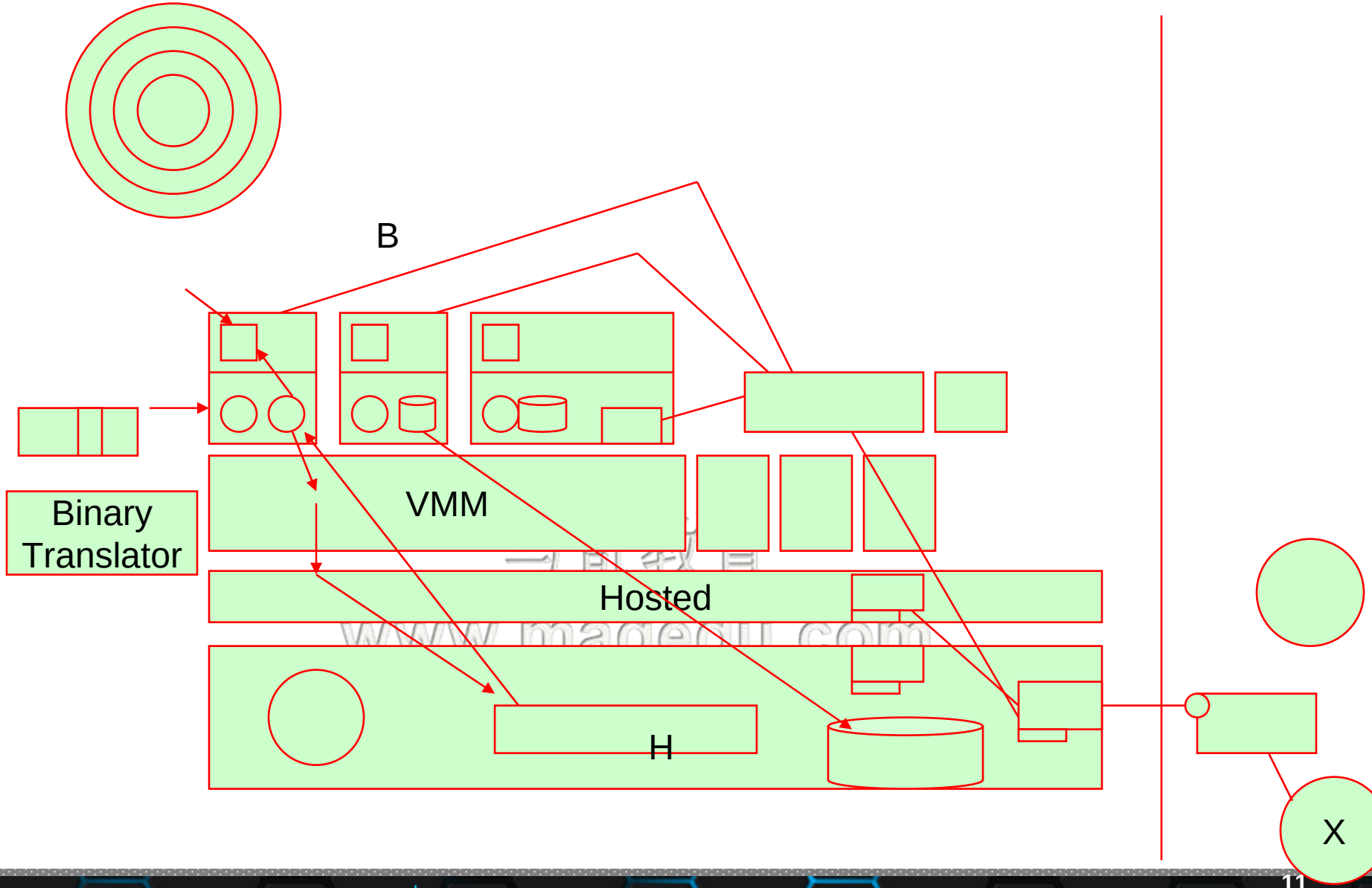




CPU虚拟化

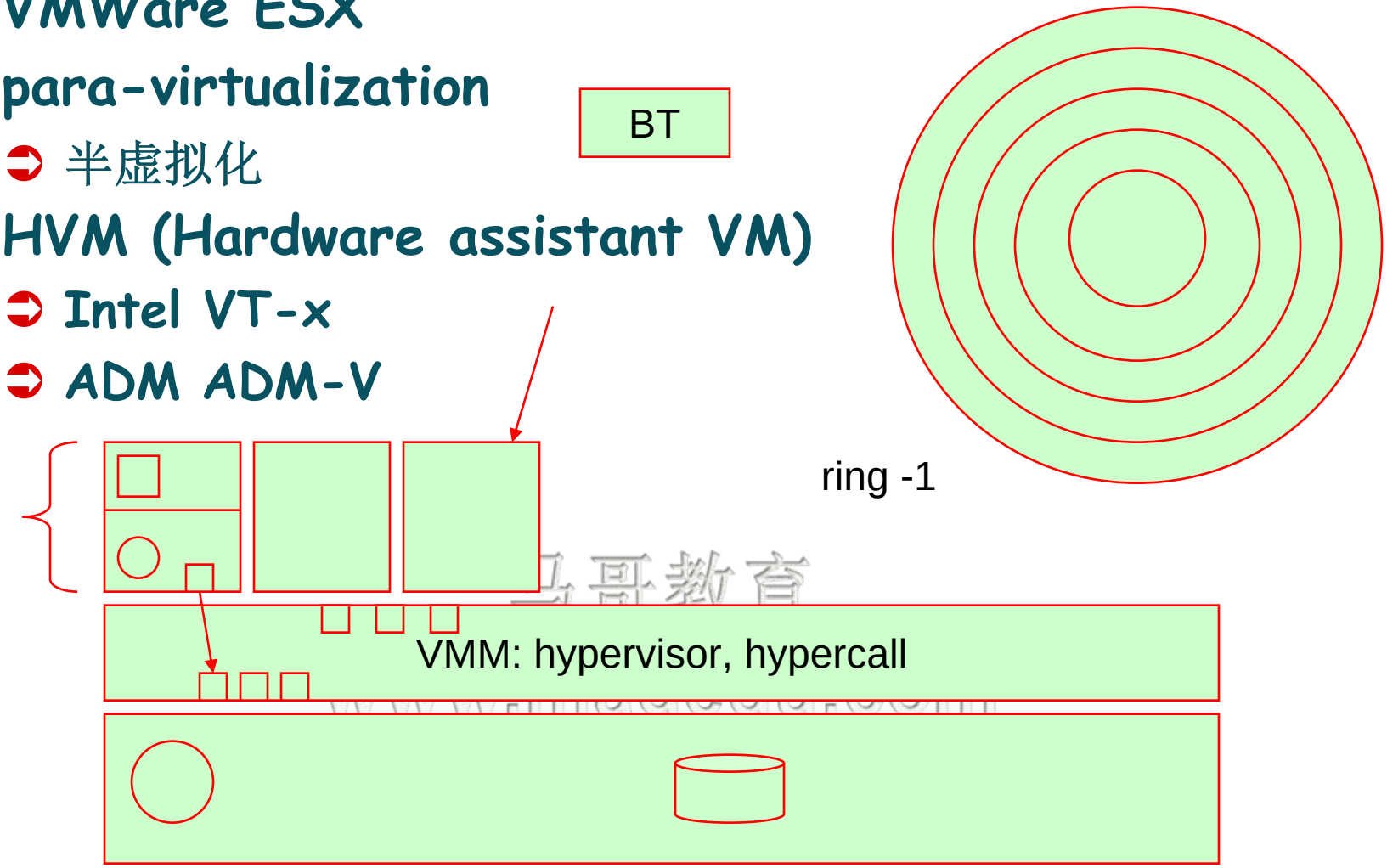


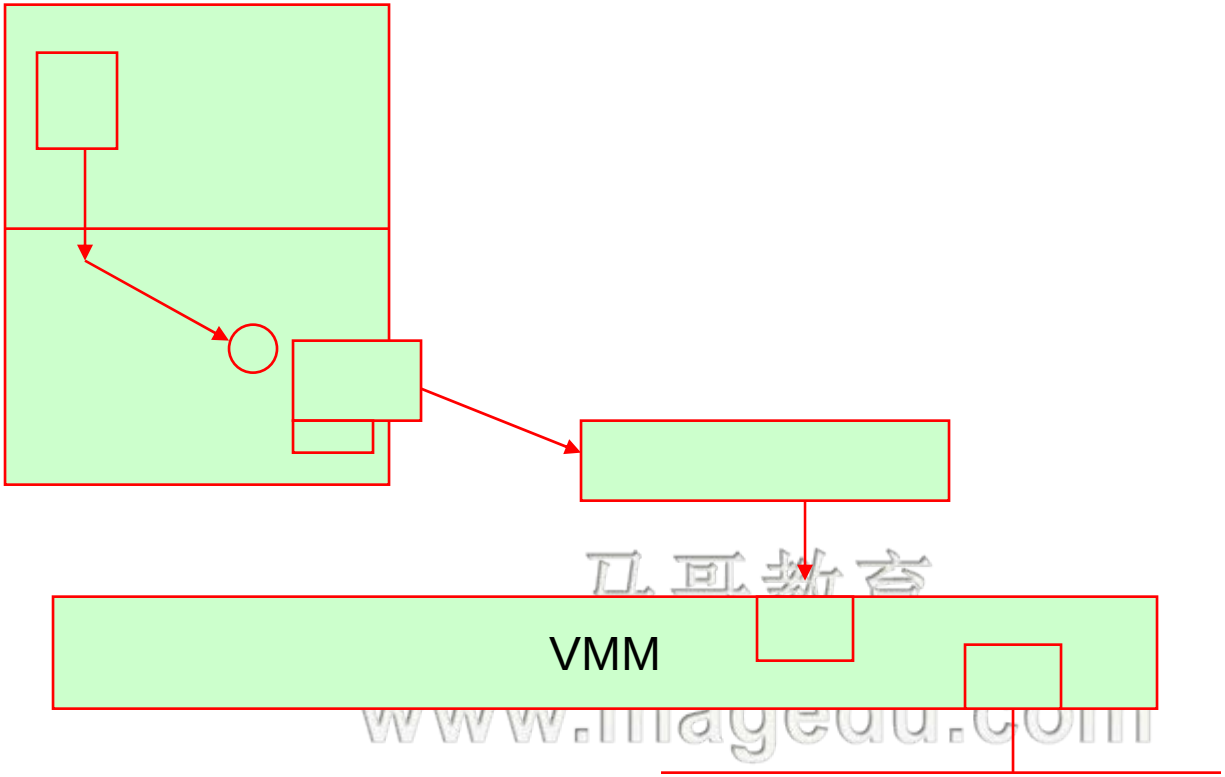
CPU:模拟，emulation，virtualization



虚拟化模型

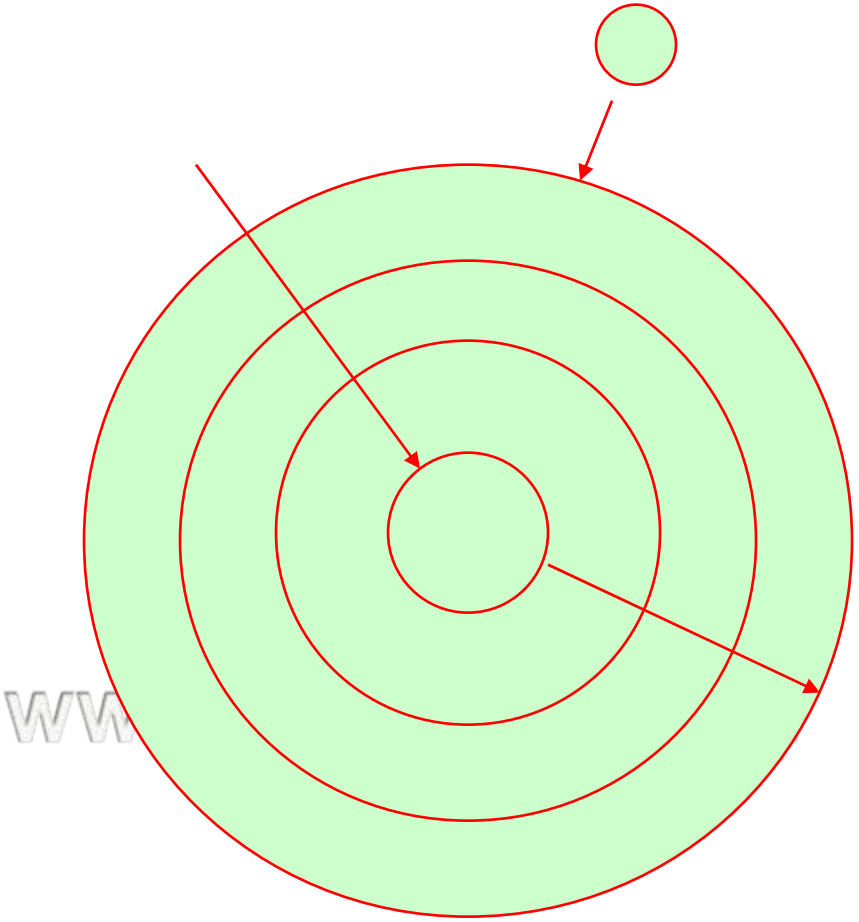
- VMWare ESX
- para-virtualization
  - 半虚拟化
- HVM (Hardware assistant VM)
  - Intel VT-x
  - ADM ADM-V

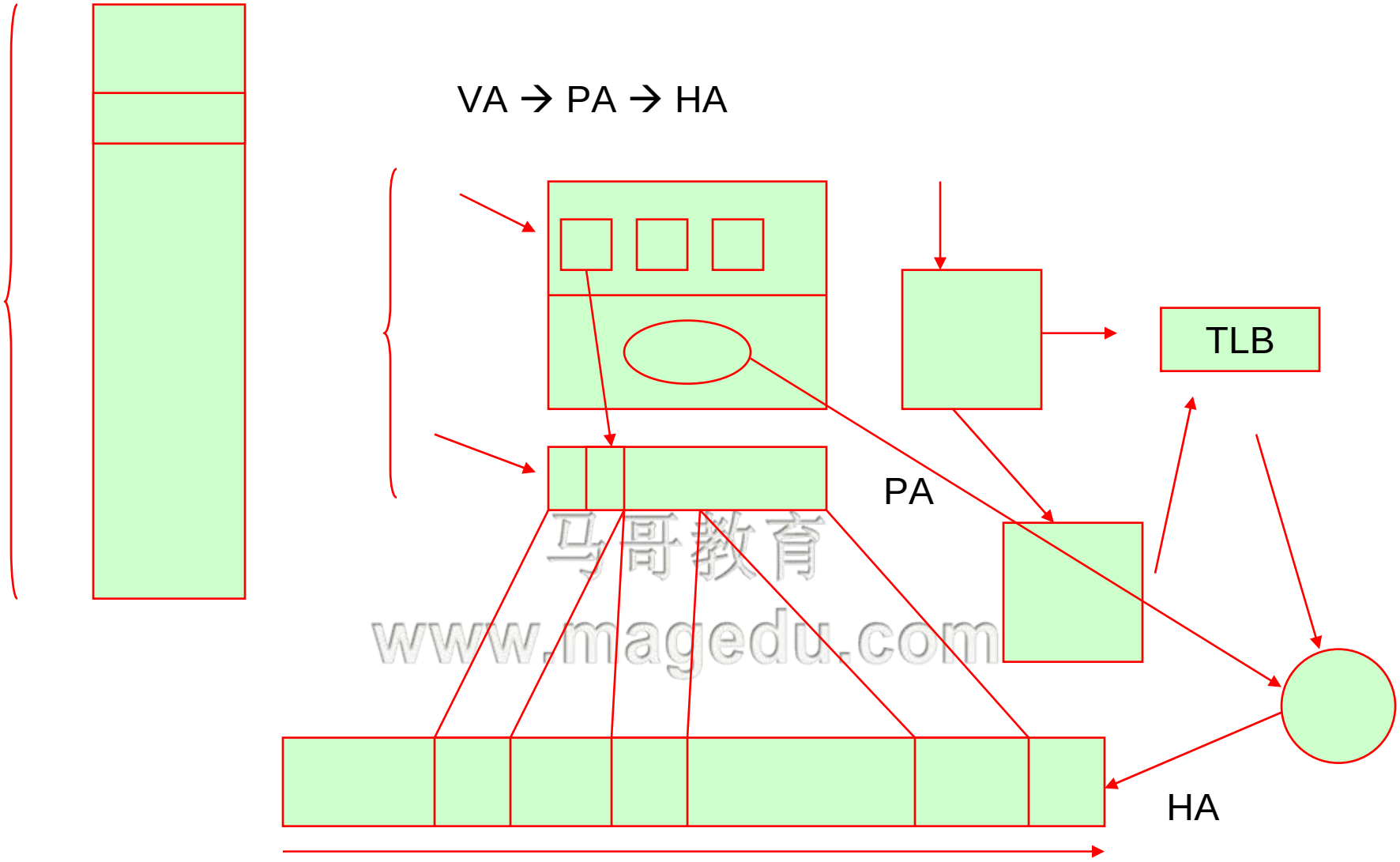




- IO虚拟化
  - 完全虚拟化
  - 半虚拟化
  - IO-through

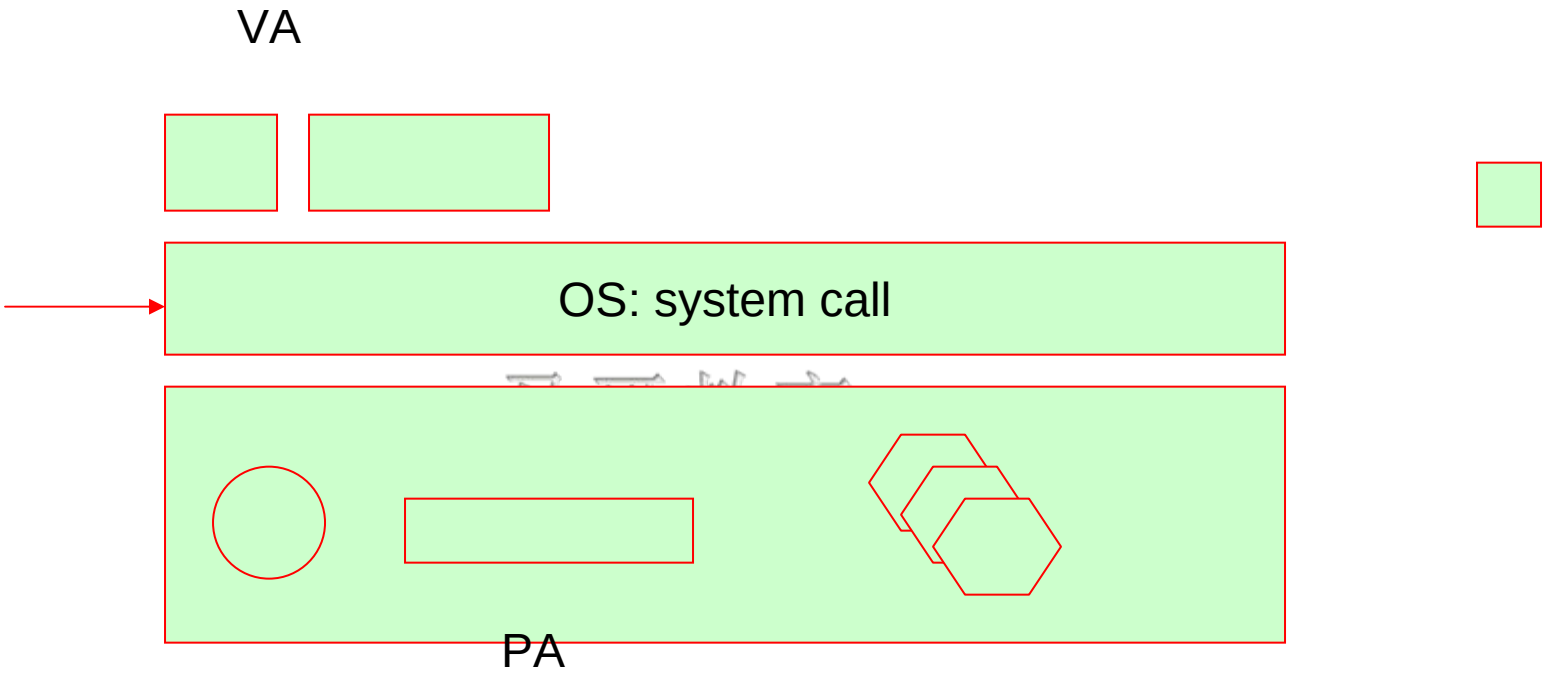
- CPU：普通指令，特权指令，敏感指令





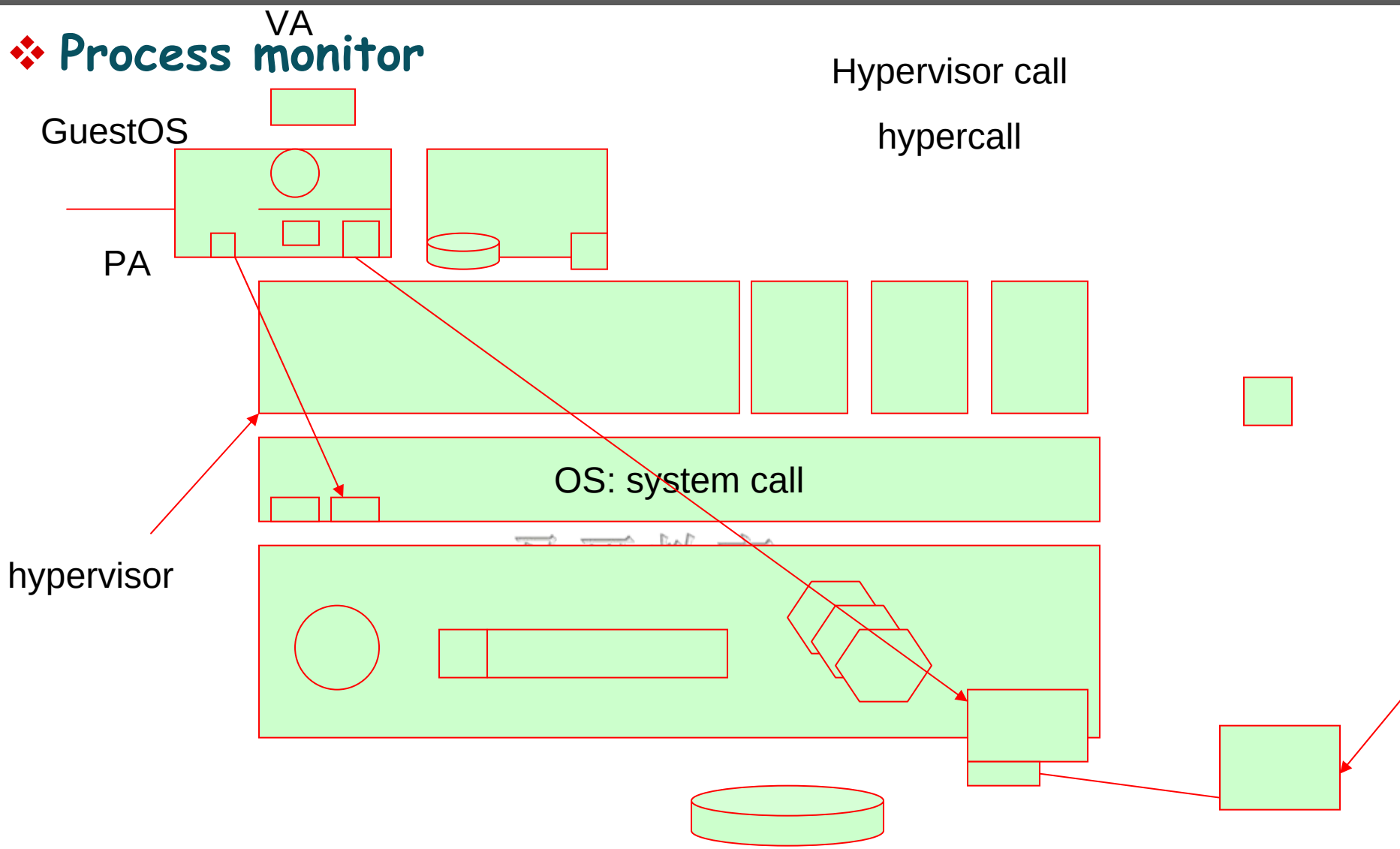
Rss, page cache, anon page

- Process monitor

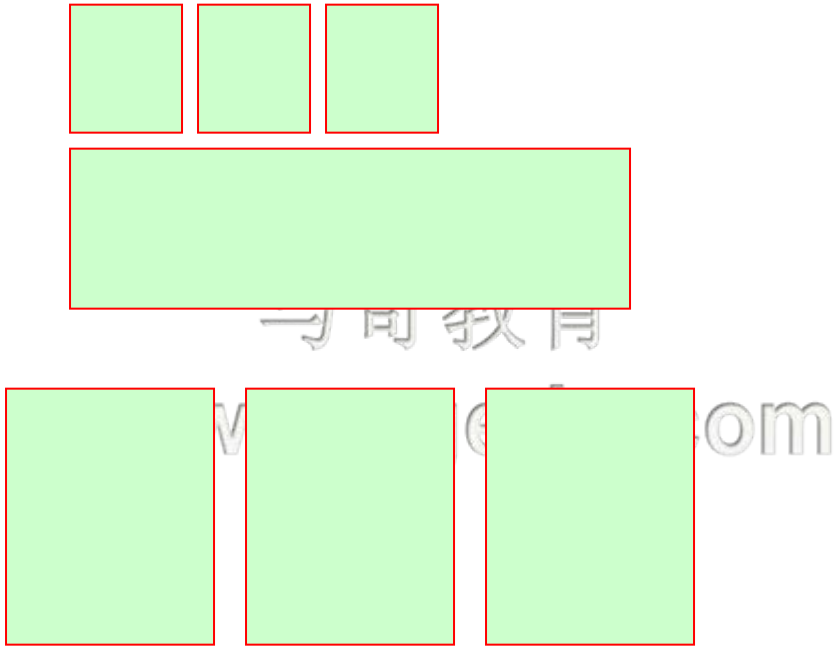


Rss, page cache, anon page

- Process monitor



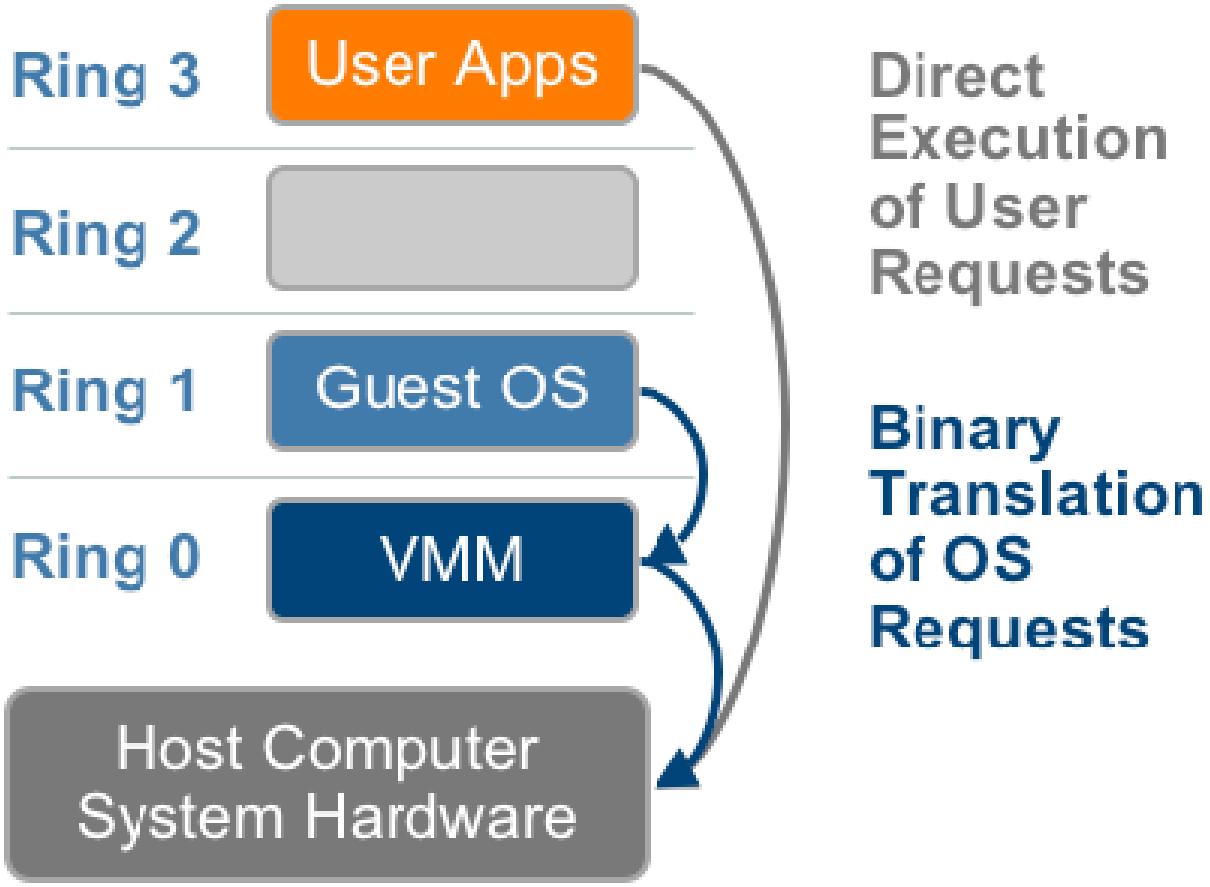
- CPU：全量CPU时间片
- 内存：连续、全部内存空间（0x0000，）
- I/O：全部可用IO



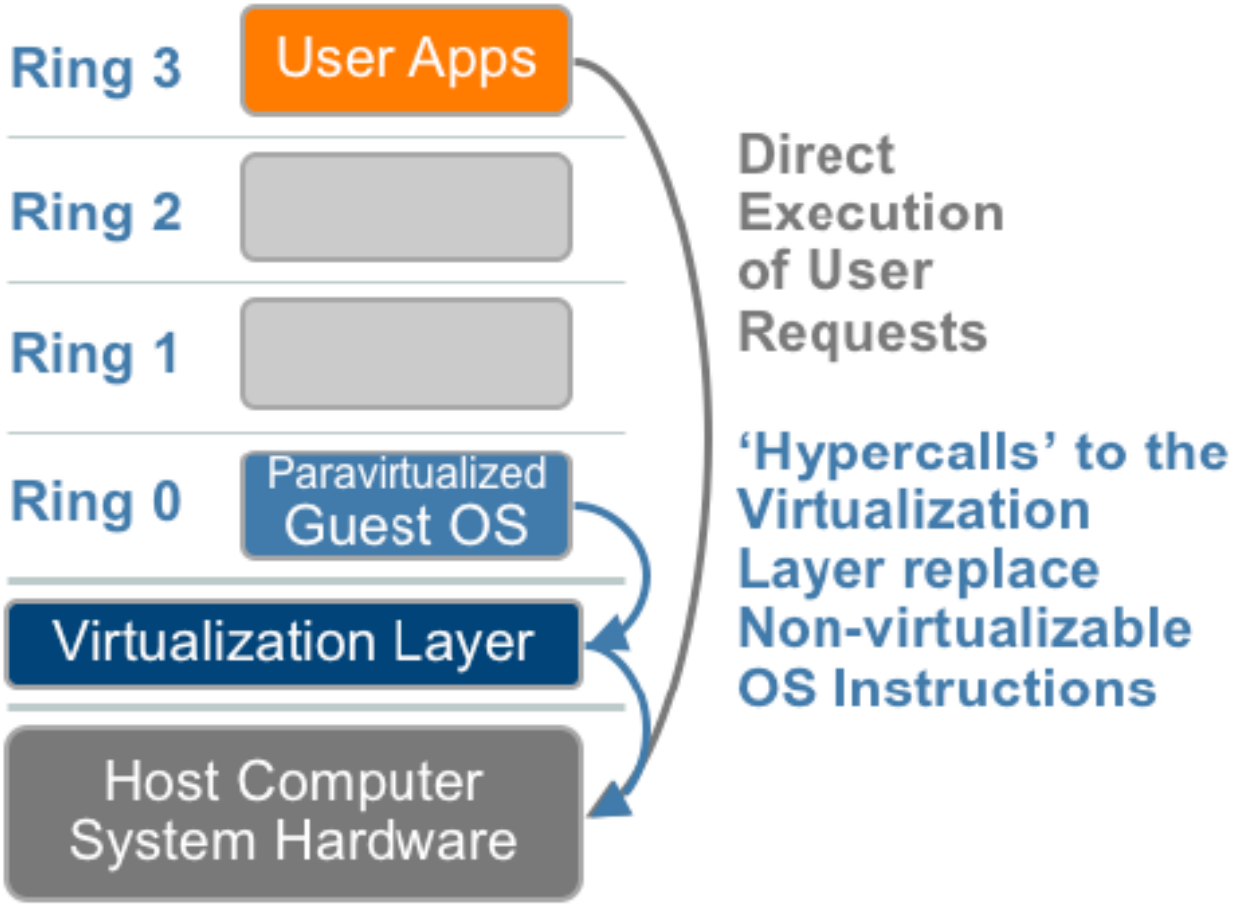
CPU虚拟化

- 基于“二进制翻译(Binary Translation)”技术的完全虚拟化
- 操作系统辅助的虚拟化，也即半虚拟化(paravirtualization)或准虚拟化
- 硬件辅助的虚拟化

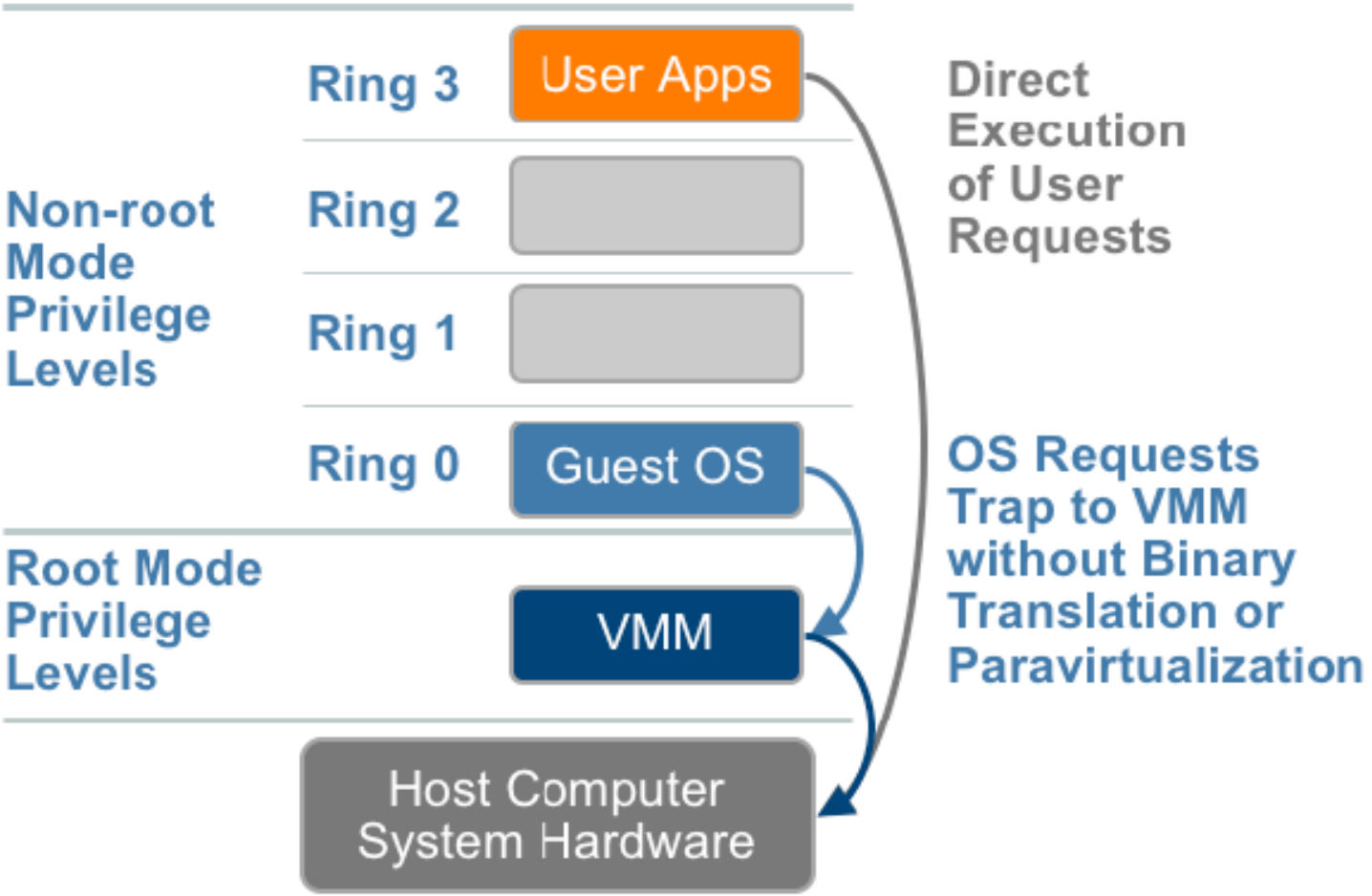
Binary Translation (BT)



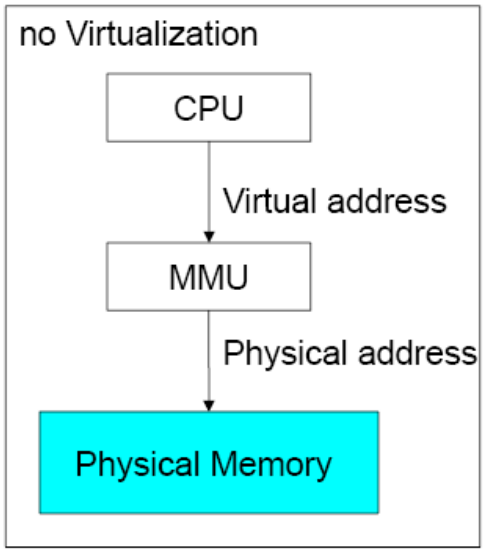
半虚拟化(Paravirtualization)

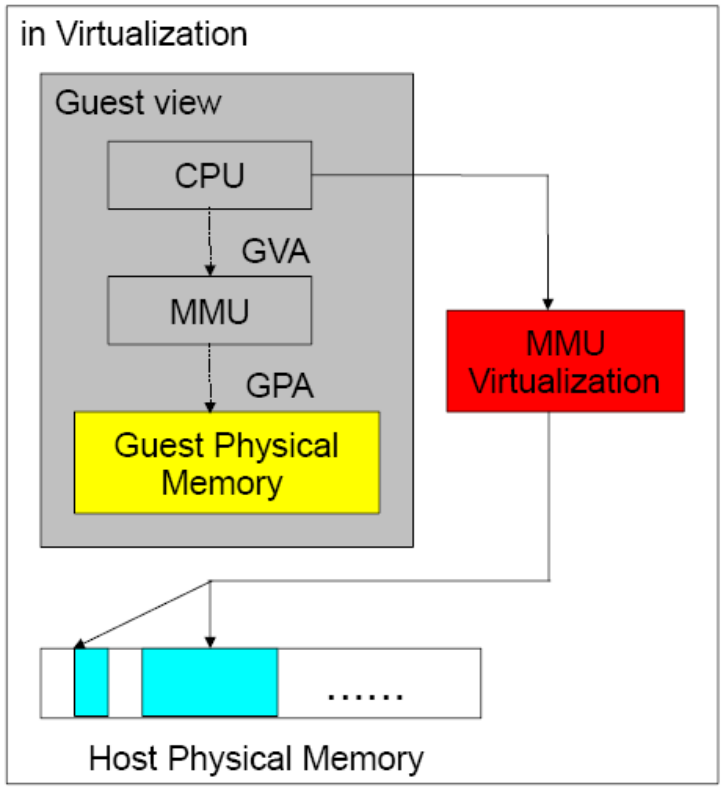


硬件辅助的虚拟化


Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Guest OS
Non-root Mode Privilege Levels
Root Mode Privilege Levels
VMM
Host Computer System Hardware
Direct Execution of User Requests
OS Requests Trap to VMM without Binary Translation or Paravirtualization

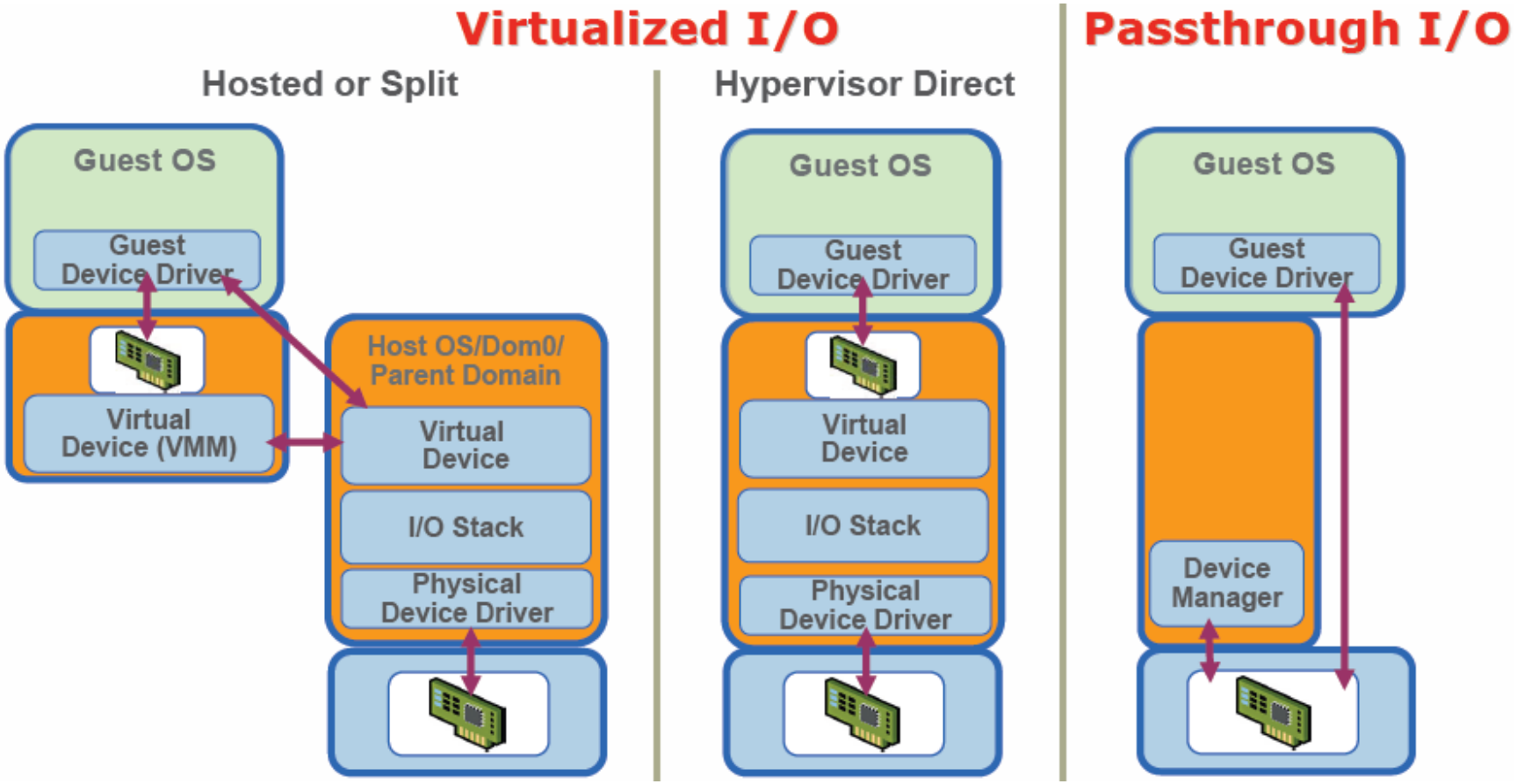
内存虚拟化

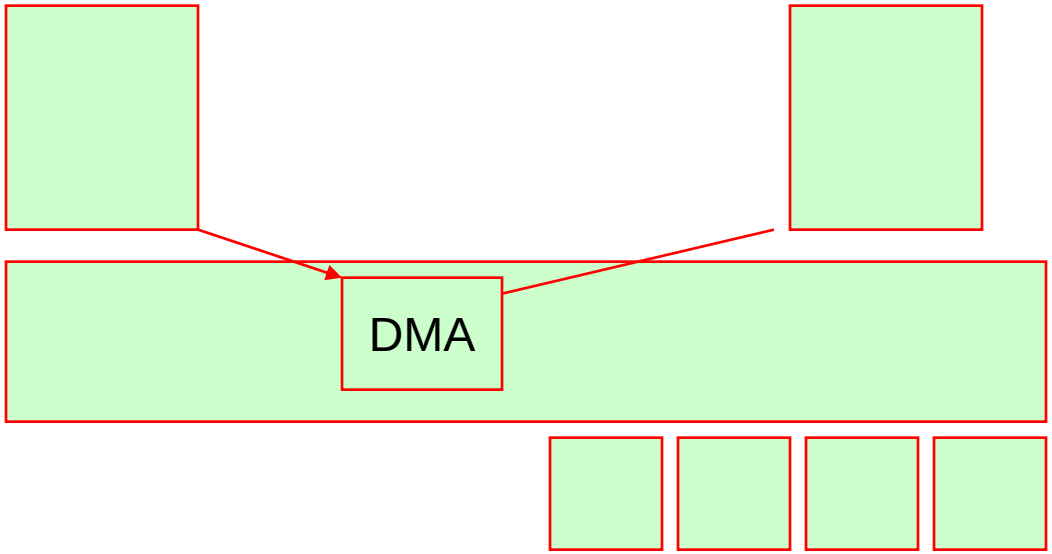




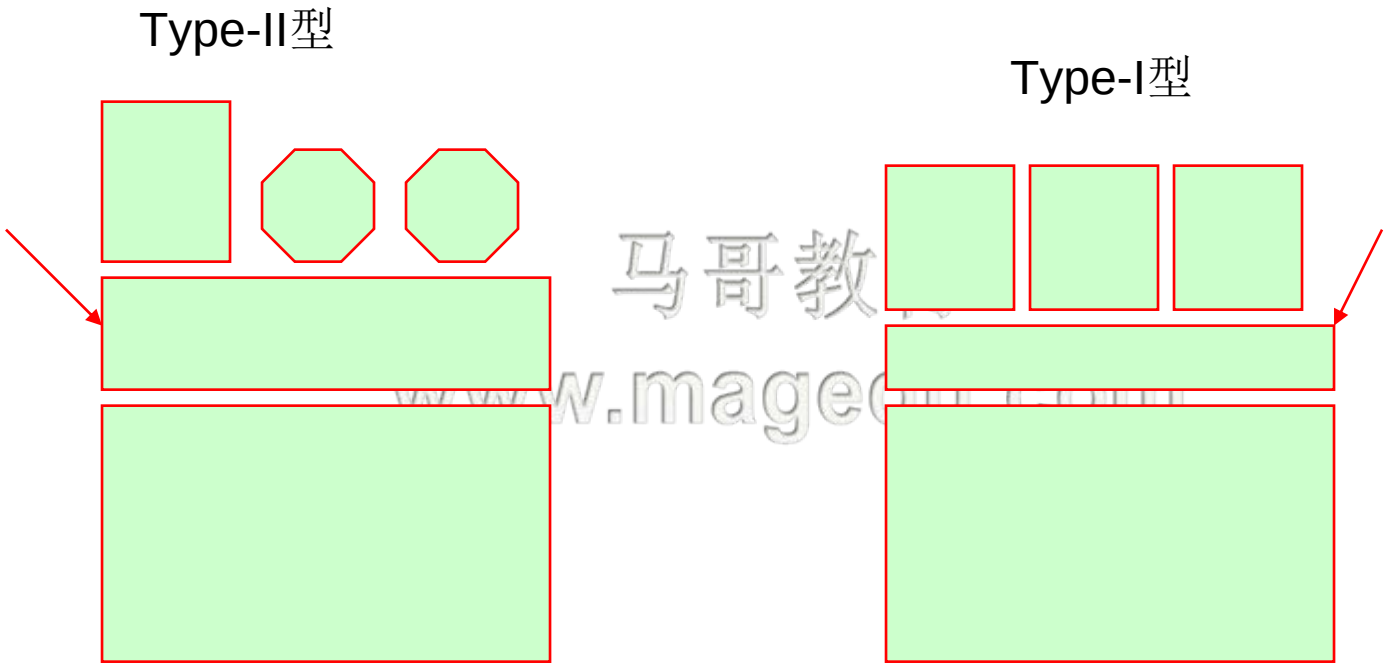
GVA: guest virtual address
GPA: guest physical address

- Intel和AMD分别通过EPT(Extended Page Tables)和NPT(Nested Page Tables)为虚拟化应用提升影子MMU的性能，并通过标记(tagged)TLB来避免虚拟机切换时频繁清写(flush)TLB以提高TLB缓存的命中率。


Virtualized I/O
Passthrough I/O
Hosted or Split
Hypervisor Direct
Guest OS
Guest Device Driver
Virtual Device (VMM)
Host OS/Dom0/ Parent Domain
Virtual Device
I/O Stack
Physical Device Driver
Guest OS
Guest Device Driver
Virtual Device
I/O Stack
Physical Device Driver
Guest OS
Guest Device Driver
Device Manager

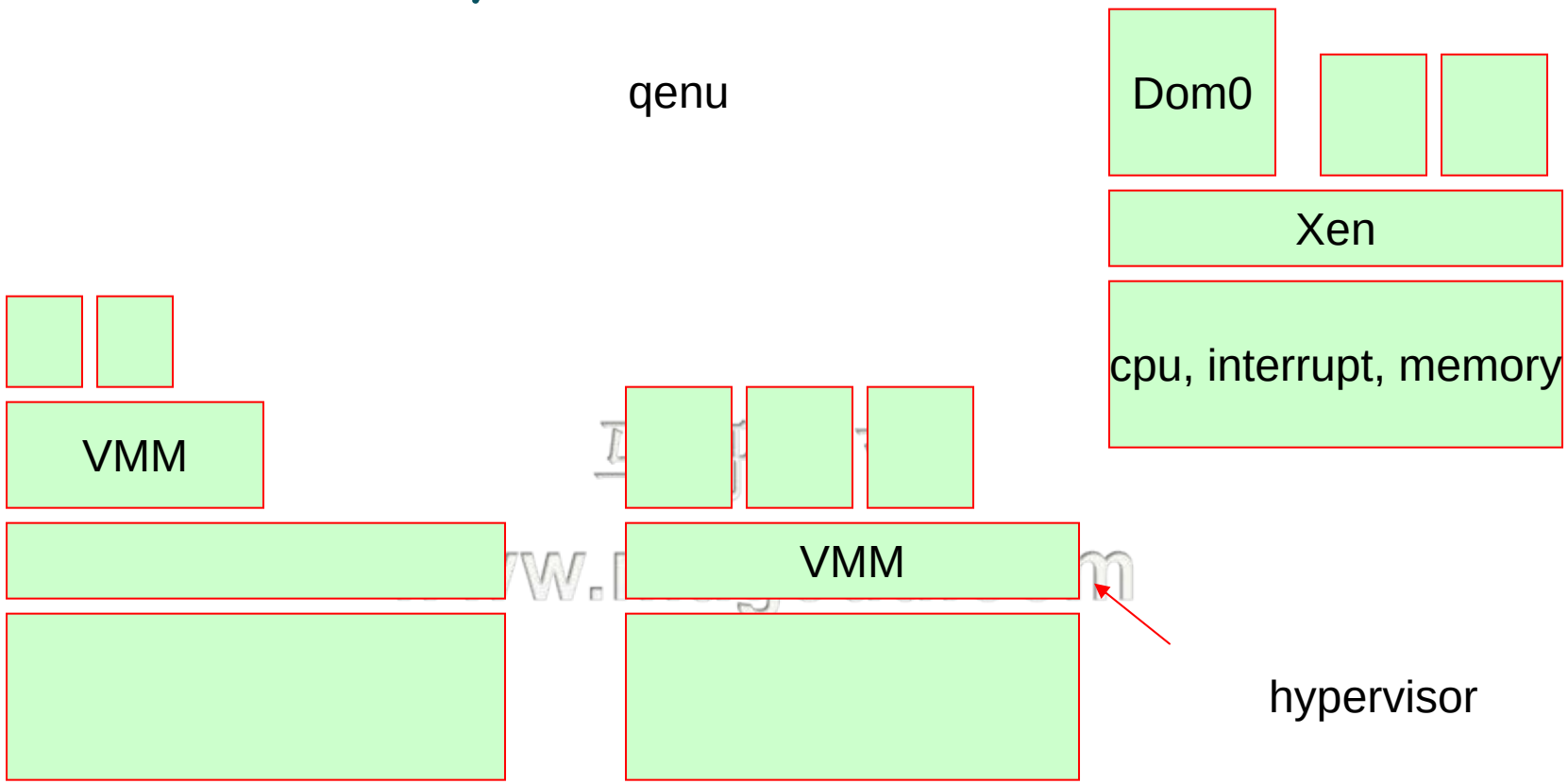


❖ Type-I型

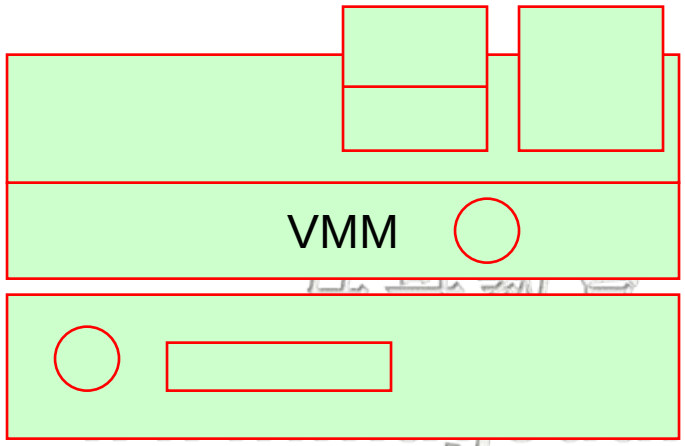


Hosted VMM

- 虚拟化模型
  - Vmware workstation, VMWare Server, VirtualBox
  - VMWare ESX, VMWare ESXi



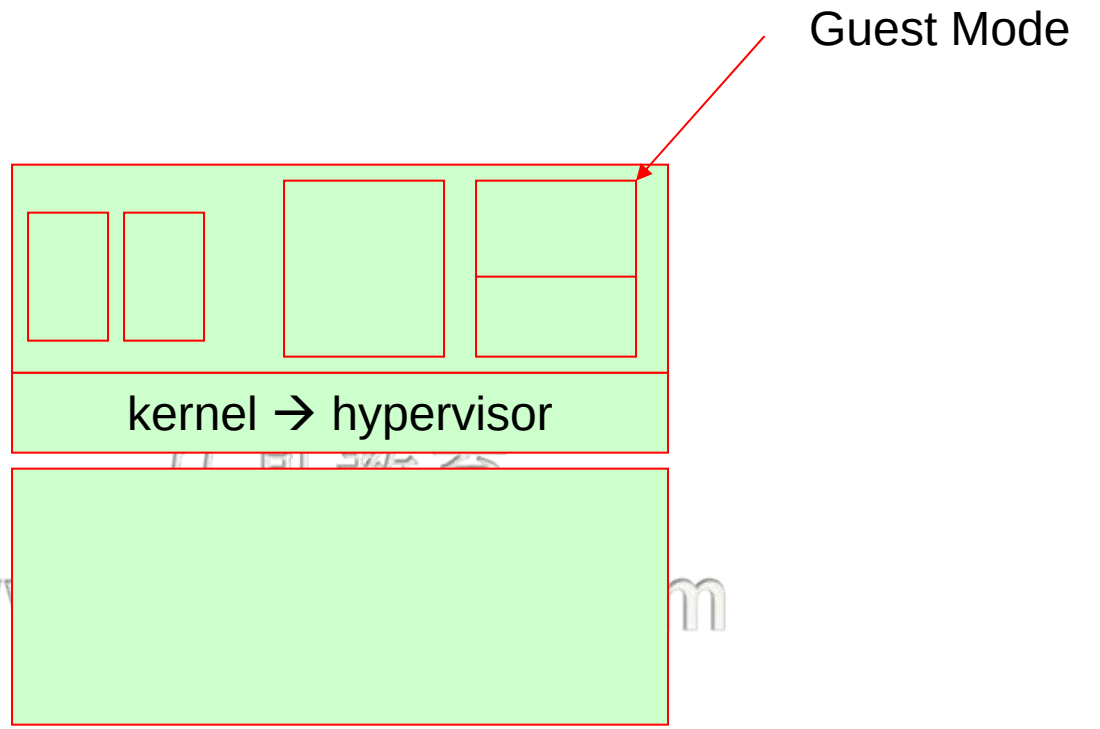
- kqemu，kvm (Kernel-based VM)



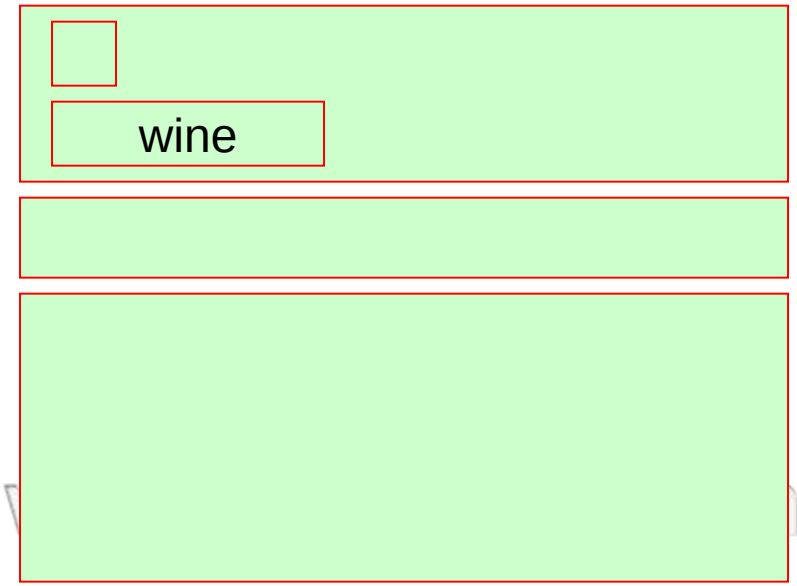
KVM: Kernel-based Virtual Machine

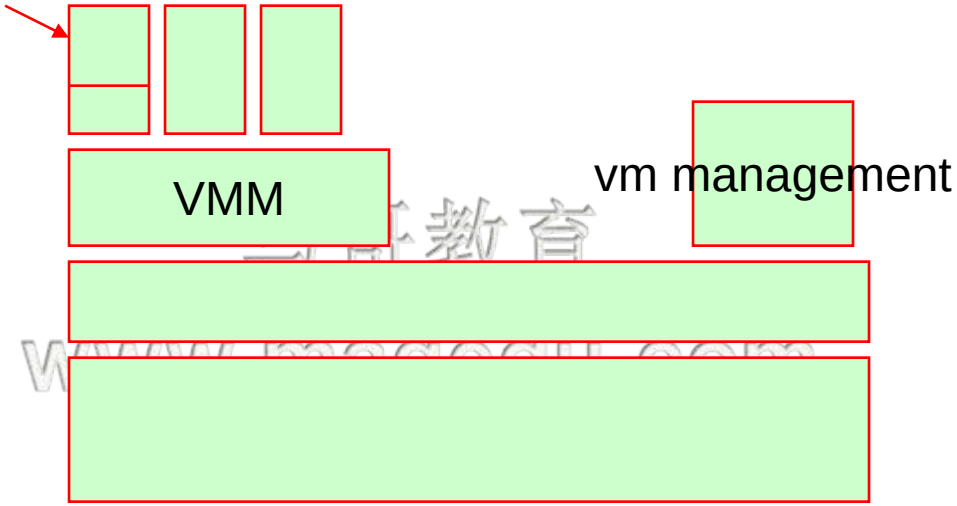
基于内核的虚拟机

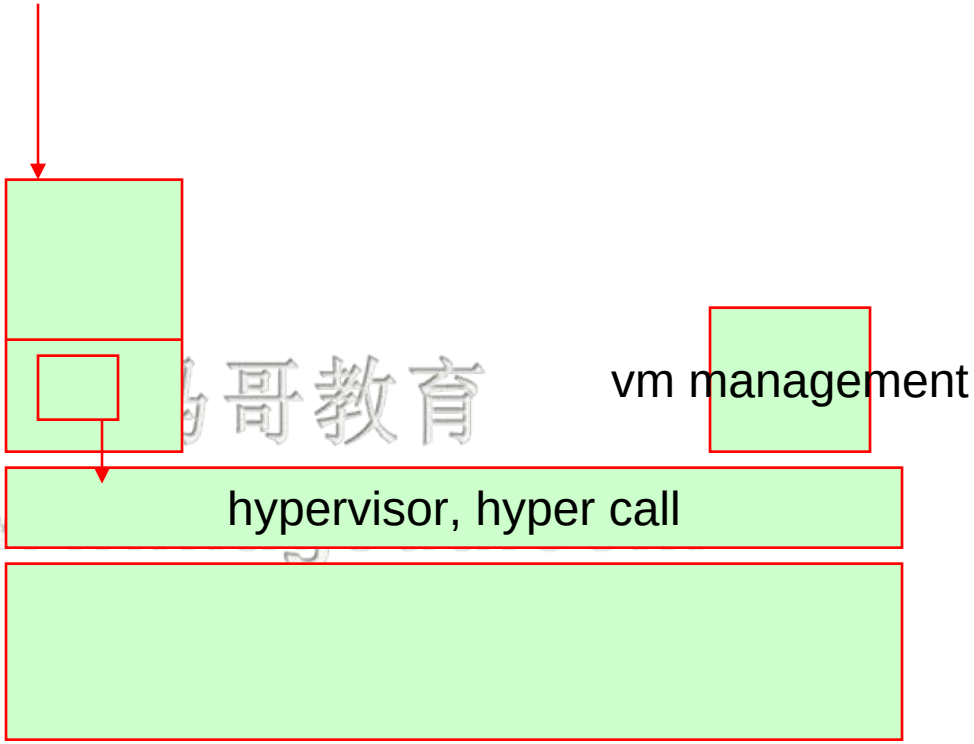
KVM: 内核模块

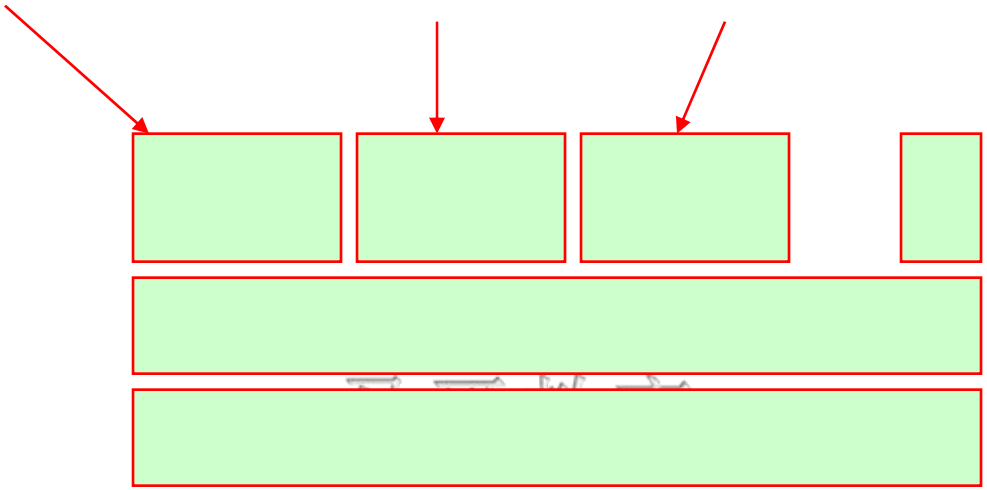


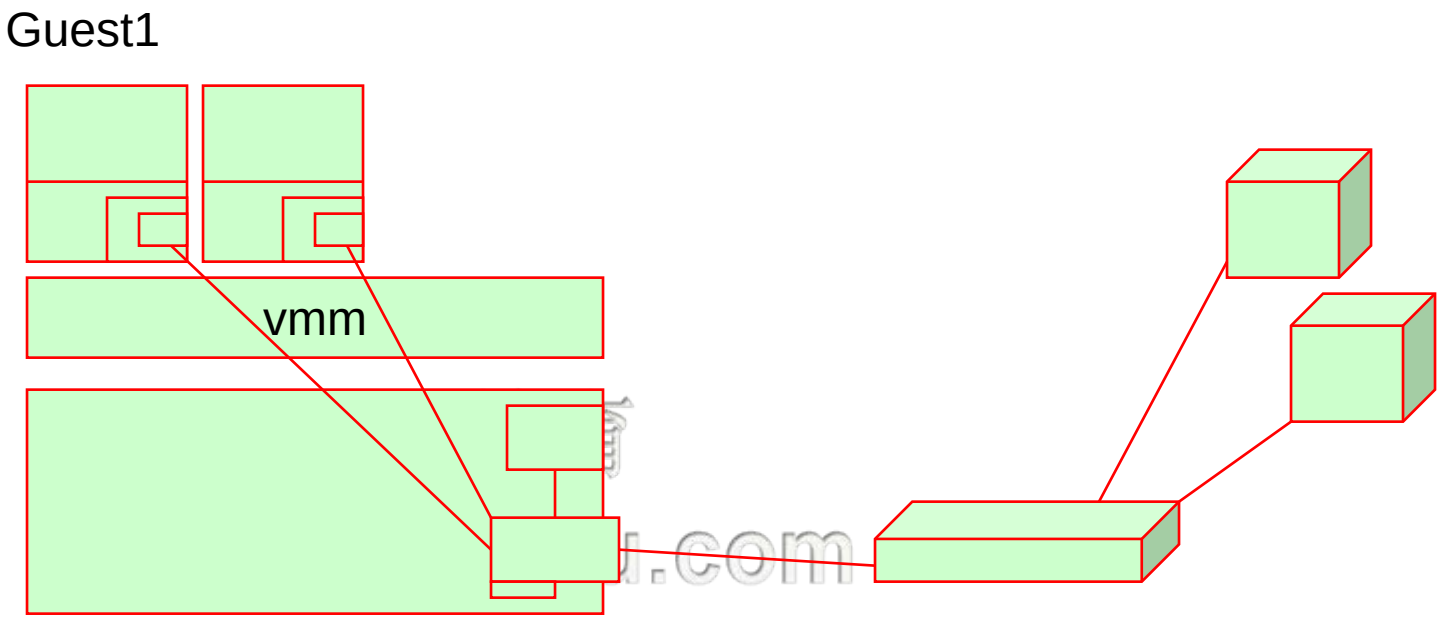
- Container:
  - OpenVZ

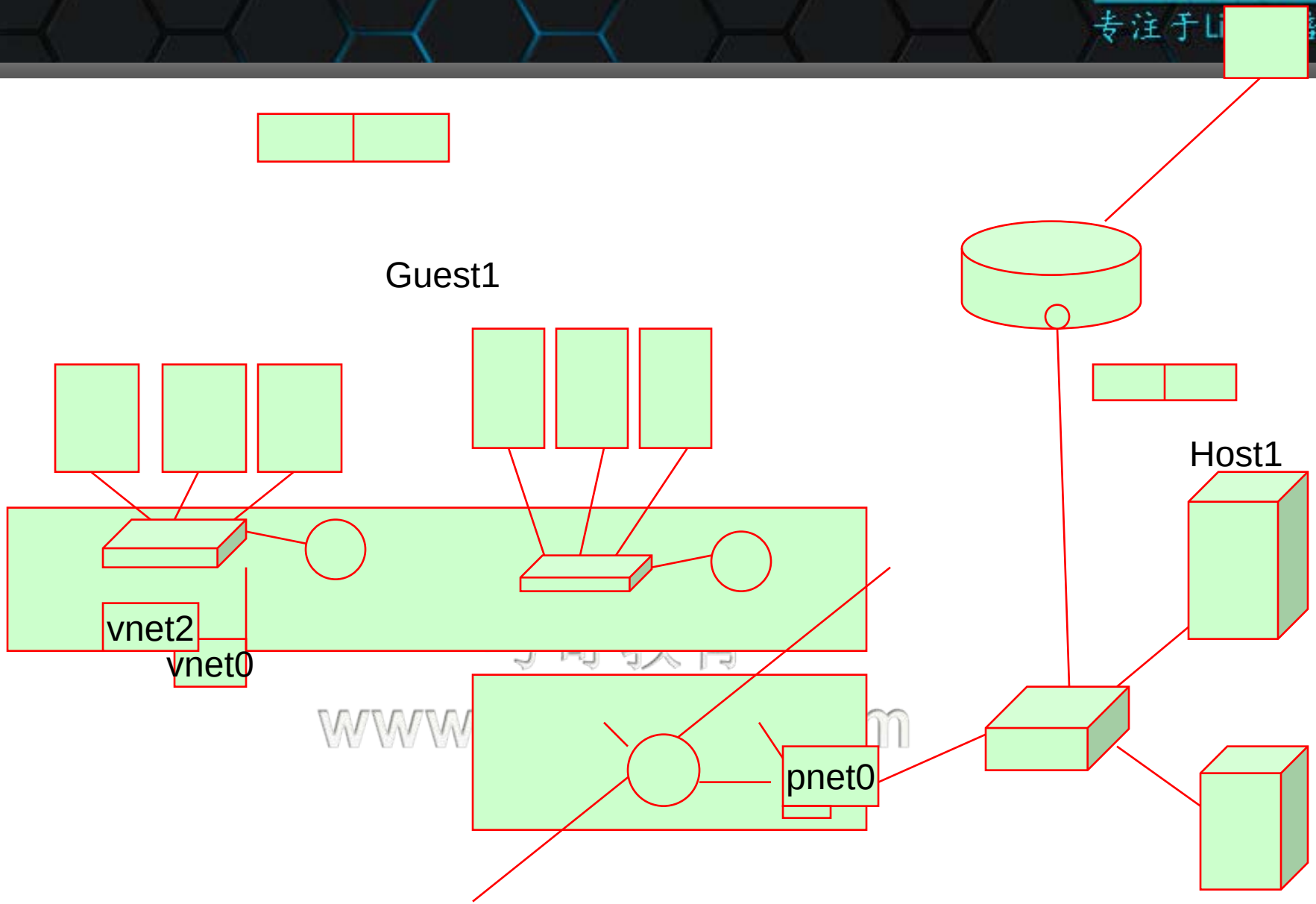


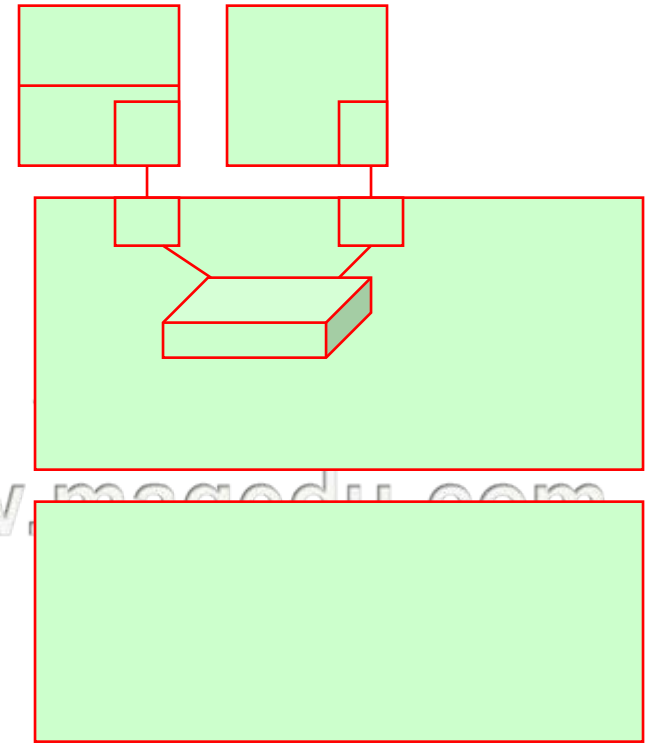

VMM
vm management

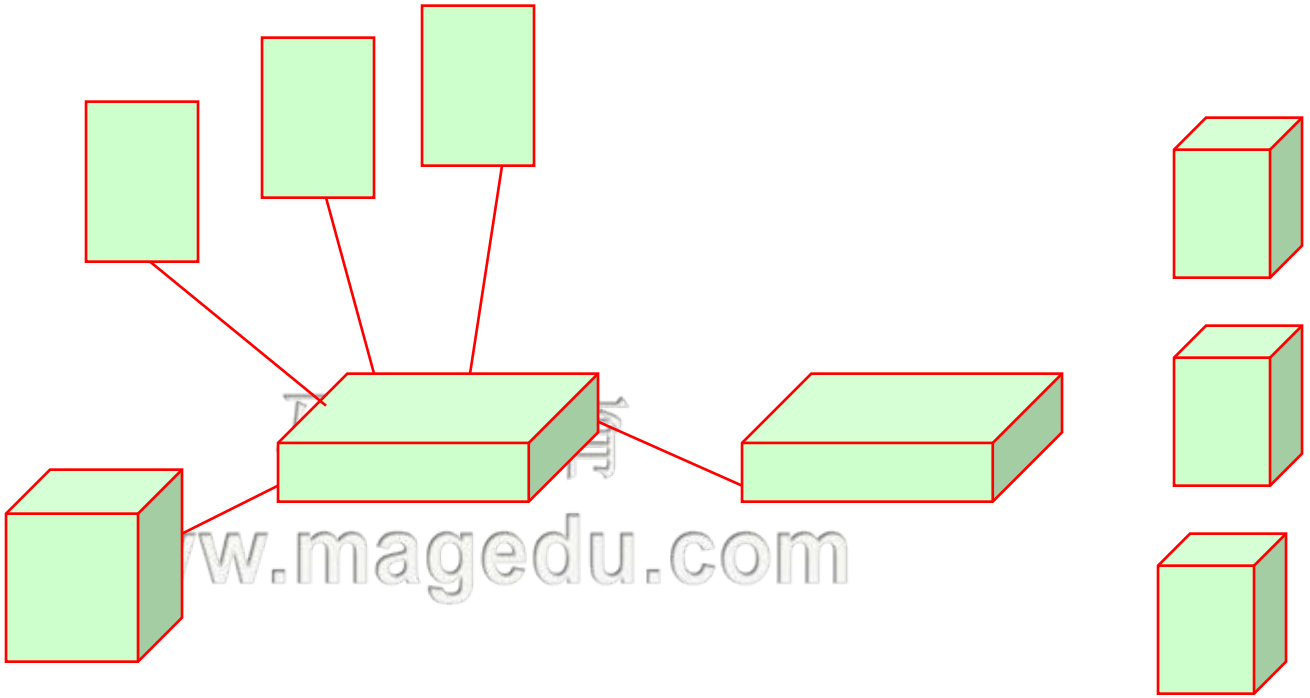

vm management
hypervisor, hyper call



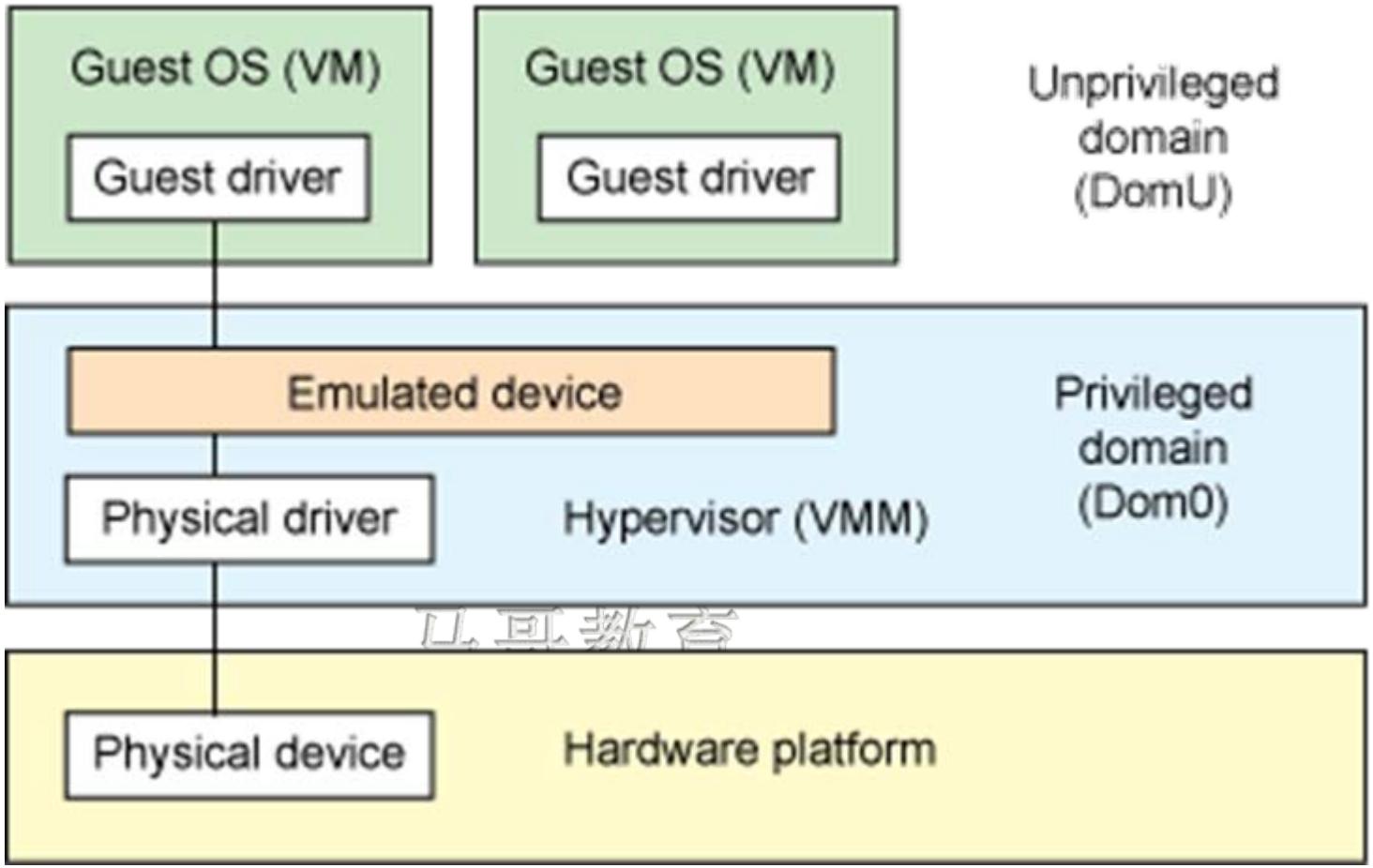

Guest1
vmm


Guest1
Host1
vnet2
vnet0
pnet0

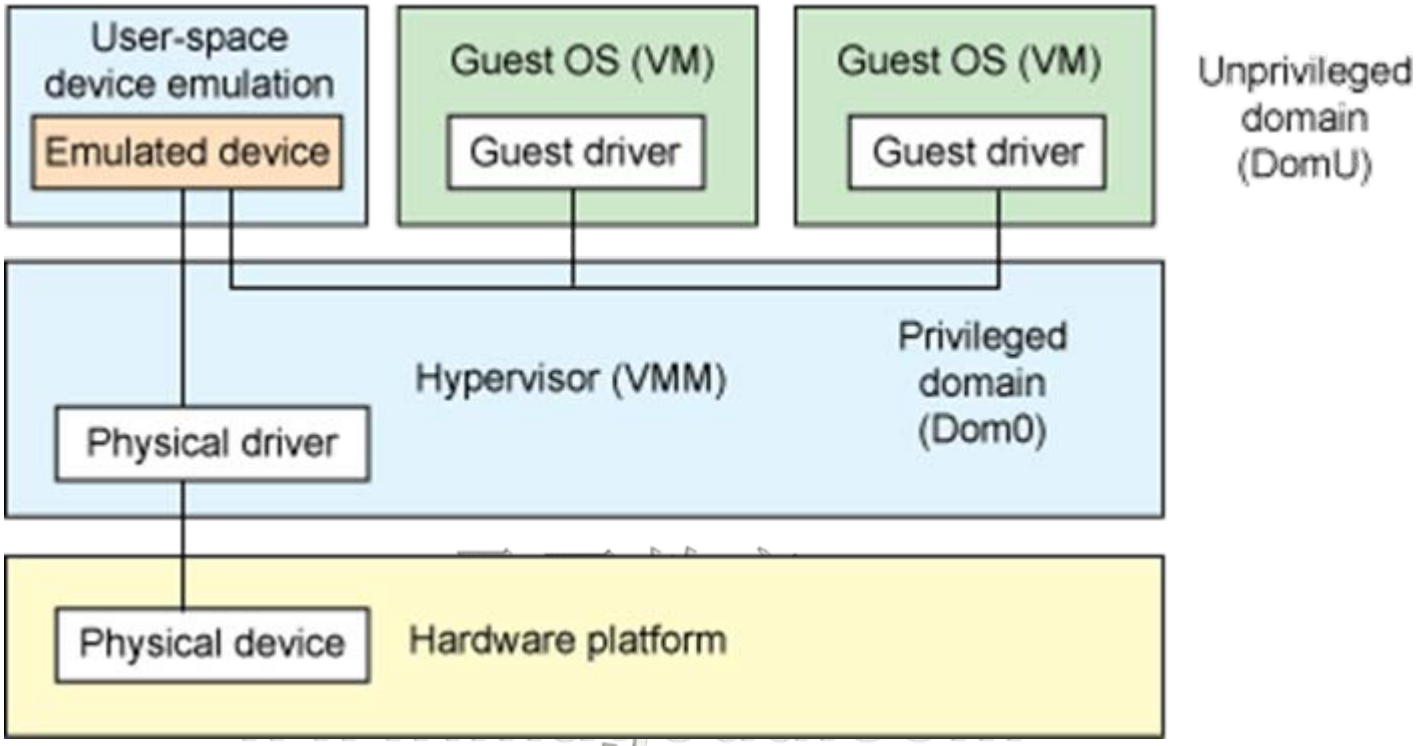




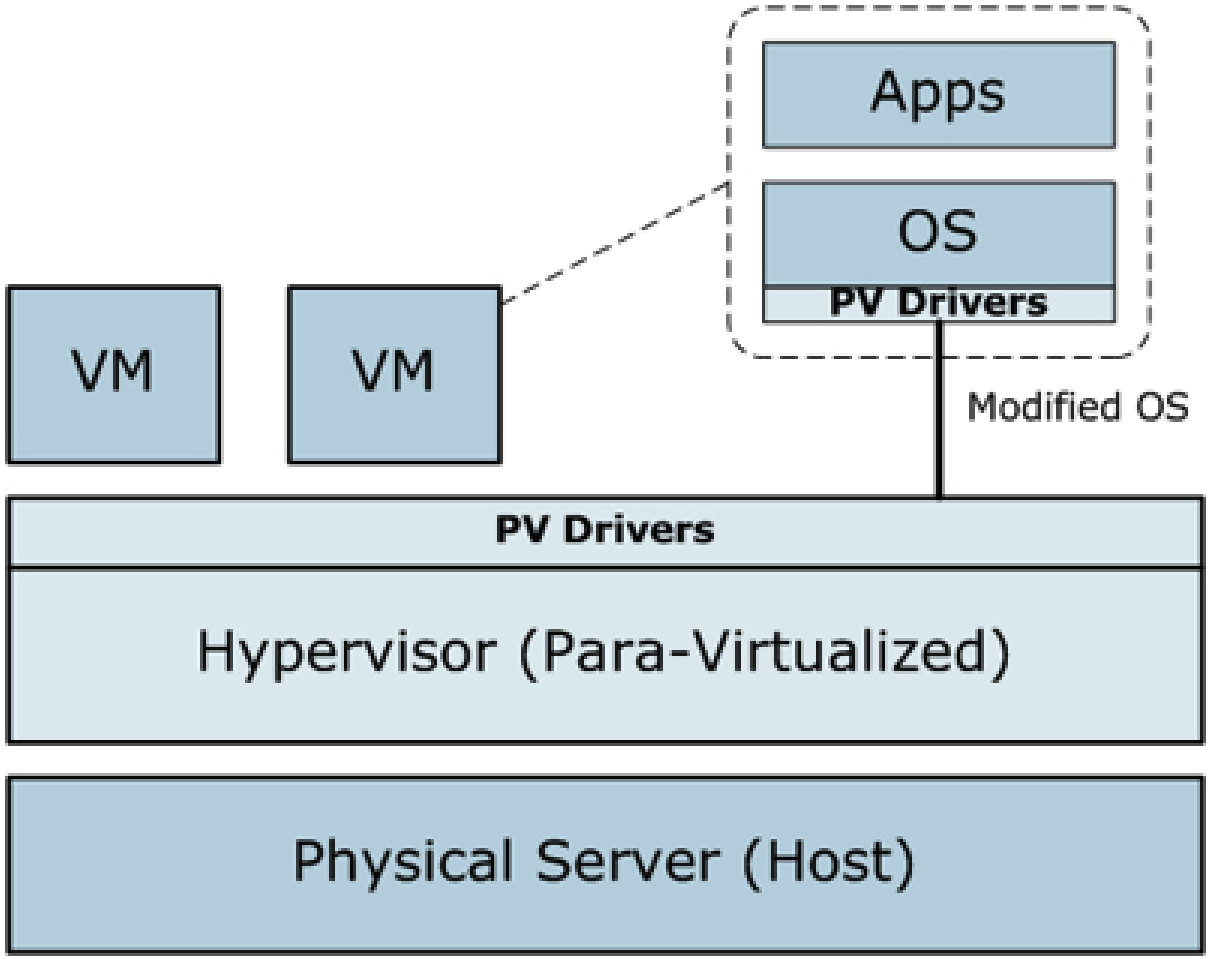
模拟


Guest OS (VM)
Guest driver
Guest OS (VM)
Guest driver
Unprivileged domain (DomU)
Emulated device
Physical driver
Hypervisor (VMM)
Privileged domain (Dom0)
Physical device
Hardware platform

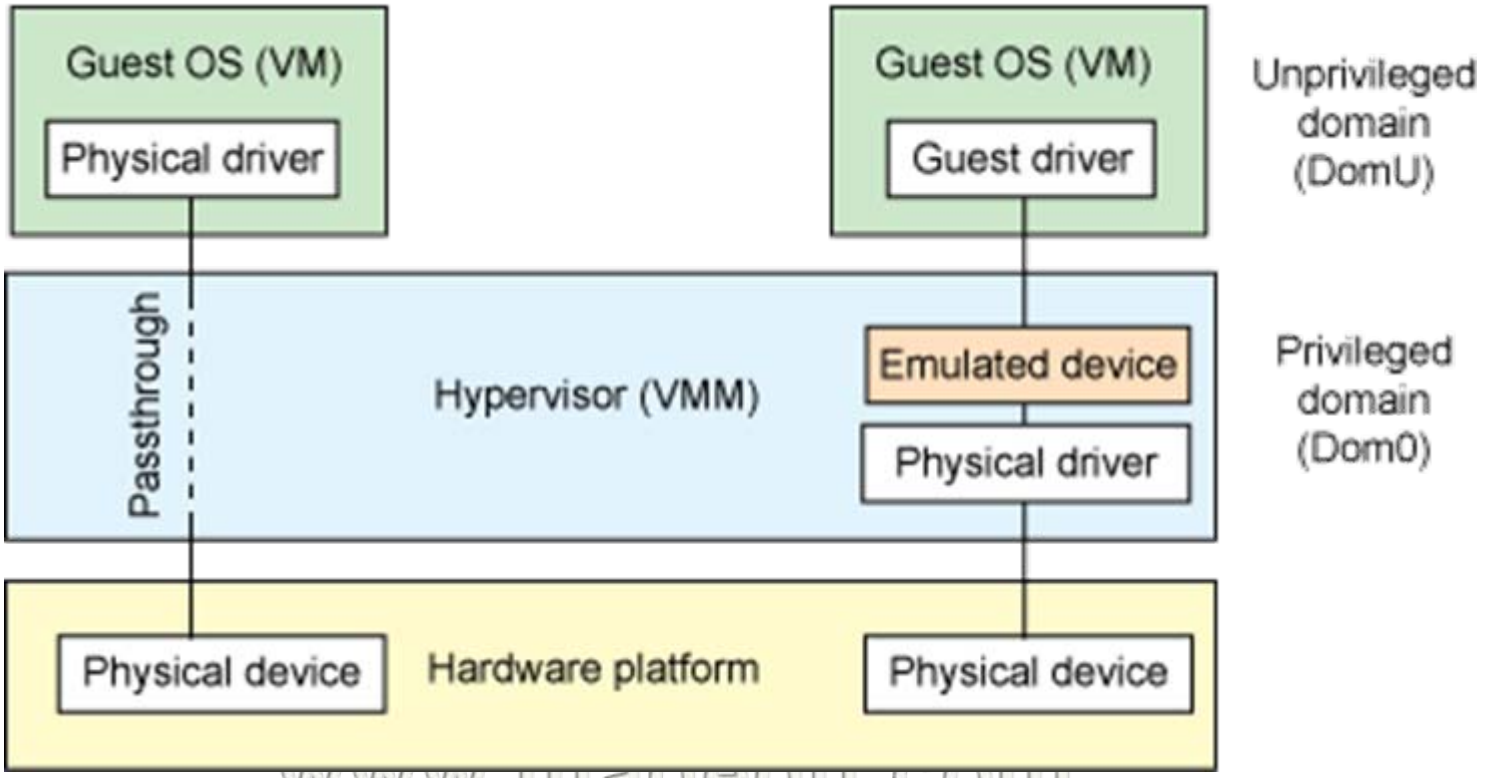
基于用户空间的设备模拟



半虚拟化


Apps
OS
PV Drivers
VM
VM
Modified OS
PV Drivers
Hypervisor (Para-Virtualized)
Physical Server (Host)

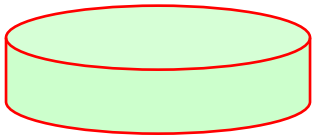
IO透传



马哥教育

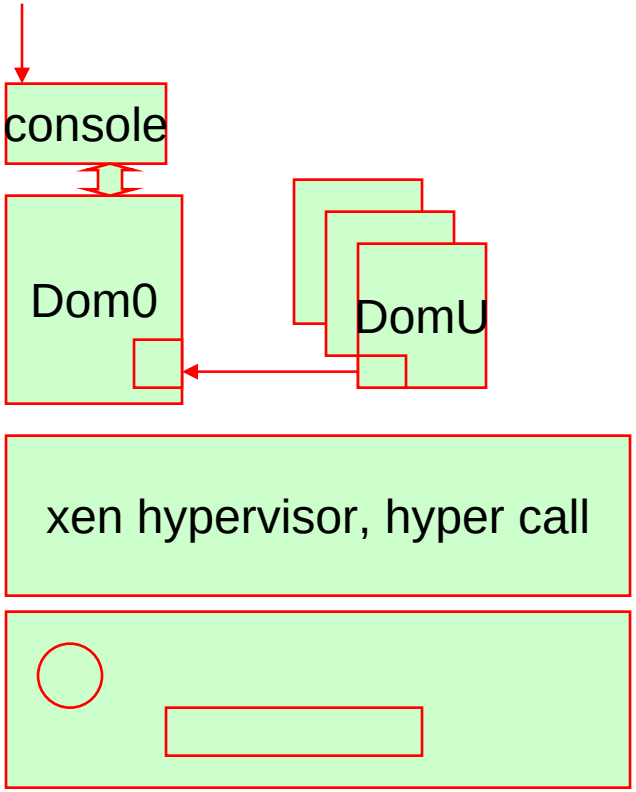
Xen

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com



- dom0的内核
  - DomU，initramfs
  - init，→ ldd
  - bash → ldd

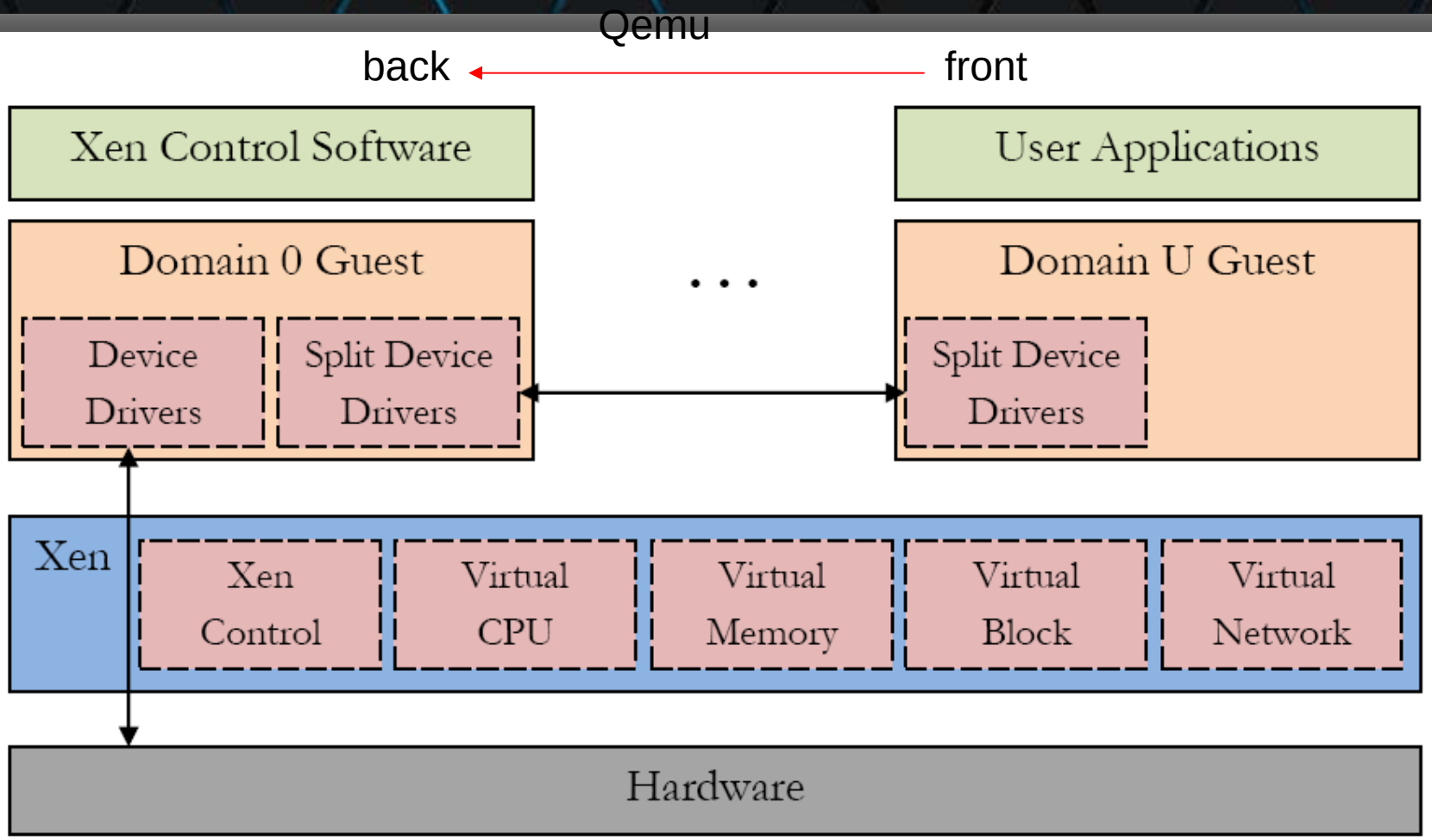
xm



Xen被收录进Linux内核的步骤

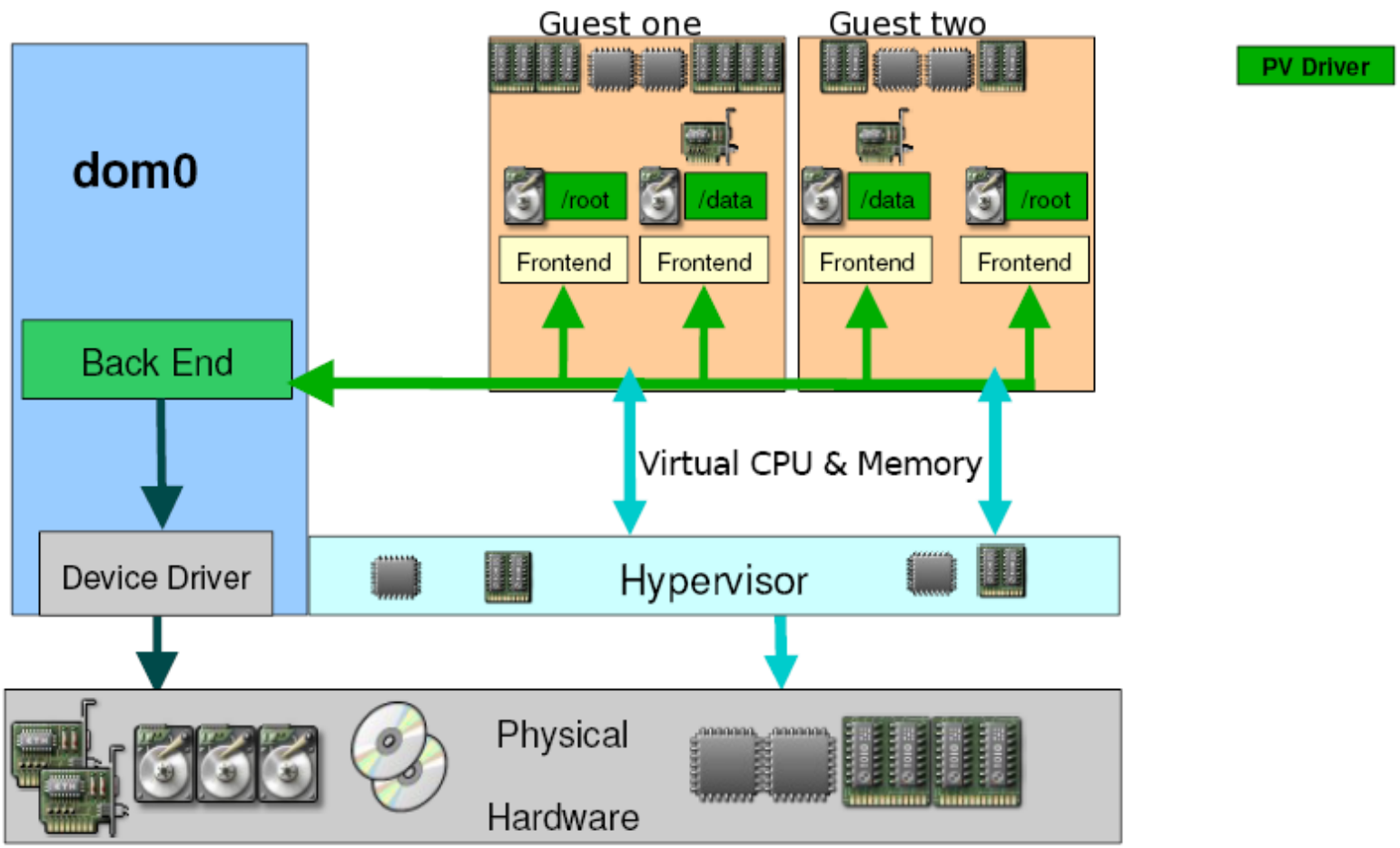
- Linux pvops framework (DomU)
  - Linux-2.6.24
- Dom0
  - Linux-2.6.37
- 关键驱动及优化
  - Linux-3.0.0

Xen Arch


Qemu
back
front
Xen Control Software
User Applications
Domain 0 Guest
Domain U Guest
Device Drivers
Split Device Drivers
Split Device Drivers
Xen
Xen Control
Virtual CPU
Virtual Memory
Virtual Block
Virtual Network
Hardware

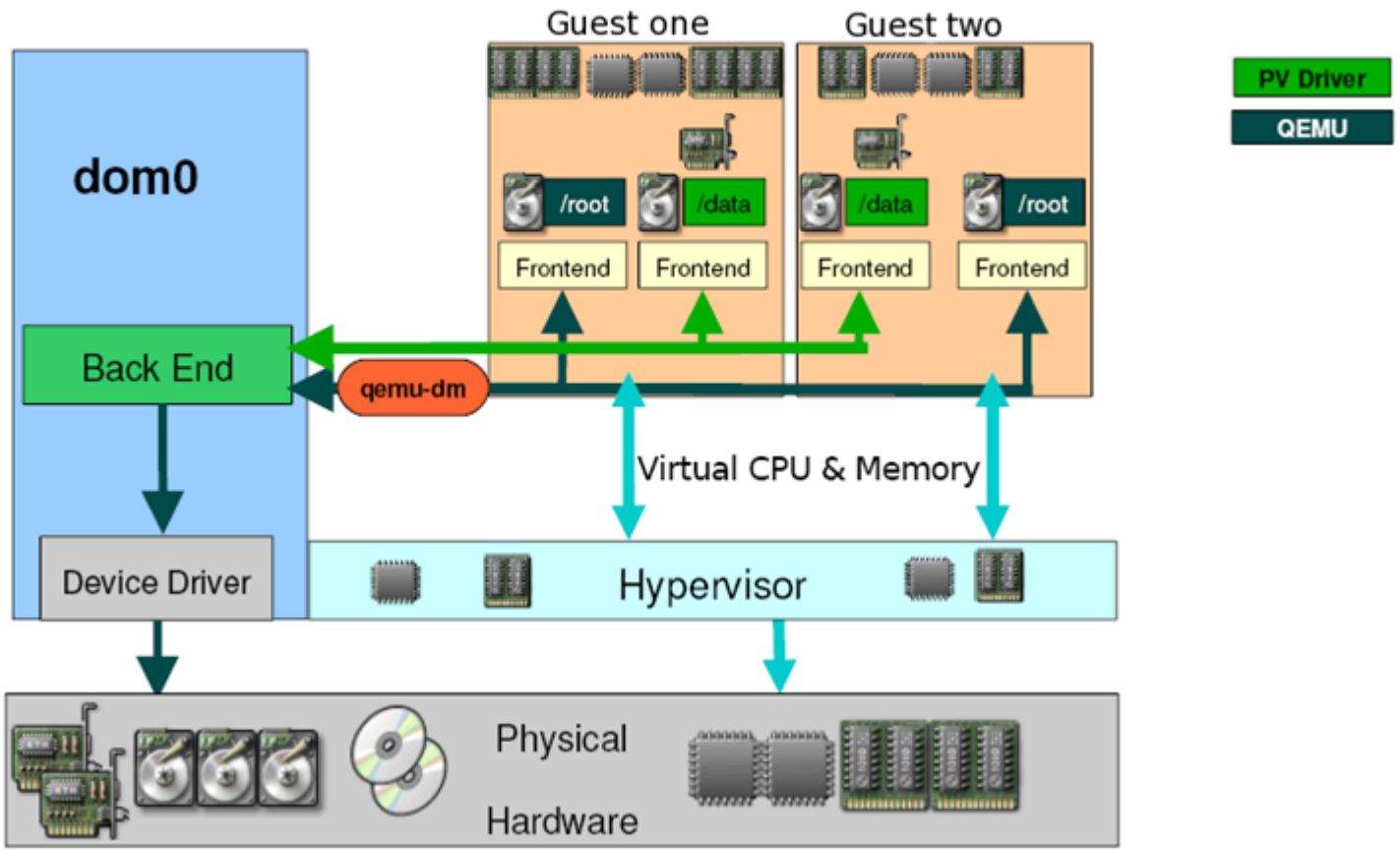
- Xcp
- Linux → OS
  - Xen
  - /etc/grub.conf
    - Kernel /xen.gz
    - Module /vmlinuz
    - Mouule /initrd

Xen Para-virtualization Architecture

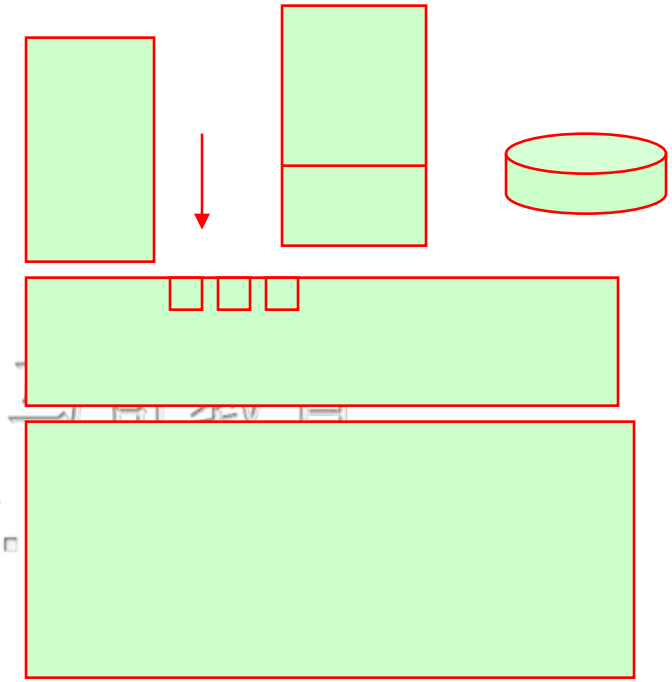


Xen Full Virtualization Architecture

With the para-virtualized drivers



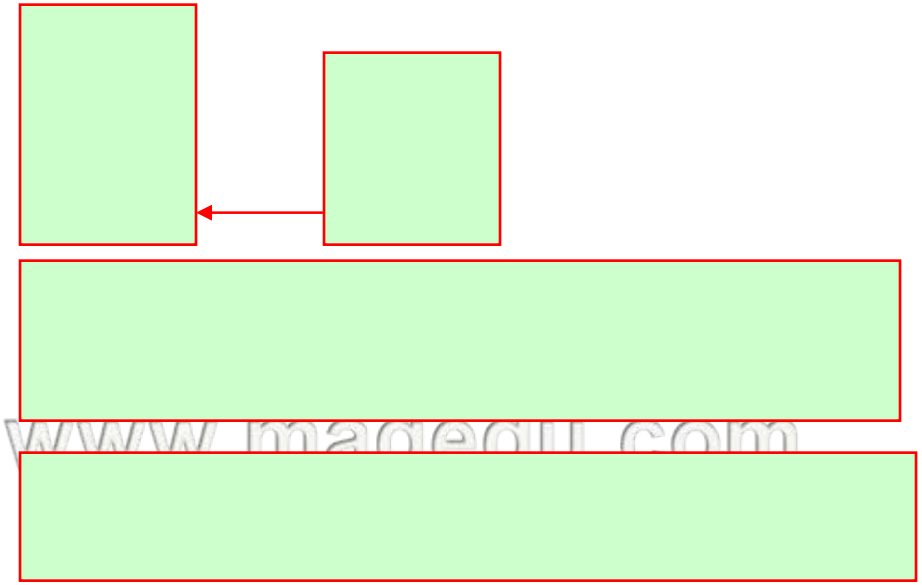
❖ 配置文件



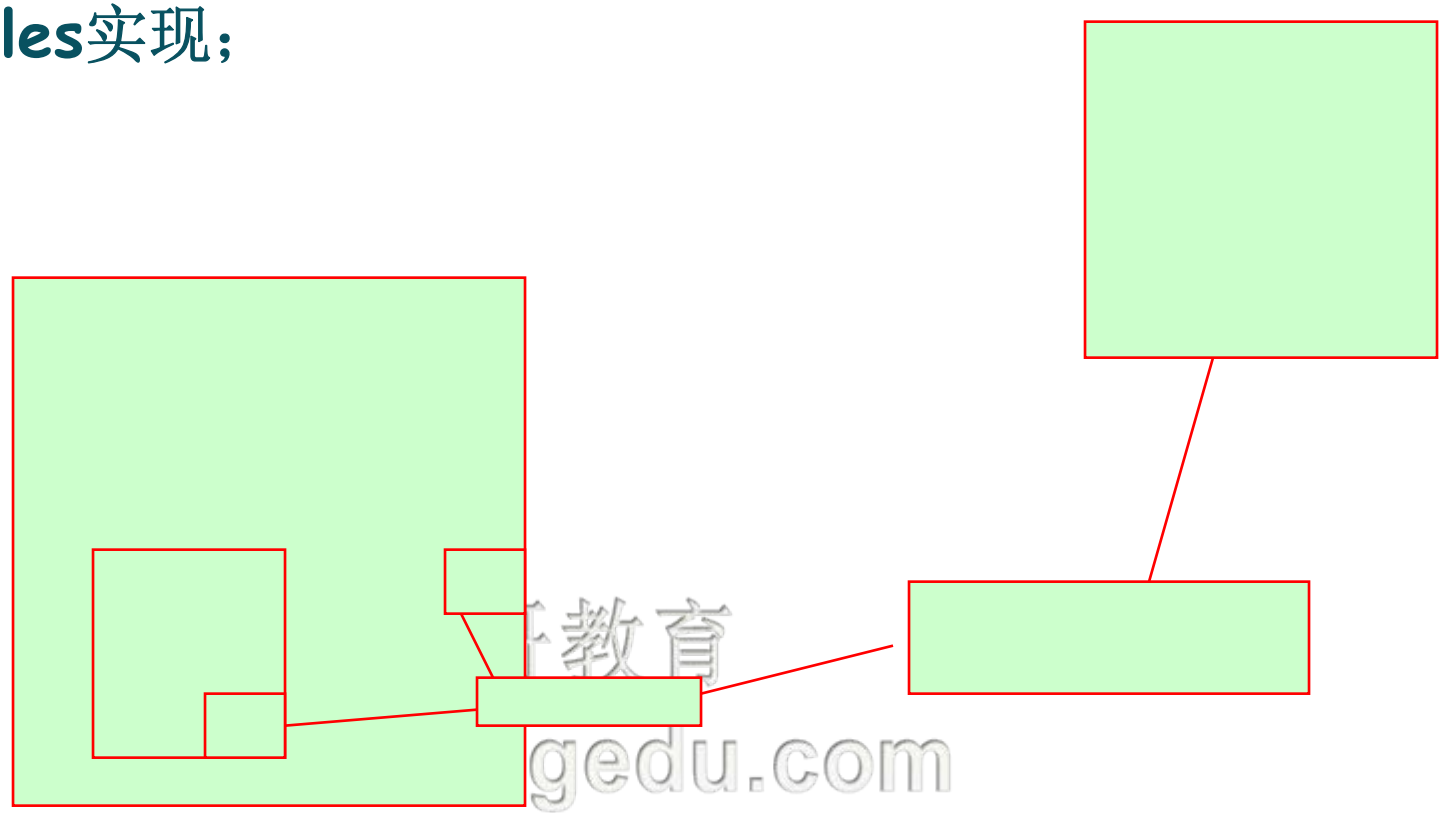
实时迁移

- 实时迁移的需求：
  - GuestOS映像文件放轩在共享存储上，如iSCSI、NFS或GFS2等；
  - 目标物理主机的Xen要能够兼容源物理主机的Xen及其准备迁移的GuestOS；
  - 在两台主机上，共享存储必须挂载在同一位置，且挂载的目录名必须一致；
  - 两台物理主机的CPU需要具有相同类型的特性；
  - 虚拟机没使用透传I/O；
  - 两台物理主机的时间要同步；
  - 两台物理主机必须有一致的网络配置，且所有桥接和网络配置必须完全一致；

- grub.conf
- kernel /xen.gz
- module /vmlinuz
- module /initramfs



- Linux上各种虚拟化解决方案在实现网络功能，大多数都是基于iptables实现；



Dom0 Kernel

- CONFIG_ACPI_PROCFS=y
- CONFIG_XEN=y
- CONFIG_XEN_MAX_DOMAIN_MEMORY=32
- CONFIG_XEN_SAVE_RESTORE=y
- CONFIG_XEN_DOM0=y
- CONFIG_XEN_PRIVILEGED_GUEST=y
- CONFIG_XEN_PCI=y
- CONFIG_PCI_XEN=y
- CONFIG_XEN_BLKDEV_FRONTEND=y
- CONFIG_XEN_NETDEV_FRONTEND=y
- CONFIG_XEN_KBDDEV_FRONTEND=y
- CONFIG_HVC_XEN=y
- CONFIG_XEN_FBDEV_FRONTEND=y
- CONFIG_XEN_BALLOON=y

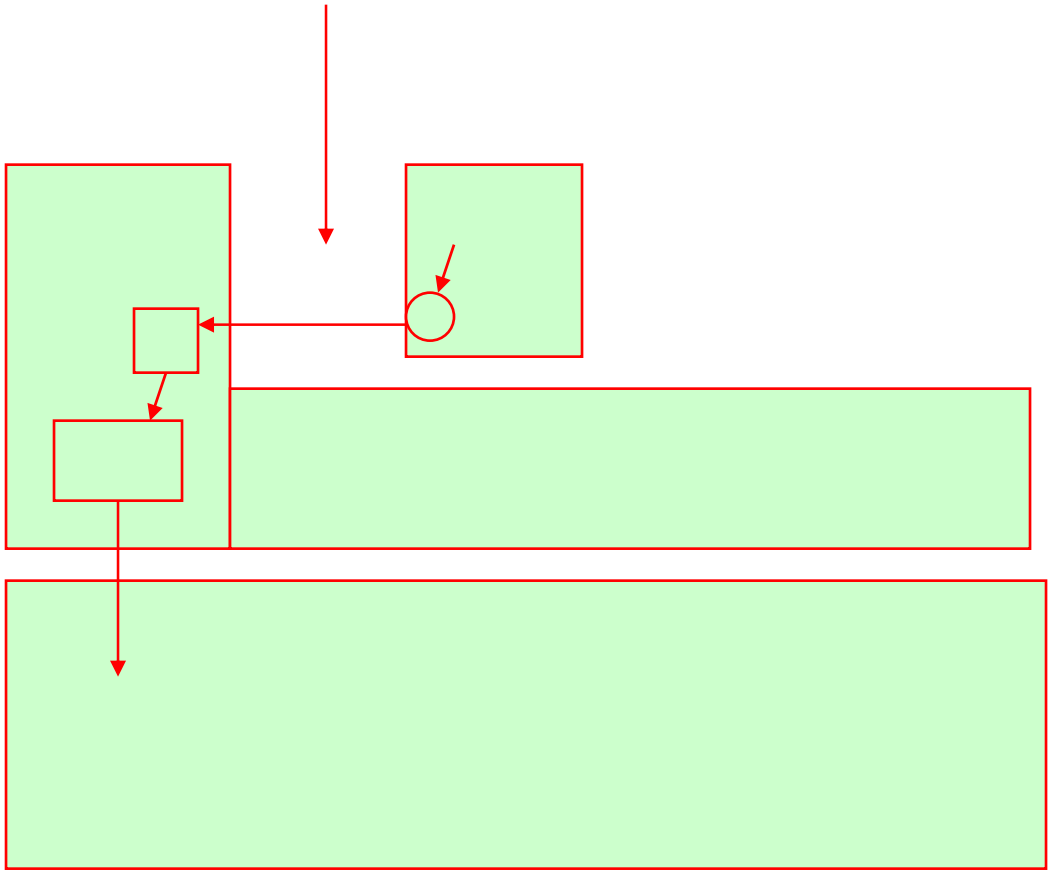
Dom0 Kernel (2)

- CONFIG_XEN_SCRUB_PAGES=y
- CONFIG_XEN_DEV_EVTCHN=y
- CONFIG_XEN_GNTDEV=y
- CONFIG_XEN_BACKEND=y
- CONFIG_XEN_BLKDEV_BACKEND=y
- CONFIG_XEN_NETDEV_BACKEND=y
- CONFIG_XENFS=y
- CONFIG_XEN_COMPAT_XENFS=y
- CONFIG_XEN_XENBUS_FRONTEND=y
- CONFIG_XEN_PCIDEV_FRONTEND=y

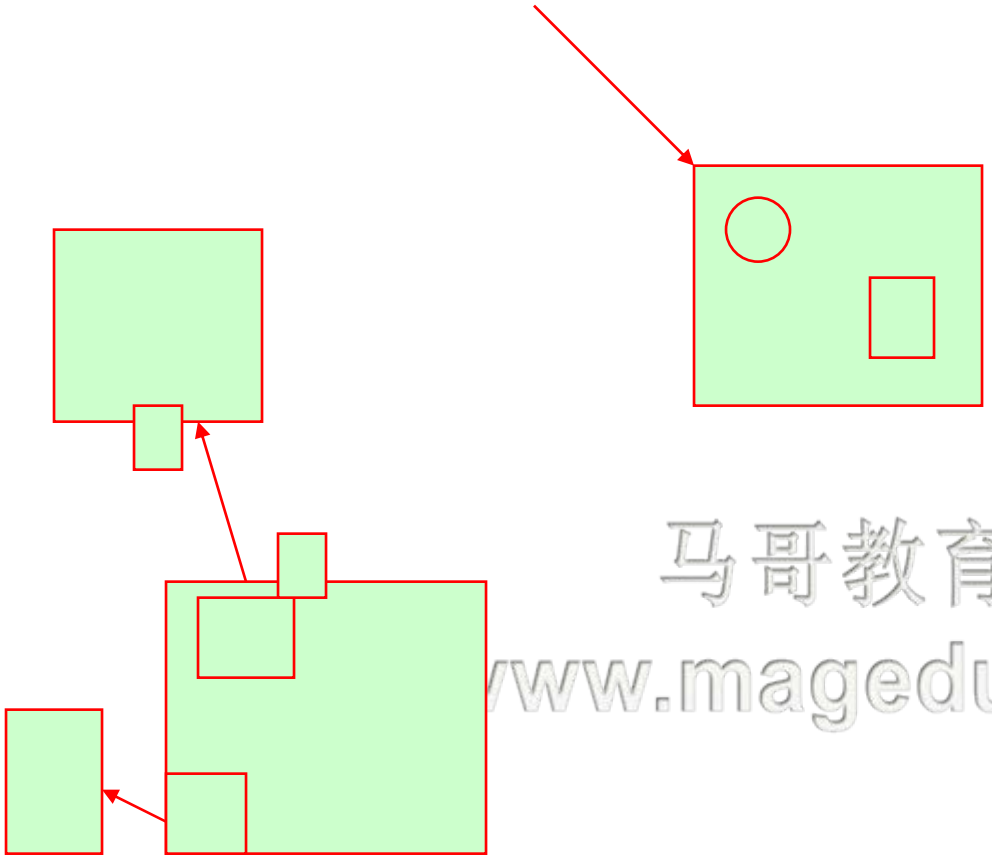
DomU Kernel

- CONFIG_XEN=y
- CONFIG_PARAVIRT_GUEST=y
- CONFIG_PARAVIRT=y
- CONFIG_XEN_PVHVM=y
- CONFIG_XEN_MAX_DOMAIN_MEMORY=128
- CONFIG_XEN_SAVE_RESTORE=y
- CONFIG_PCI_XEN=y
- CONFIG_XEN_PCIDEV_FRONTEND=y
- CONFIG_XEN_BLKDEV_FRONTEND=y
- CONFIG_XEN_NETDEV_FRONTEND=y
- CONFIG_INPUT_XEN_KBDDEV_FRONTEND=y
- CONFIG_HVC_XEN=y
- CONFIG_XEN_FBDEV_FRONTEND=y
- CONFIG_XEN_DEV_EVTCHN=y
- CONFIG_XEN_XENBUS_FRONTEND=y

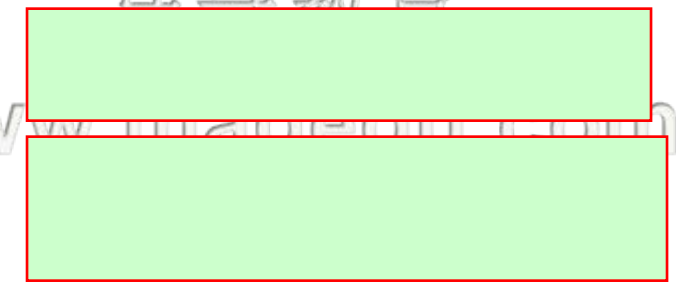
- xenfront, xvd[a-z]



- c/s



libvirt



Xen Guest File

- **kernel:** Set this to the path of the domU kernel you compiled for use with Xen.
- **ramdisk:** Set this to the path of the initramfs you created for the above domU kernel.
- **memory:** Set this to the size of domain's memory in megabytes.
- **name:** Give a unique name to the new domain.
- **vif:** Define network interfaces. The syntax of this option is:

  vif = ['mac=MAC_ADDRESS', 'bridge=BRIDGE']

  where MAC_ADDRESS is a valid MAC address for the new virtual interface and BRIDGE is a valid BRIDGE on dom0. You can also specify only the bridge:

  vif = [", 'bridge=BRIDGE']

  or let the system to choose the defaults:

  vif = ["]

- **disk:** Define the disk devices you want the domain to have access to and what you want them accessible as. Each disk entry is of the form phy:UNAME,DEV,MODE where UNAME is the device, DEV is the device name the domain will see, and MODE is 'r' for read-only, 'w' for read-write.

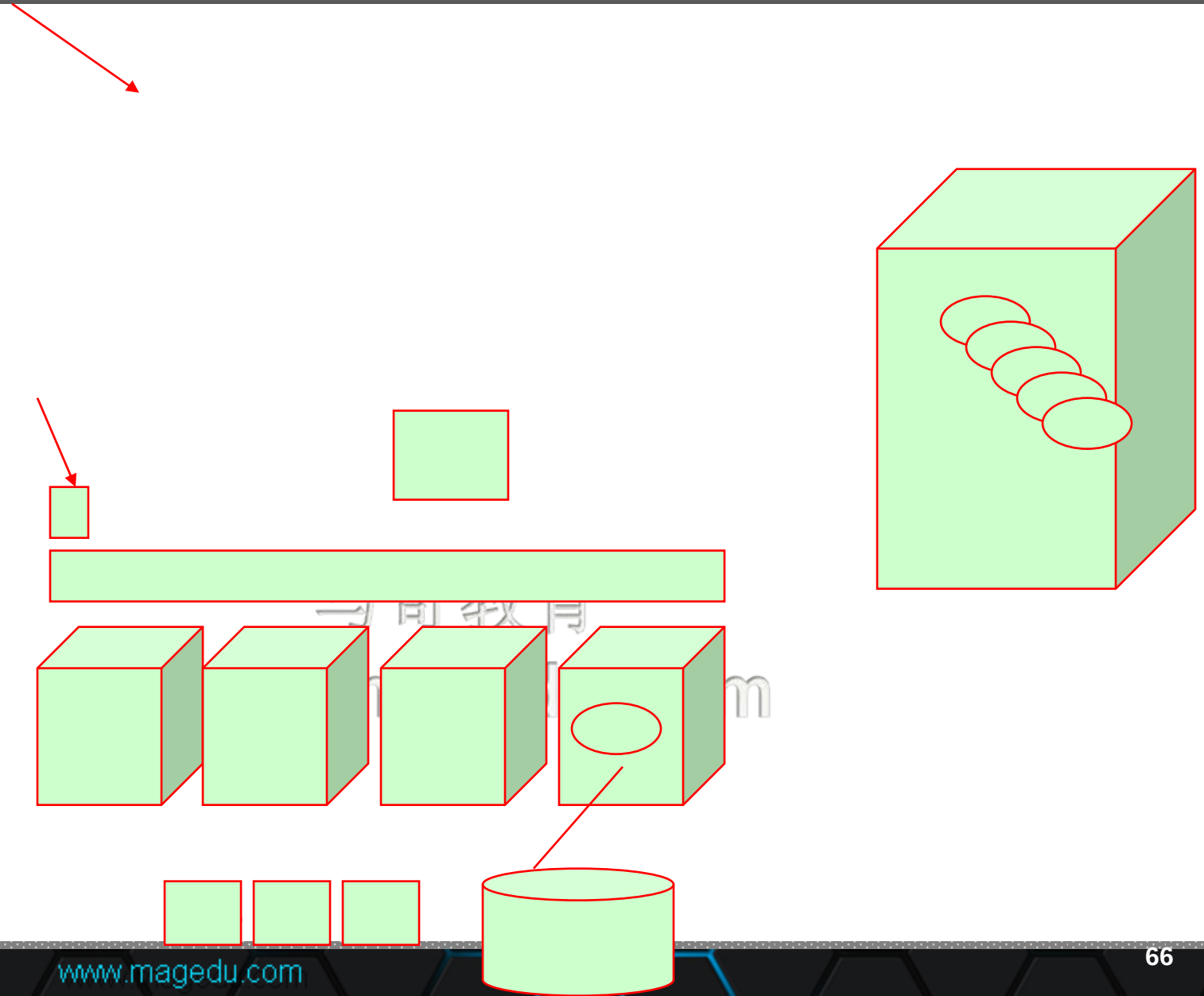
- **dhcp:** Uncomment the dhcp variable, so that the domain will receive its IP address from a DHCP server.
- **root:** Set the root device. The guest will use this device to mount its root file system.
- **extra:** Set extra kernel parameters (e.g. runlevels).

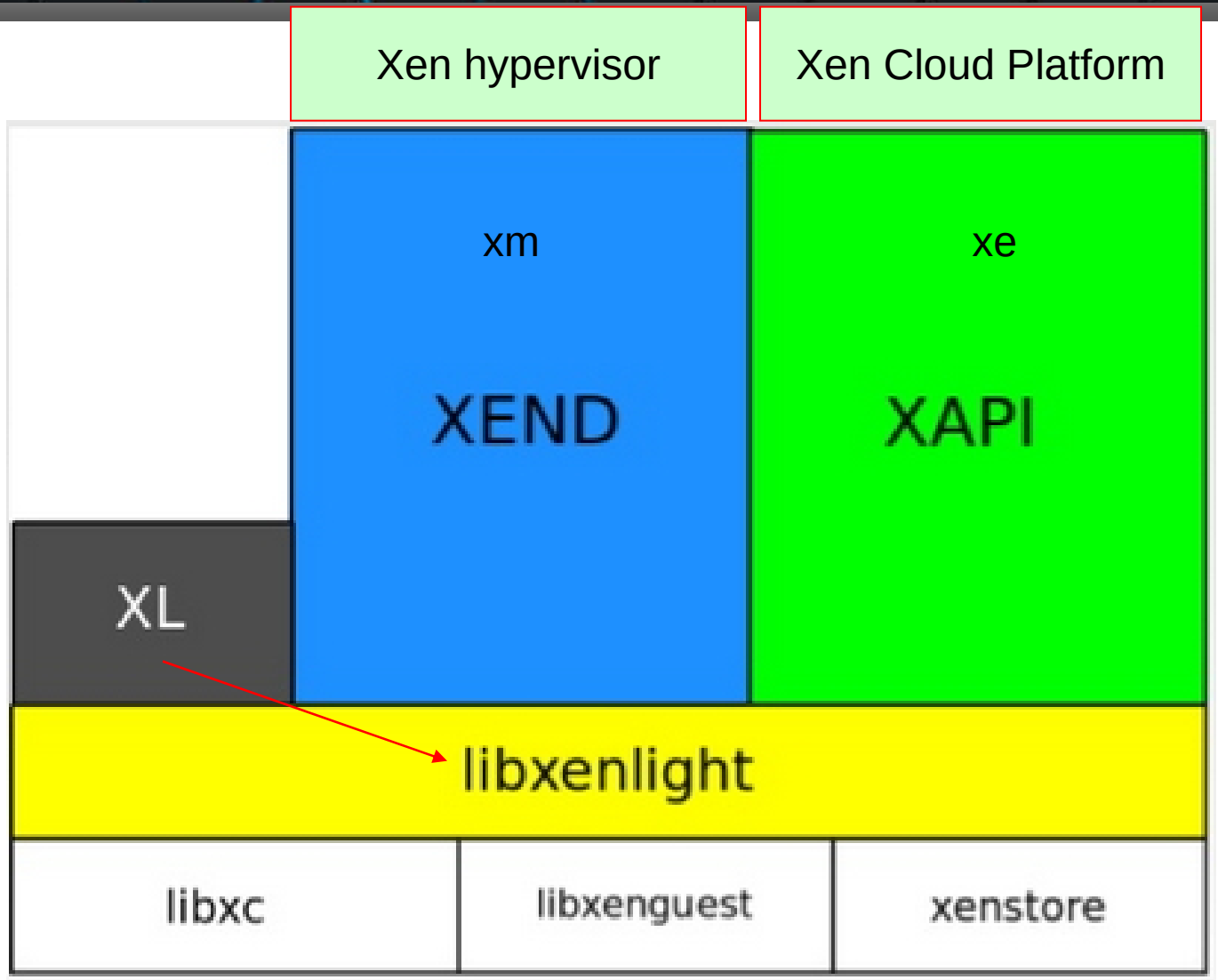
- Bridge Networking:
  (network-script network-bridge)
  (vif-script vif-bridge)

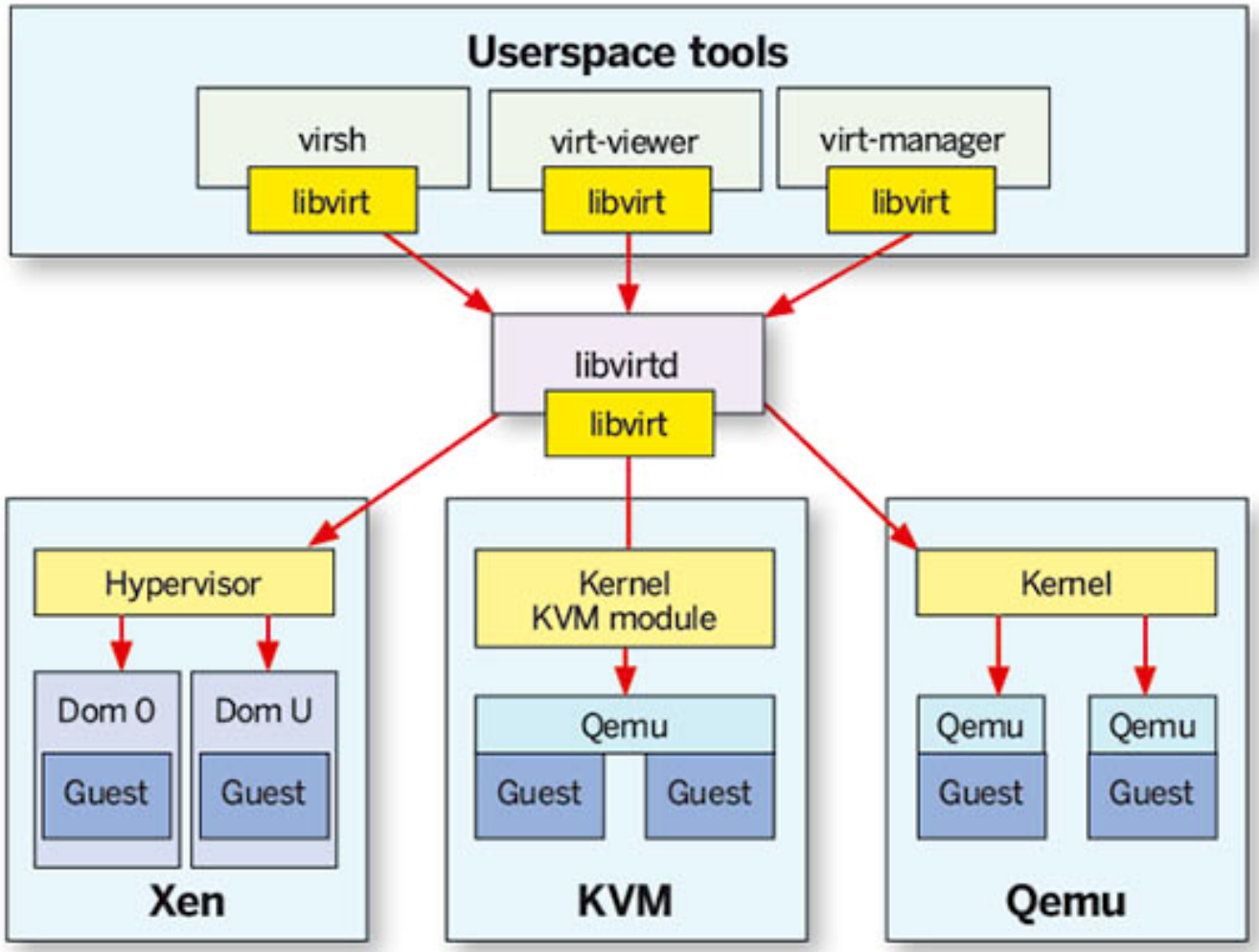
- Routed Networking with NAT:
  (network-script network-nat)
  (vif-script vif-nat)

- Two-way Routed Networking:
  (network-script network-route)
  (vif-script vif-route)

	Xen Hypervisor		XCP
Toolstack / Console	Default / XL (XM)	Libvirt / VIRSH	XAPI / XE
	Increased level of functionality and integration with other components		
Get Binaries from …	Linux Distros	Linux Distros	Debian & Ubuntu XCP from Xen.org
Products	Oracle VM	Huawei UVP	Citrix XenServer
Used by …	amazon web services™	Many Others	the rackspace cloud

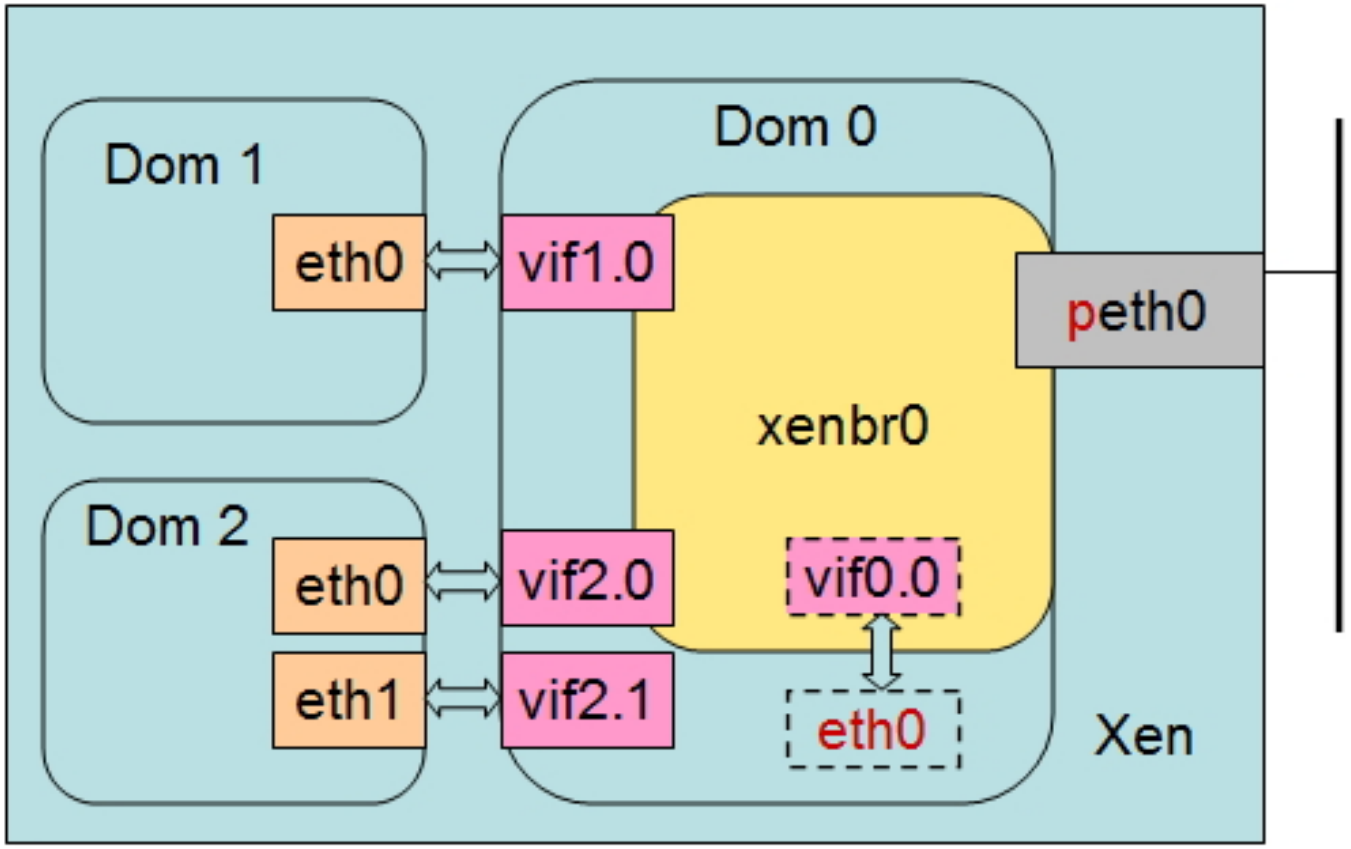



Xen hypervisor
Xen Cloud Platform
xm
xe
XEND
XAPI
XL
libxenlight
libxc
libxenguest
xenstore

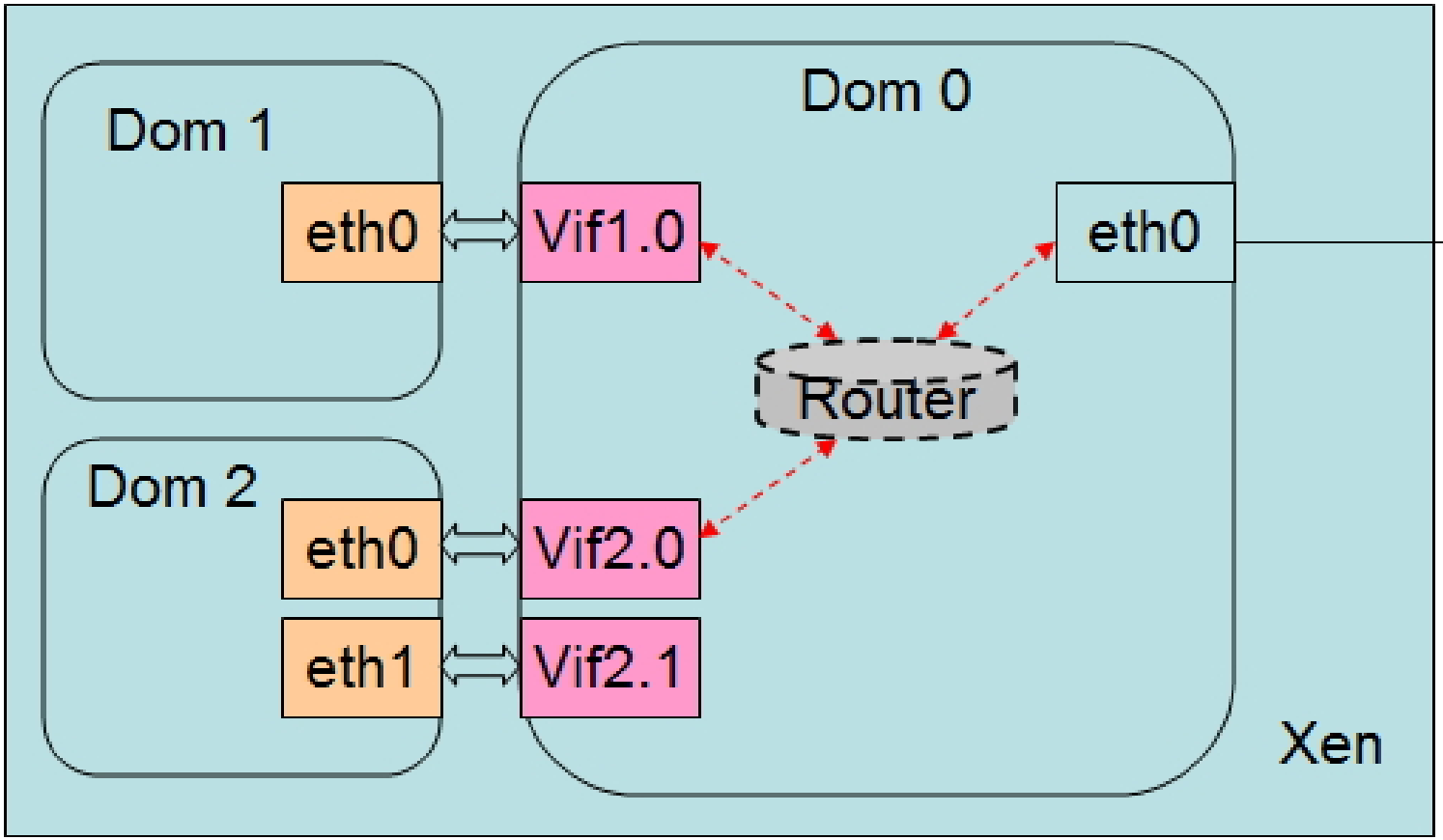

Userspace tools
virsh
libvirt
virt-viewer
libvirt
virt-manager
libvirt
libvirtd
libvirt
Hypervisor
Dom 0
Guest
Dom U
Guest
Xen
Kernel
KVM module
Qemu
Guest
Guest
KVM
Kernel
Qemu
Guest
Qemu
Guest
Qemu

Features	xl	xapi	libvirt
Purpose-built for Xen	✓	✓	✗
Basic VM Operations	✓	✓	✓
Managed Domains	✗	✓	✓
Live Migration	✓	✓	✓
PCI Passthrough	✓	✓	✓
Host Pools	✗	✓	✗
Flexible, Advanced Storage Types	✗	✓	✗
Built-in advanced performance monitoring (RRDs)	✗	✓	✗
Host Plugins (XAPI)	✗	✓	✗

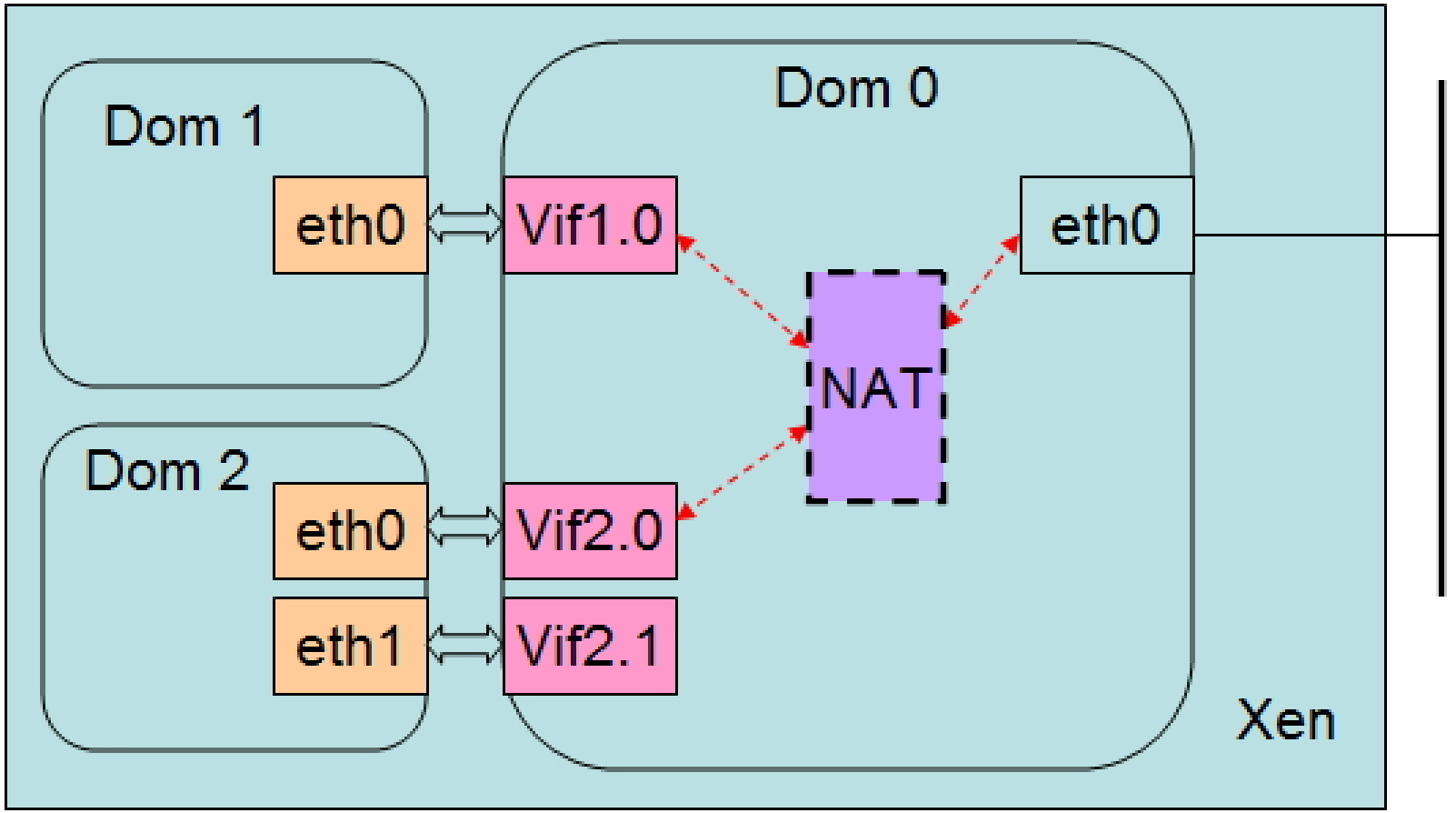
桥接


Dom 1
eth0
Dom 0
vif1.0
peth0
xenbr0
Dom 2
eth0
vif2.0
vif0.0
eth1
vif2.1
eth0
Xen

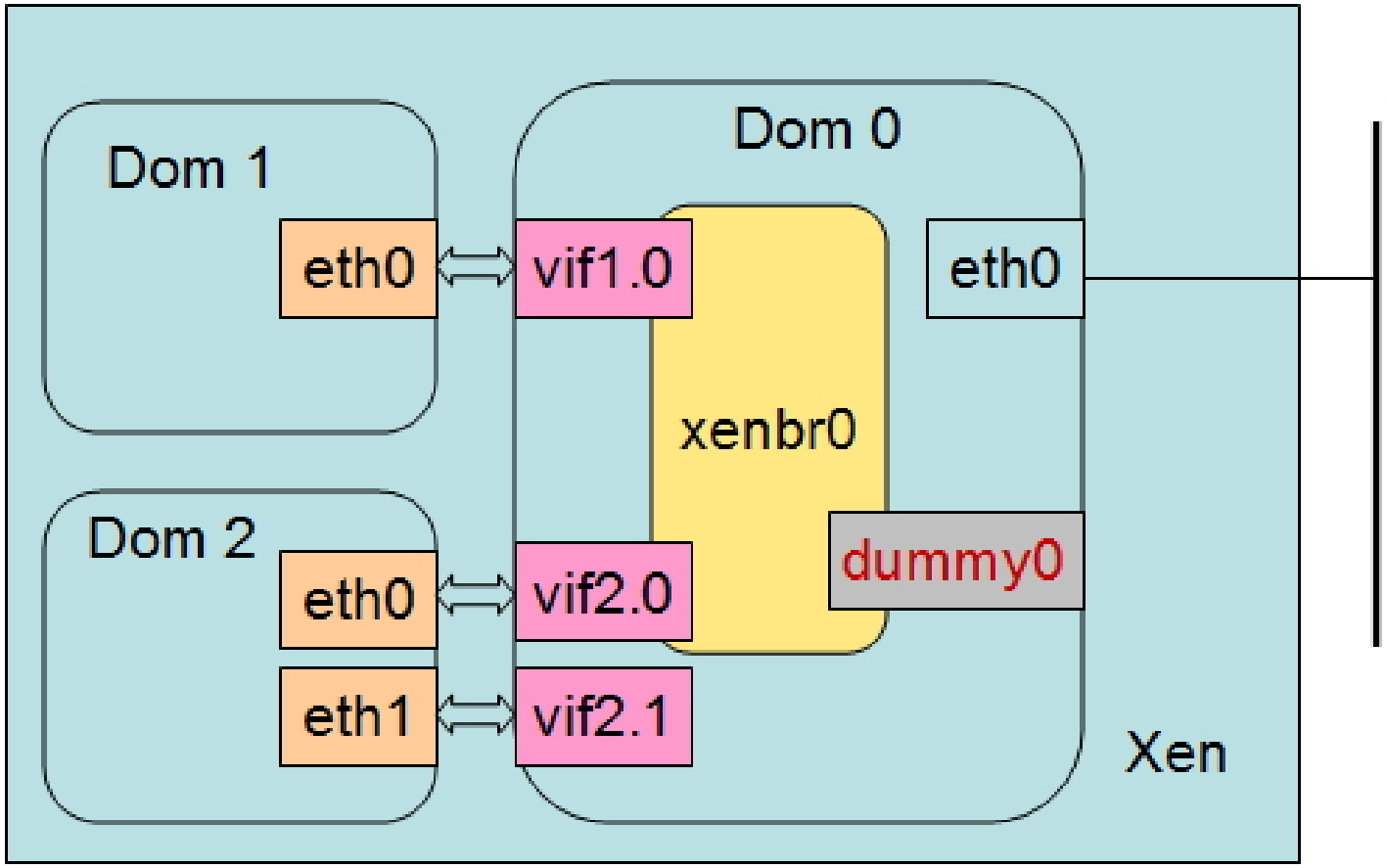
路由模式


Dom 1
eth0
Dom 0
Vif1.0
eth0
Router
Dom 2
eth0
Vif2.0
eth1
Vif2.1
Xen

NAT模式

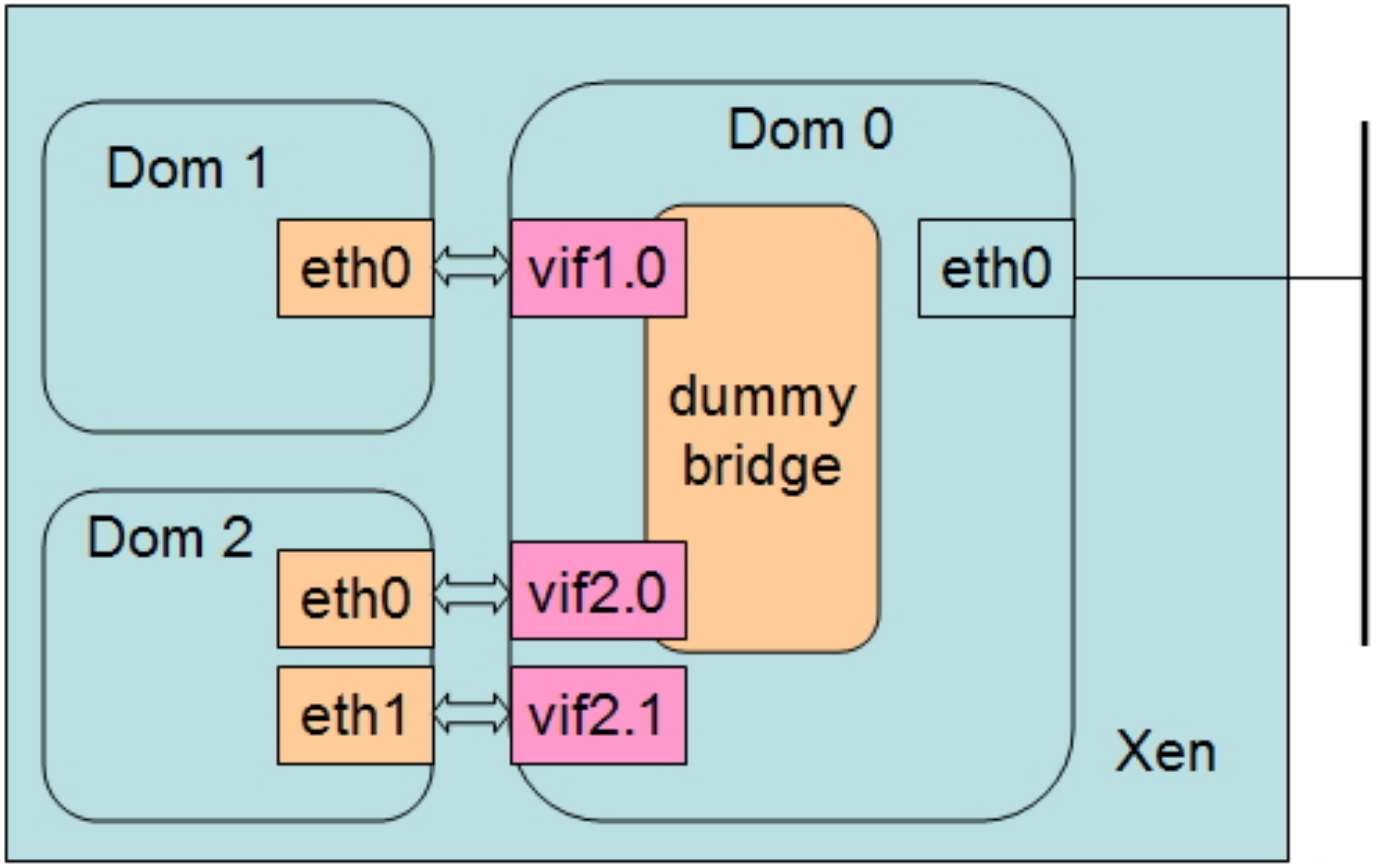


伪网络接口(dummy0)



仅主机

伪网桥


Dom 1
eth0
Dom 0
vif1.0
eth0
dummy
bridge
Dom 2
eth0
vif2.0
eth1
vif2.1
Xen

- 使用brctl命令创建桥接设备
- # brctl addbr dumbr0
- # brctl setfd dumbr0 0
- # ip link set dumbr0 up

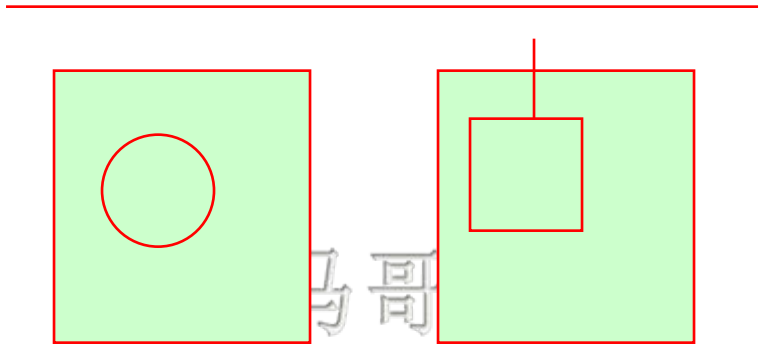
- vif = ['bridge=dumbr0',]

马哥教育

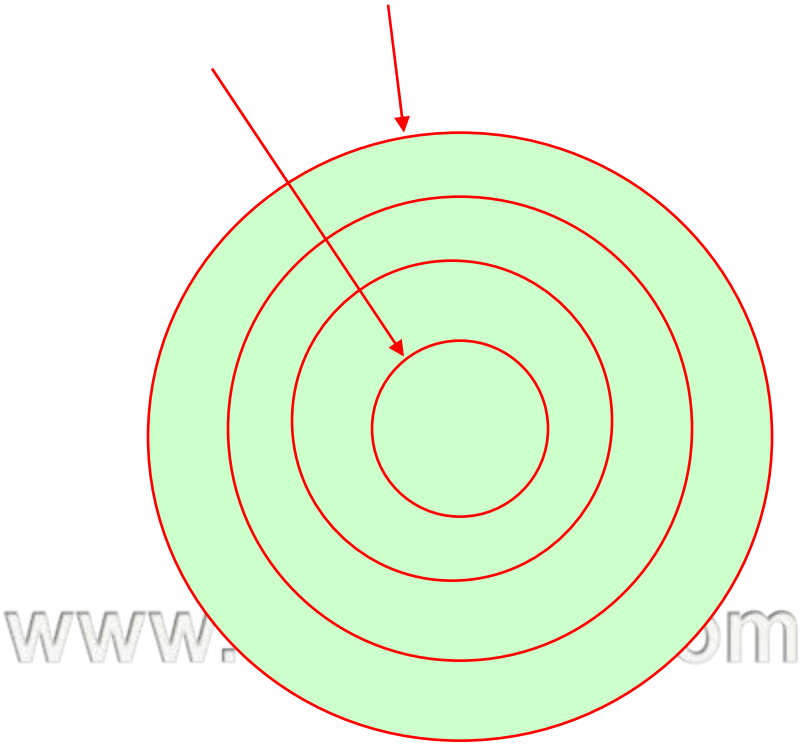
KVM

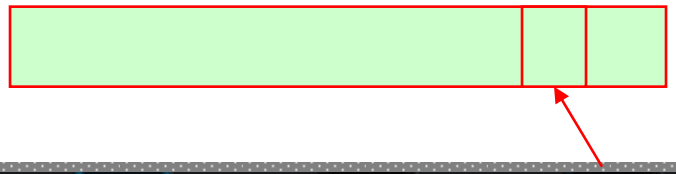
主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

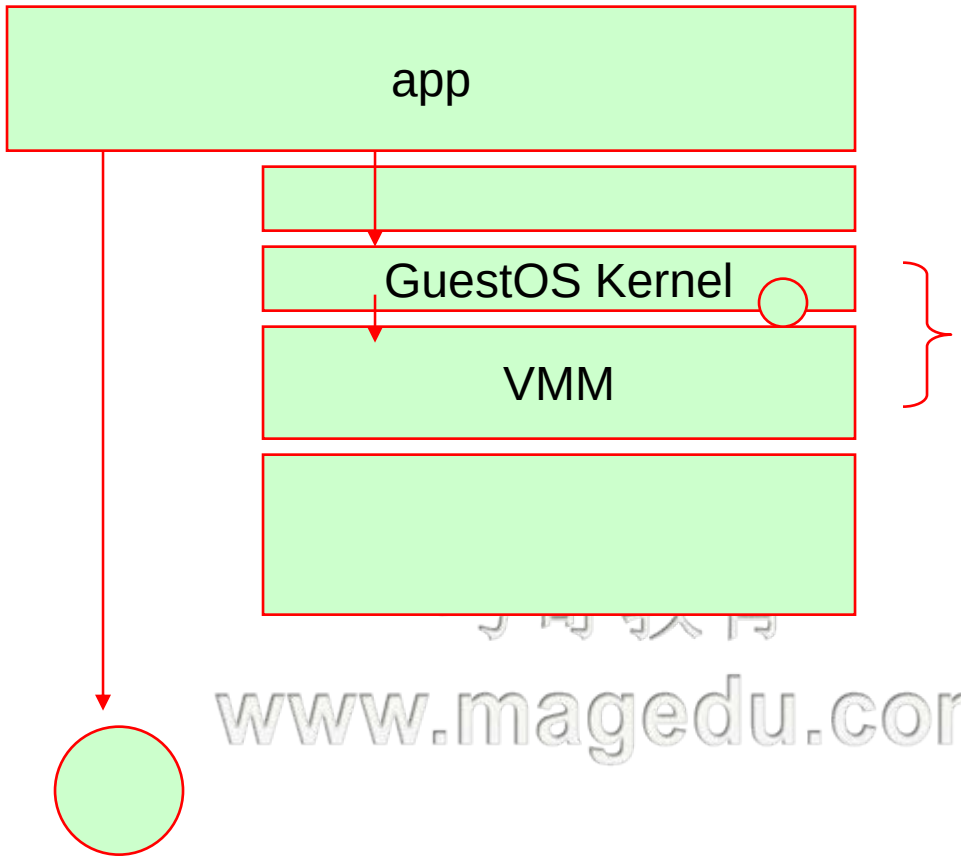
-incoming



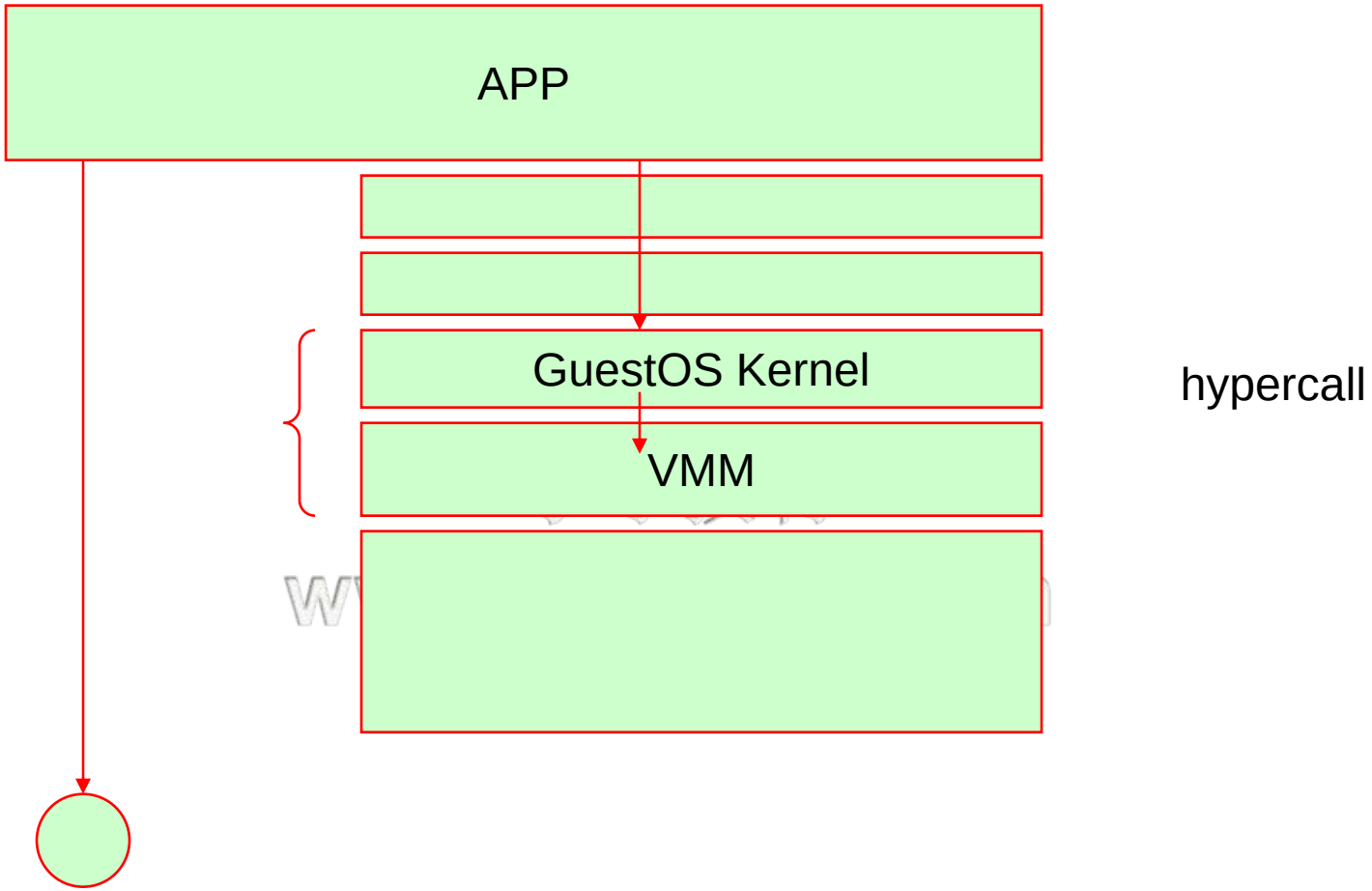
- KVM: Kernel-based Virtual Machine
  - CPU
  - Memory
  - IO

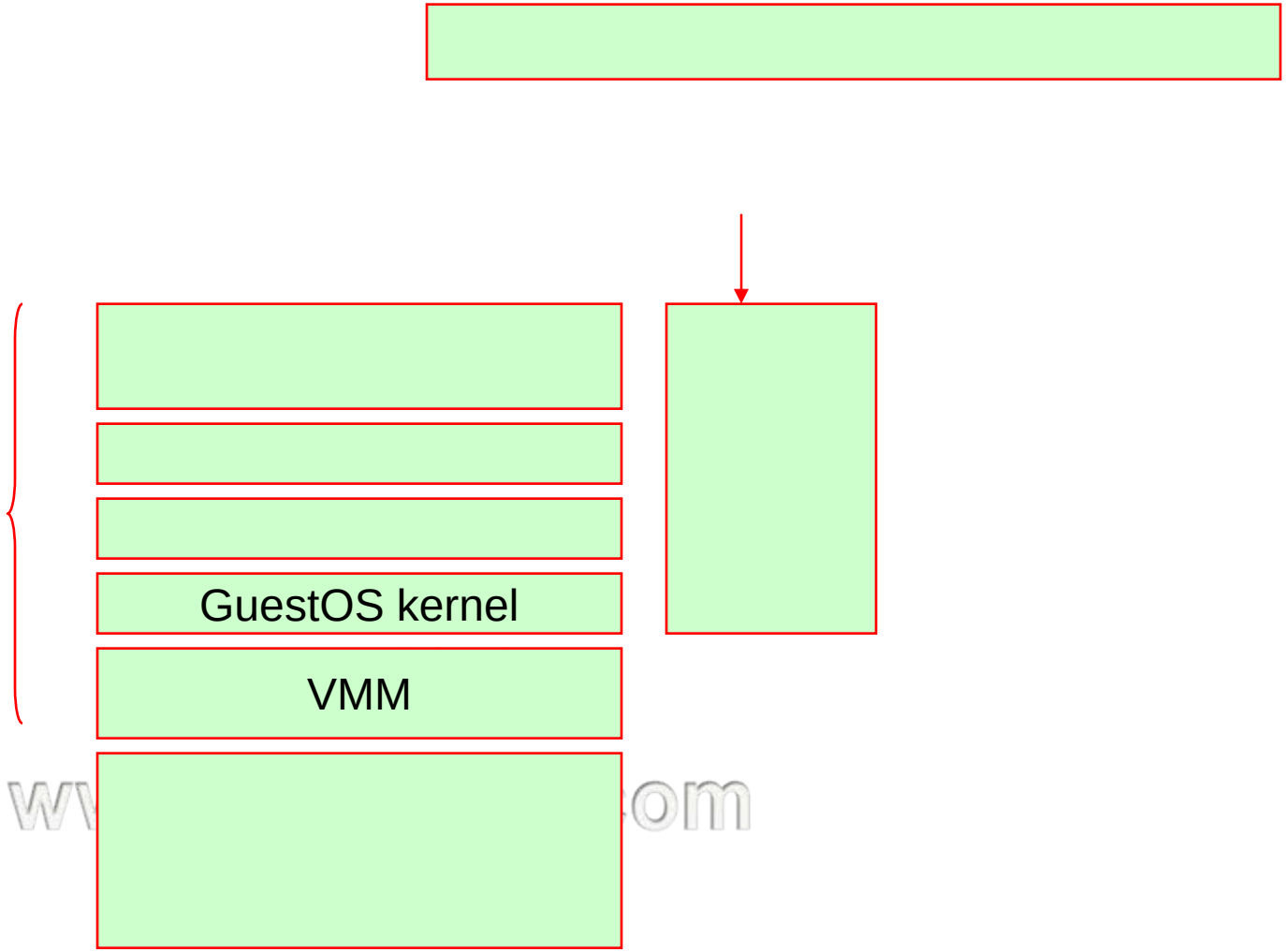




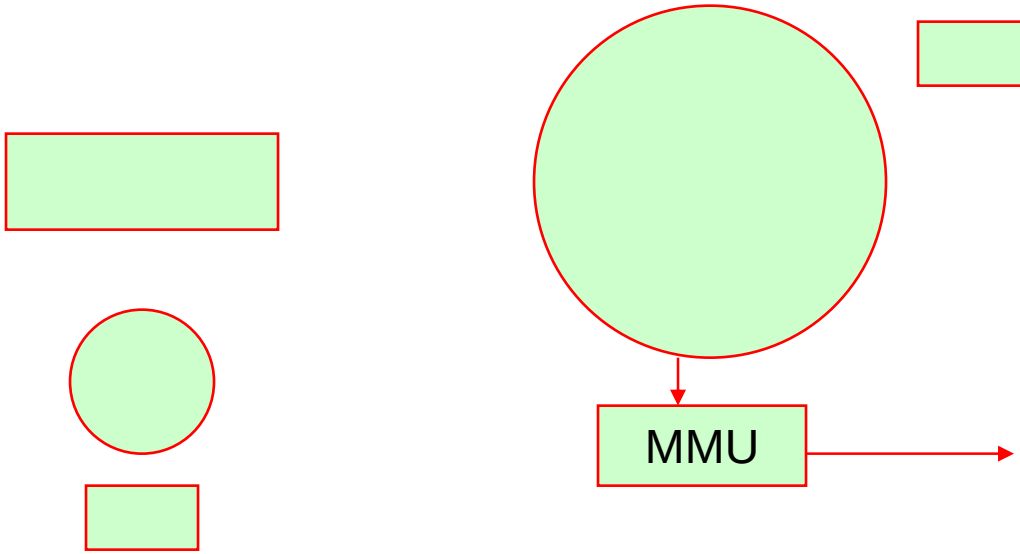


para-

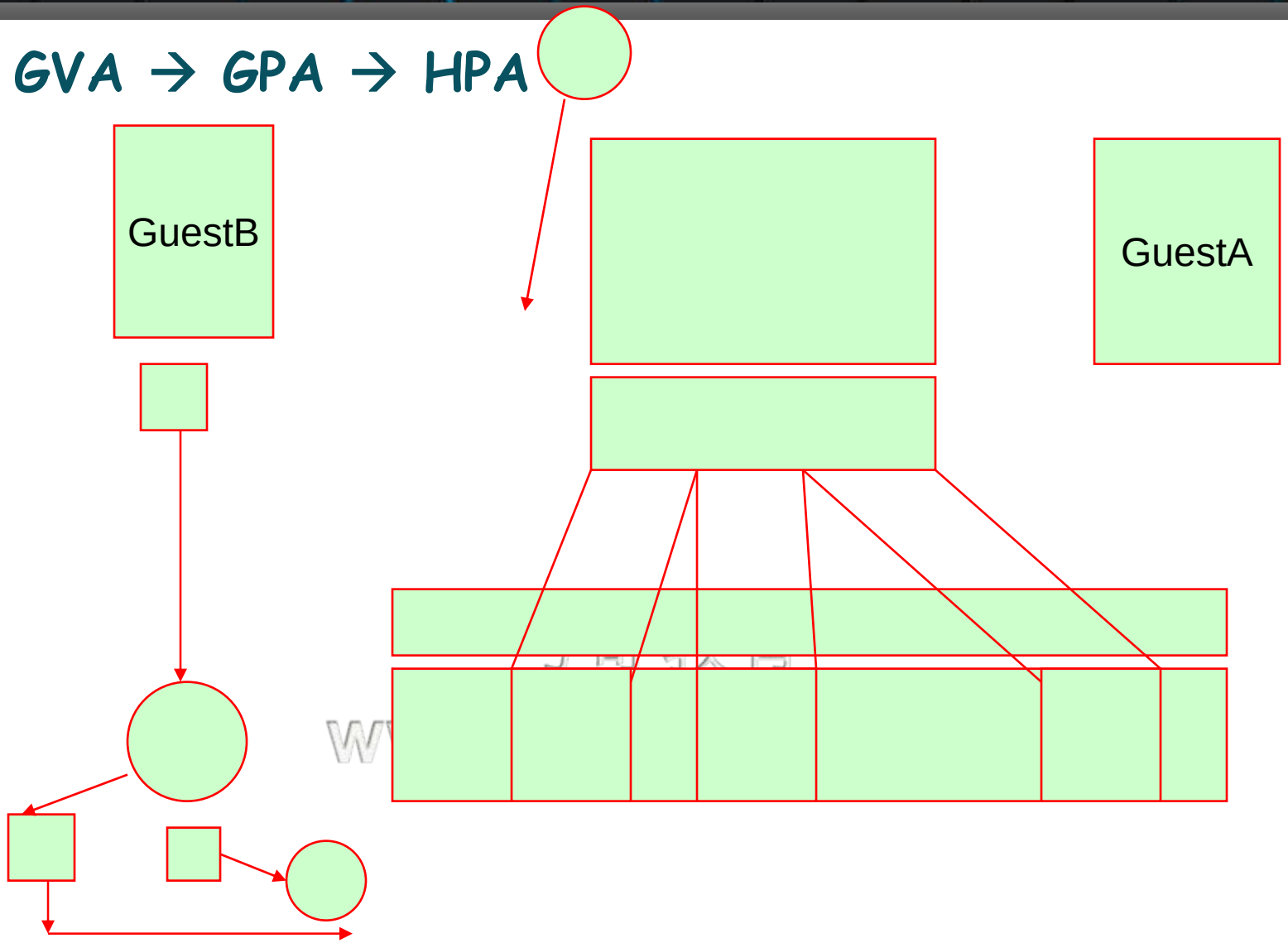



GuestOS kernel
VMM

- page table

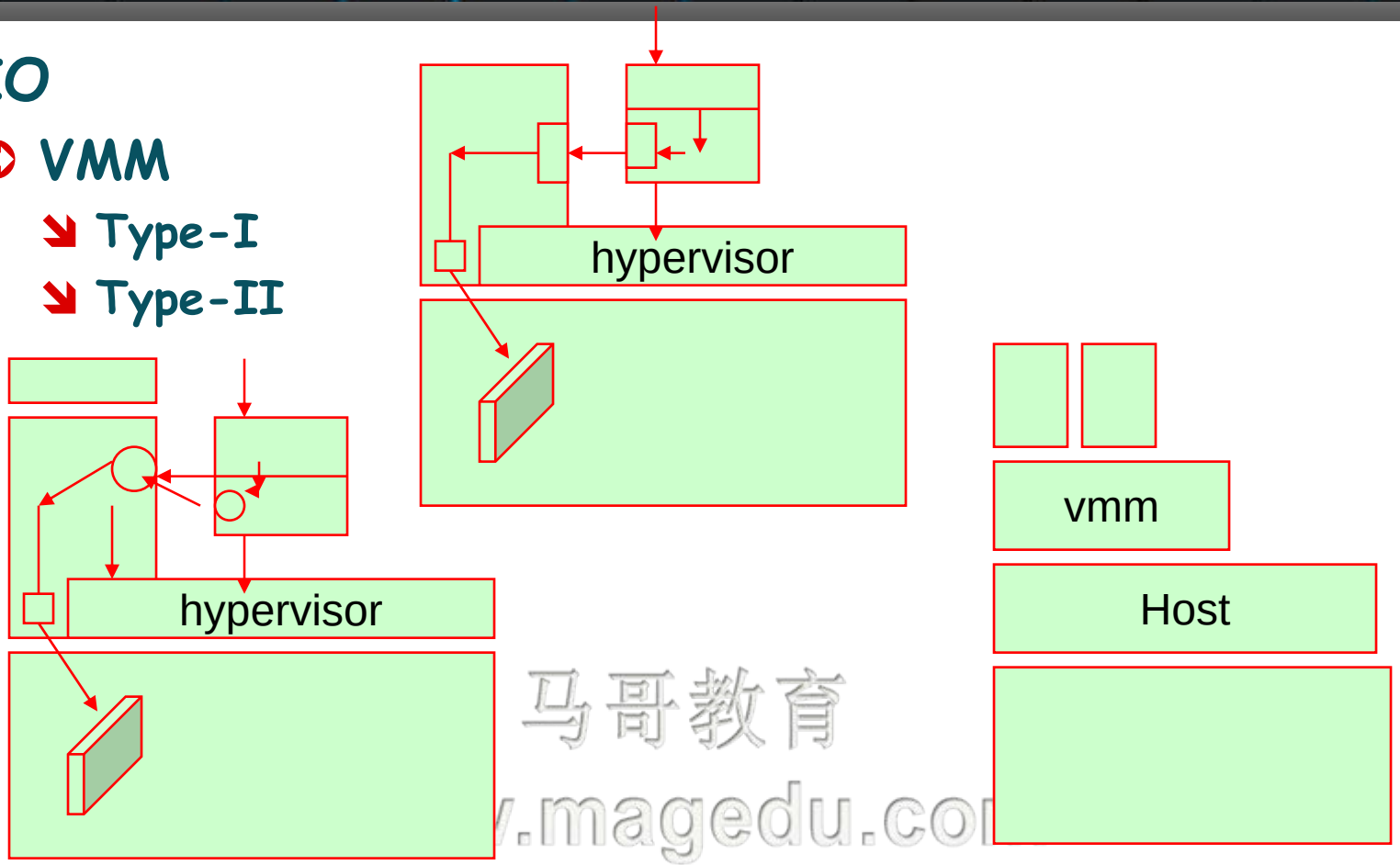


- GVA → GPA → HPA

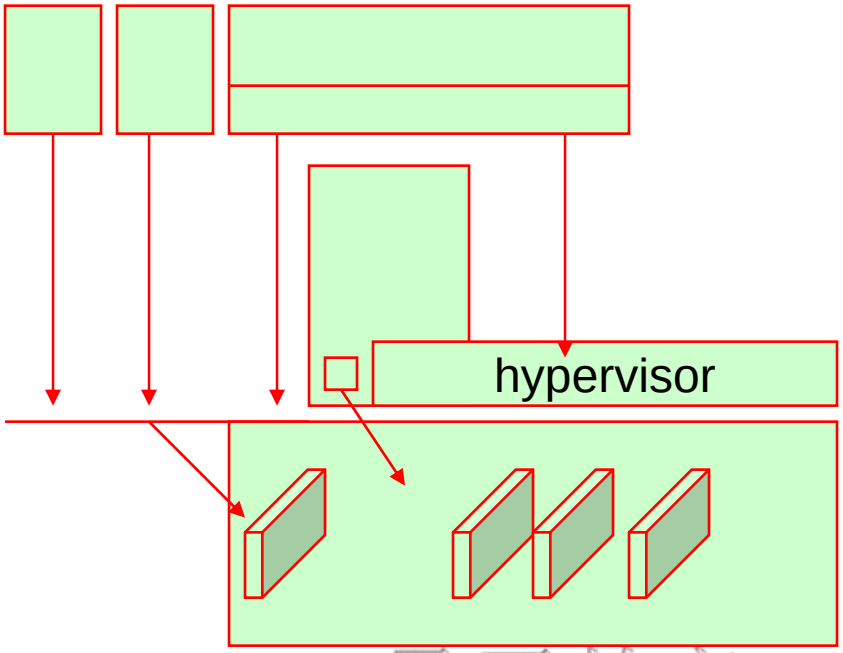


半虚拟化

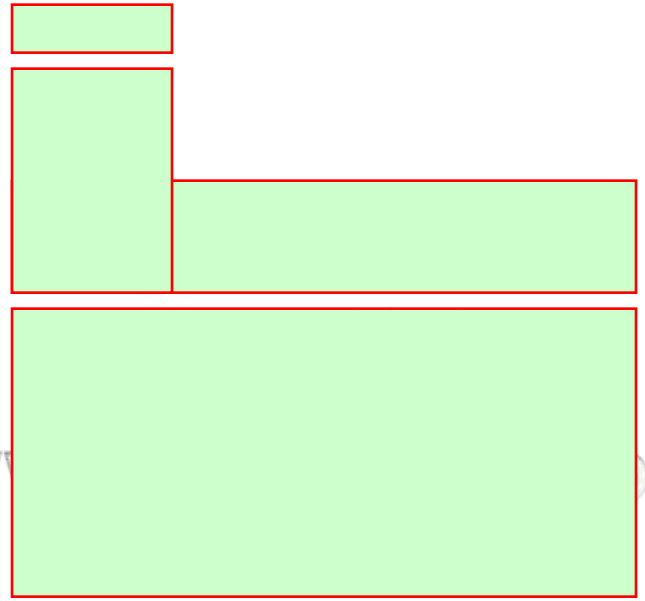
- IO
  - VMM
    - Type-I
    - Type-II



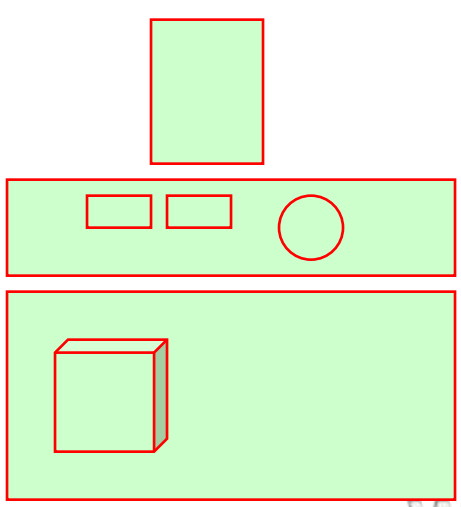
Hybrid

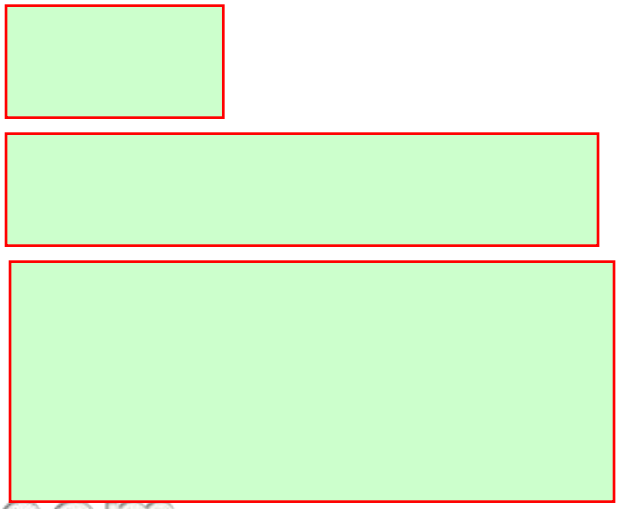


PF: → VF

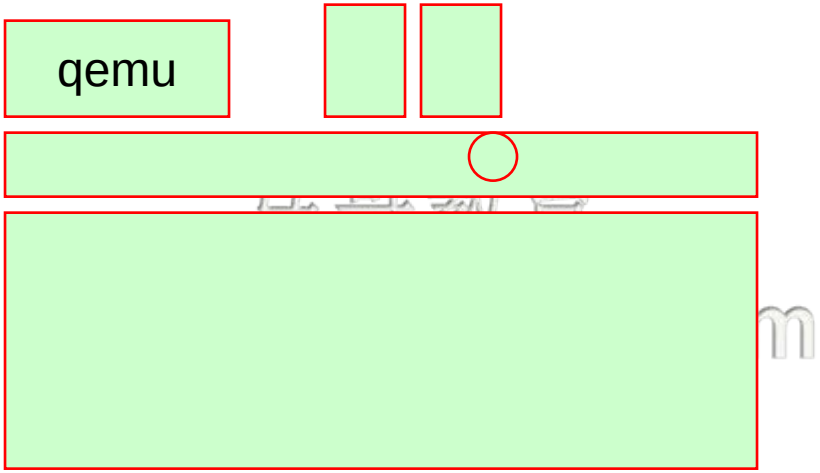


- Kernel-based Virtual Machine
- qemu：创建并管理机的工具
  - cpu：模拟器，虚拟化，kqemu

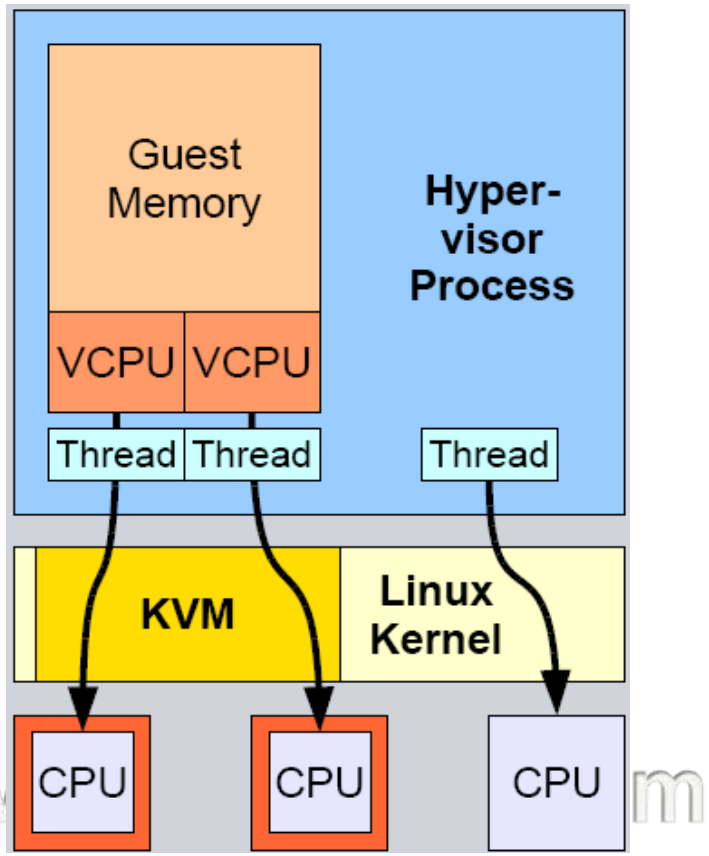




- qemu → kqemu (加速器)

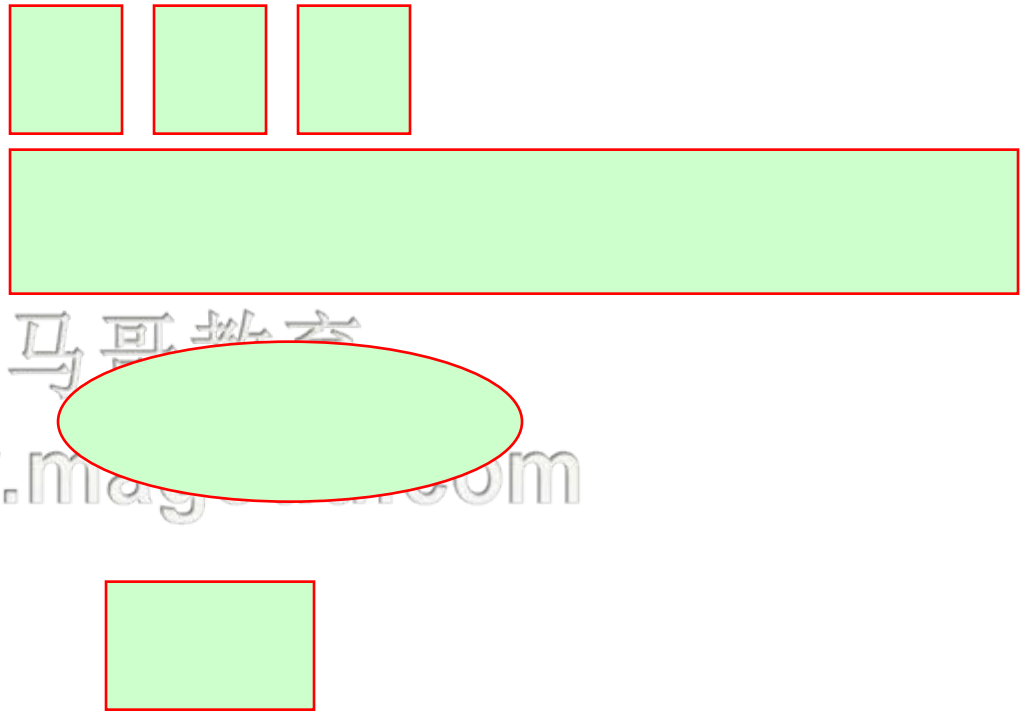


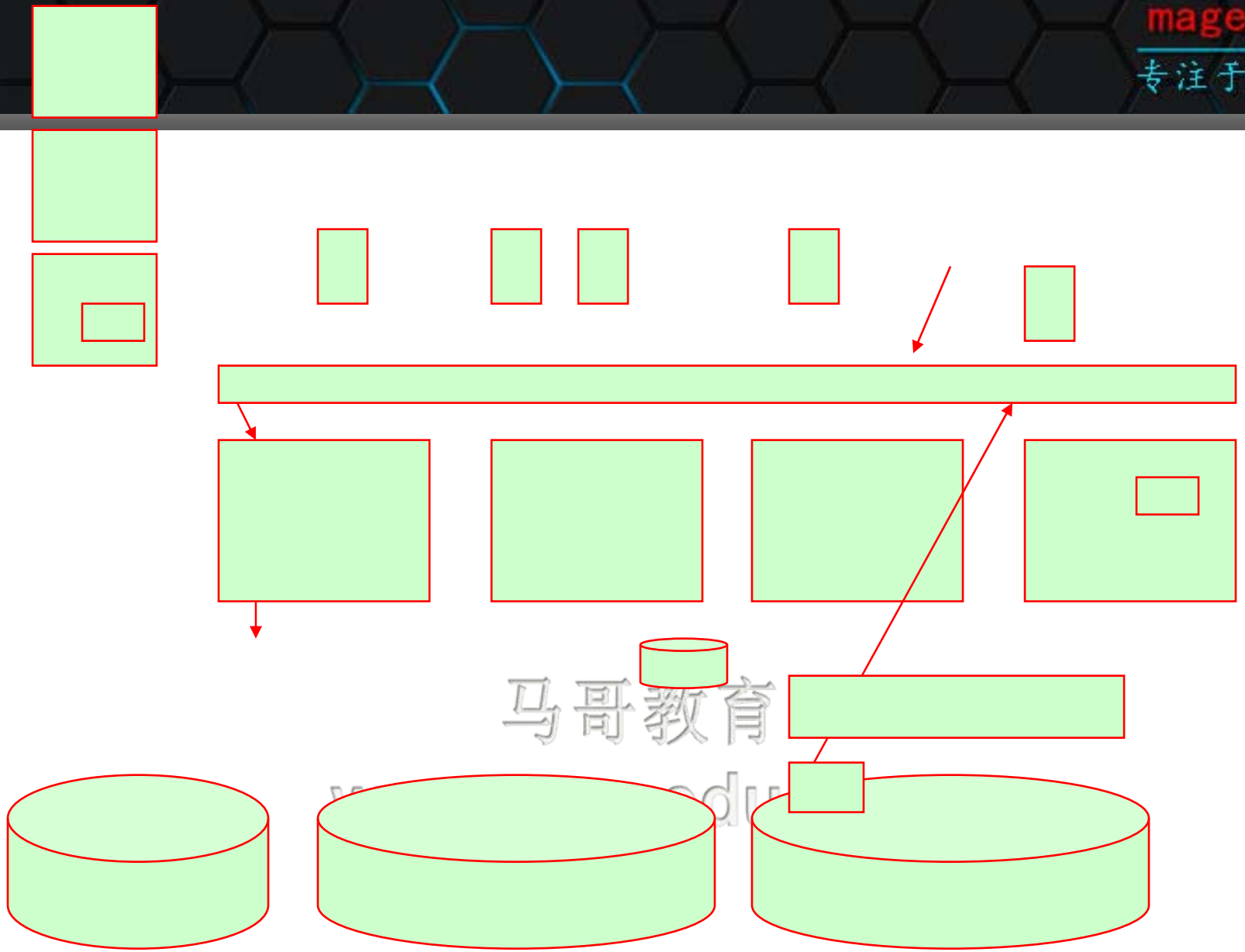
KVM架构


Guest Memory
Hyper-visor Process
VCPU
VCPU
Thread
Thread
Thread
KVM
Linux Kernel
CPU
CPU
CPU

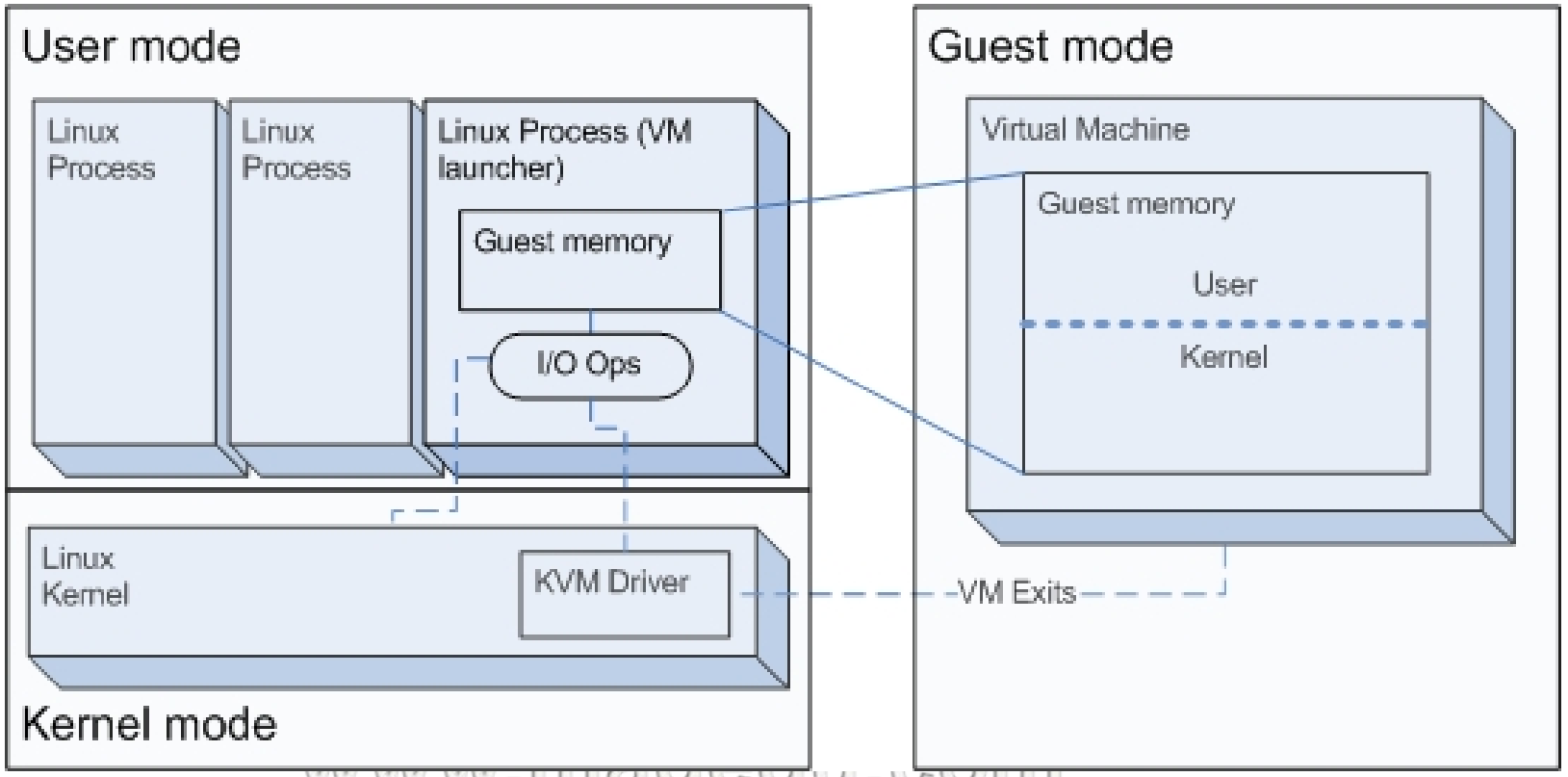
支持的GuestOS

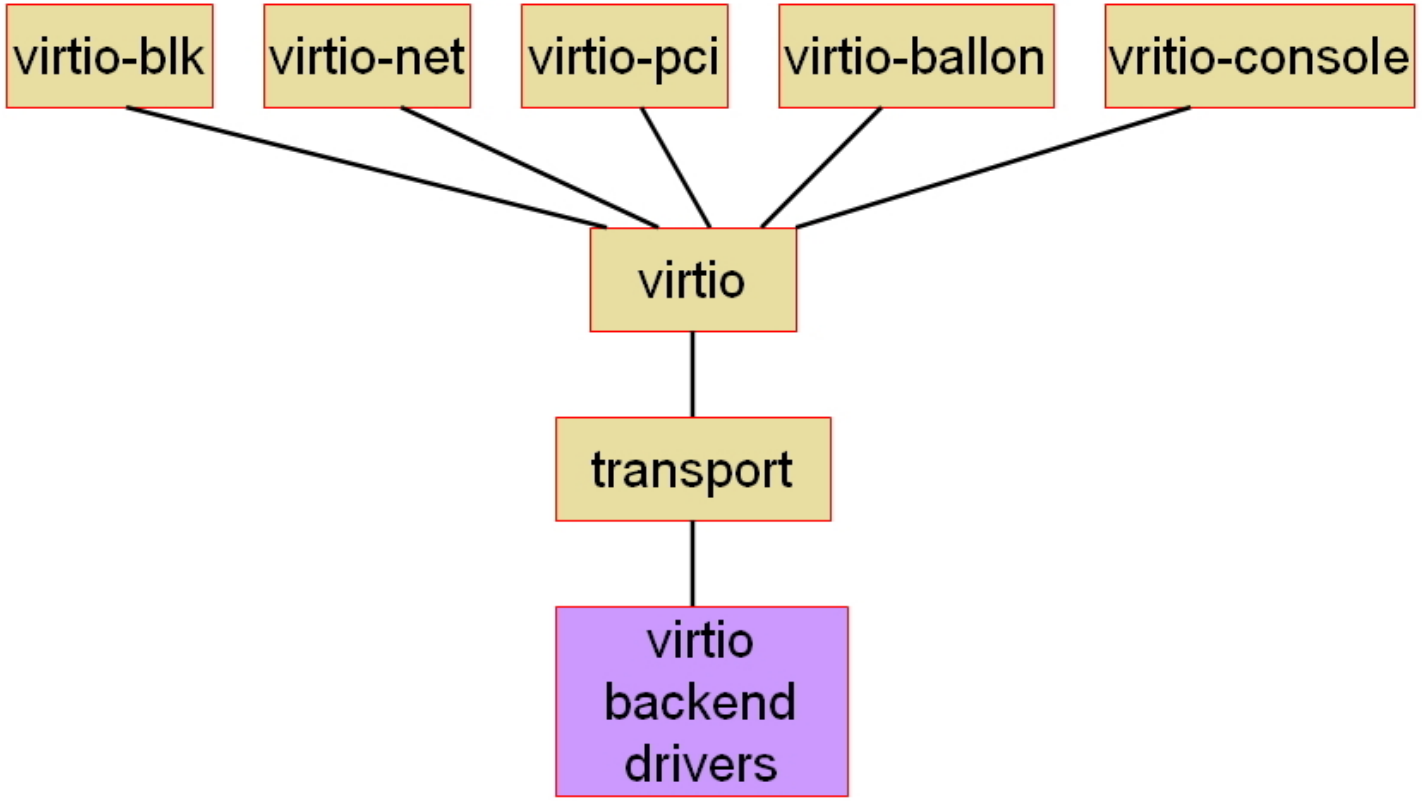
- KVM可以支持包括Linux、Windows、OpenBSD、FreeBSD、OpenSolaris、Solaris x86和MS DOS在内的众多操作系统用作GuestOS
- 此外，运行于RHEL系统上的KVM还通过了微软的的SVVP（Server Virtualization Validation Program）认证，从而运行于RHEL或RHEV-H上的KVM中的Windows可以获得微软的全部商业支持



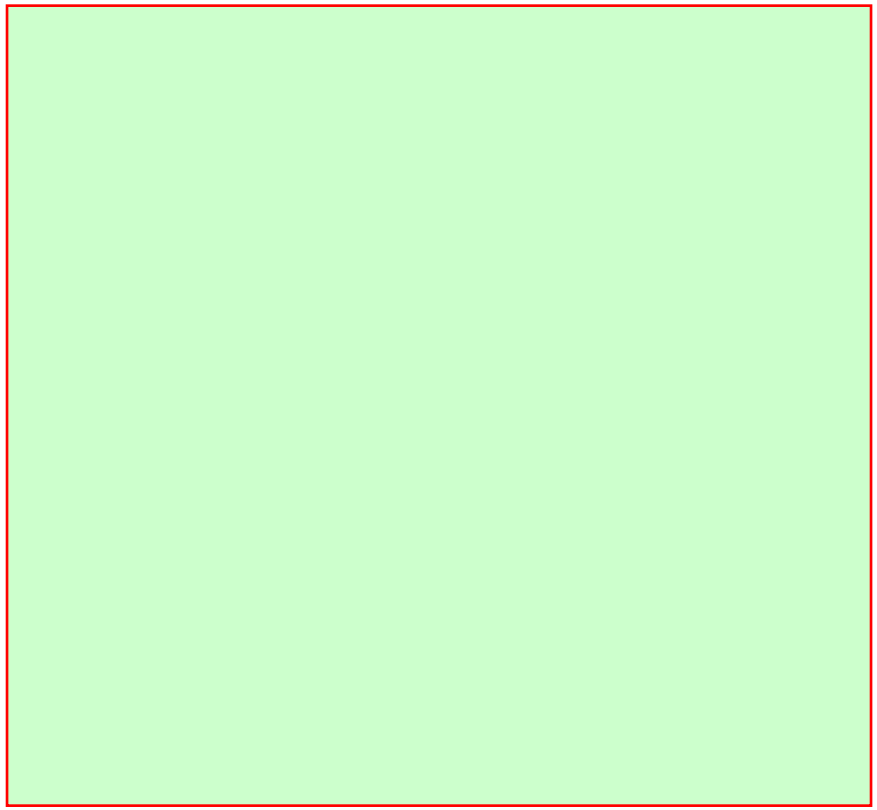


KVM架构


User mode
Linux Process
Linux Process
Linux Process (VM launcher)
Guest memory
I/O Ops
Linux Kernel
KVM Driver
Kernel mode
Guest mode
Virtual Machine
Guest memory
User
Kernel
VM Exits

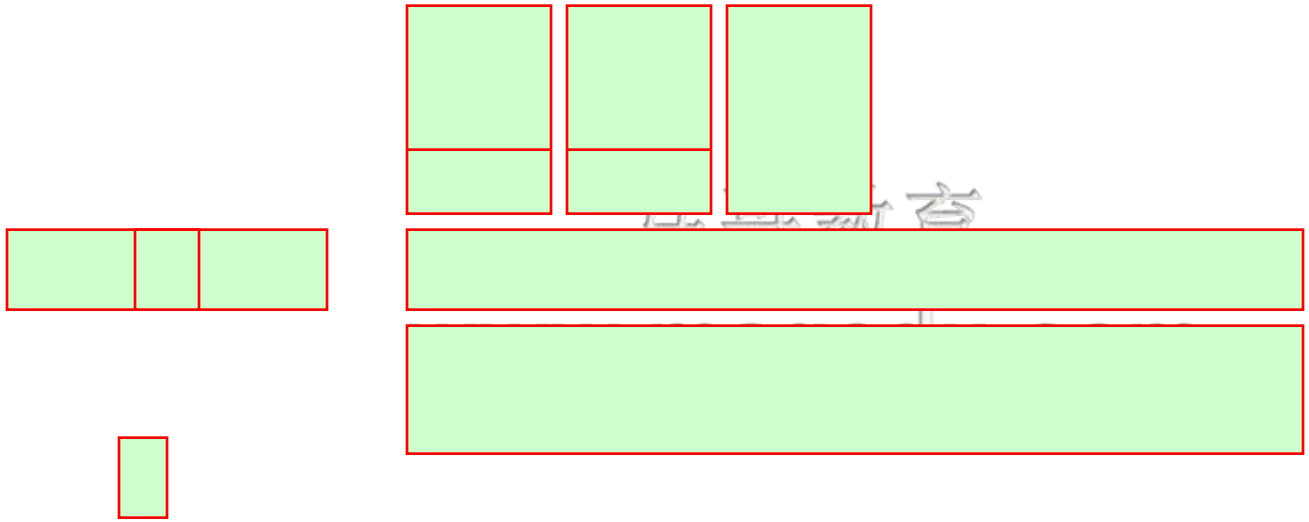

virtio-blk
virtio-net
virtio-pci
virtio-ballon
vritio-console
virtio
transport
virtio backend drivers

Category	Fully Virtualized	Paravirtualized	Host Pass-through
CPU, MMU	7%	not applicable	97% (Hardware Virtualization with Extended Page Tables(Intel) or Nested Page Tables (AMD)) 85% (Hardware Virtualization with shadow page tables)
Network I/O (1GB LAN)	60% (e1000 emulated NIC)	75% (`virtio-net`)	95%
Disk I/O	40% (IDE emulation)	85% (`virtio-blk`)	95%
Graphics (non-accelerated)	50% (VGA or Cirrus)	not applicable	not applicable
Time accuracy (worst case, using recommended settings without NTP)	95% - 105% (where 100% = accurate)	100% (`kvm-clock`)	not applicable



- KVM是一个混合类型的VMM，它能够以模拟方式支持硬件的完全虚拟化，也能够在GuestOS中安装驱动程序进而支持部分硬件的半虚拟化
  - 对网络设备和块设备来讲，半虚拟化方式能够极大地提升设备性能
- Red Hat联手IBM同Linux社区开发出了一种独立于VMM的半虚拟化驱动程序标准VirtIO，根据这种标准开发的半虚拟化设备驱动程序可兼容地运行于多种不同的VMM上，从而提高了相关VMM之间的互操作性
- VritIO驱动已经包含于2.6.25及以后版本的Linux内核中，RHEL系列的操作系统4.8+、5.3+及6.0+的内核业已支持此驱动，而且，Red Hat专门为GuestOS模式的Windows提供的VirtIO驱动程序也通过了微软的WHQL认证

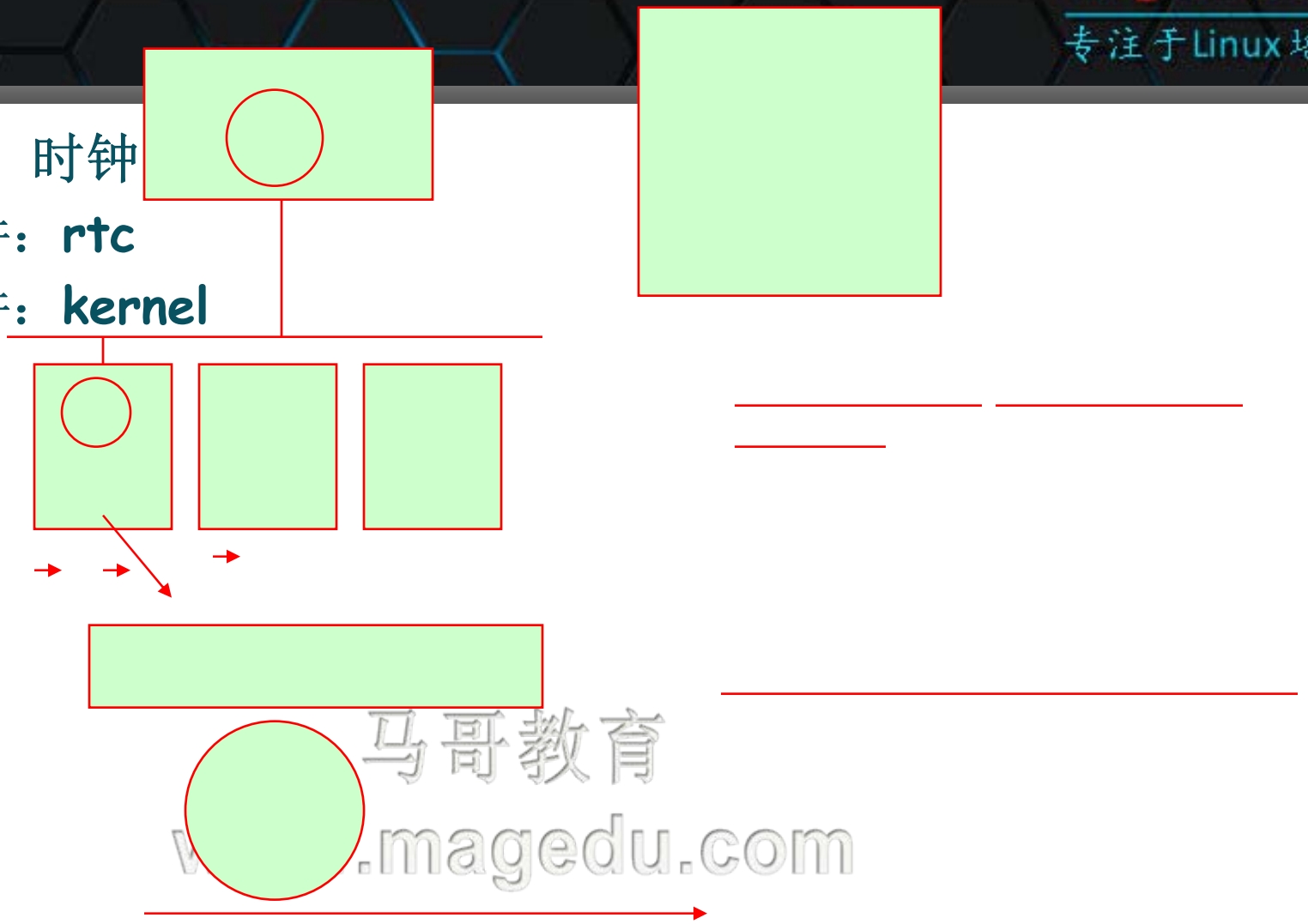
KSM: KVM Shared Memory



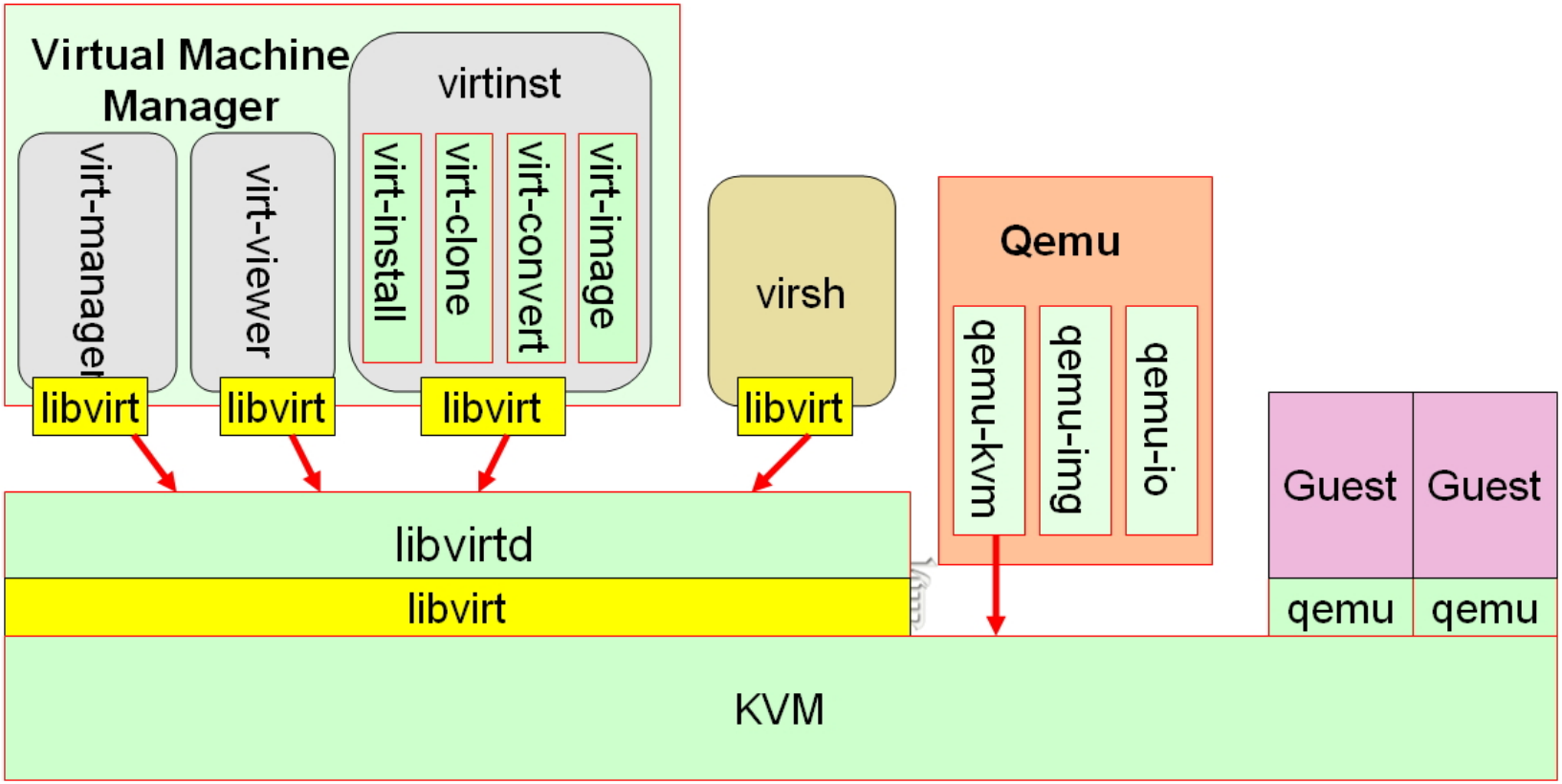
实时迁移

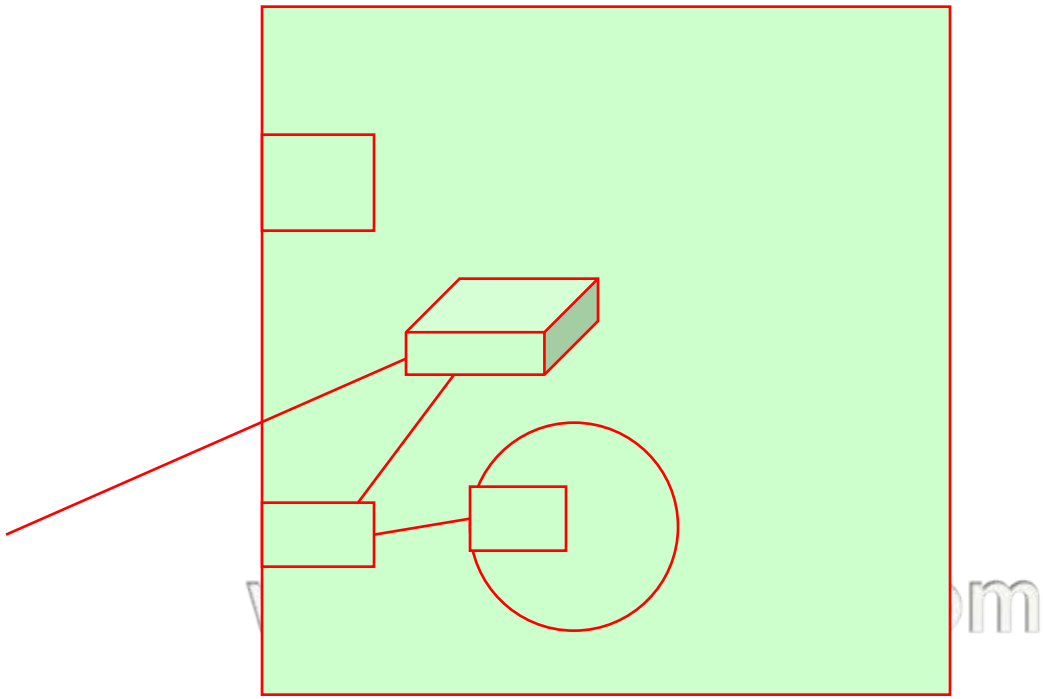
- 实时迁移的需求：
  - GuestOS映像文件放轩在共享存储上，如iSCSI、NFS或GFS2等；
  - 目标物理主机的KVM要能够兼容源物理主机的KVM及其准备迁移的GuestOS；
  - 在两台主机上，共享存储必须挂载在同一位置，且挂载的目录名必须一致；
  - 两台物理主机的CPU需要具有相同类型的特性；
  - 虚拟机没使用透传I/O；
  - 两台物理主机的时间要同步；
  - 两台物理主机必须有一致的网络配置，且所有桥接和网络配置必须完全一致；

- Linux：时钟
  - 硬件：rtc
  - 软件：kernel

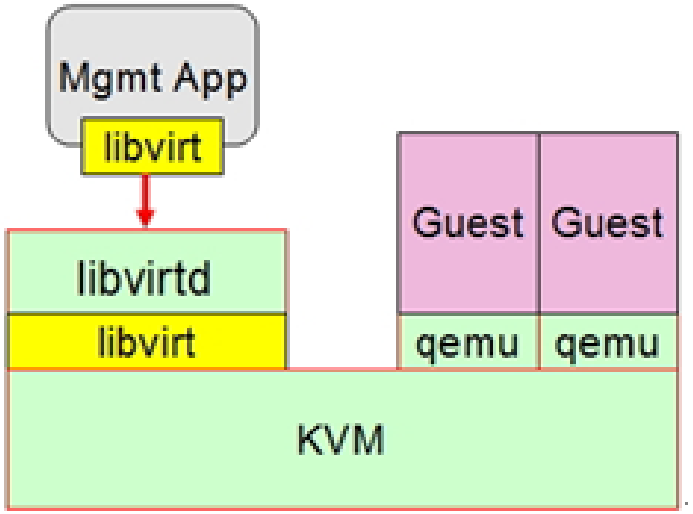


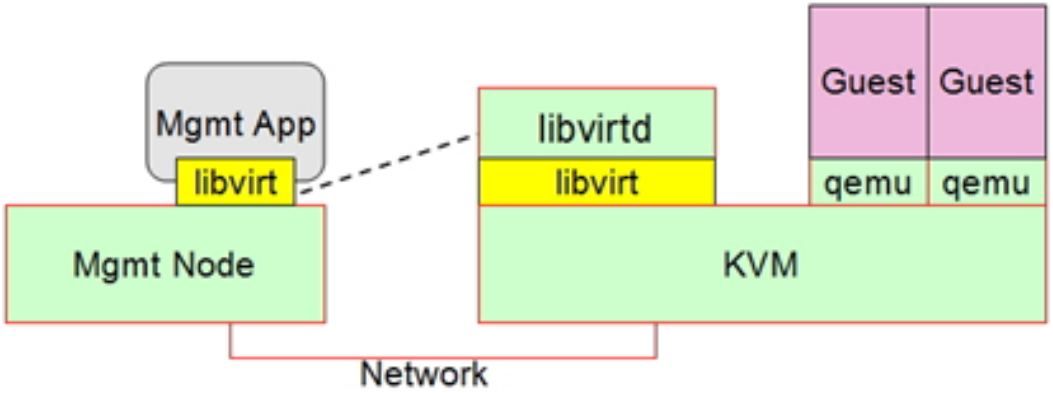
KVM管理工具


Virtual Machine Manager
virt-manager
virt-viewer
virtinst
virt-install
virt-clone
virt-convert
virt-image
virsh
Qemu
qemu-kvm
qemu-img
qemu-io
Guest
Guest
qemu
qemu
libvirt
libvirt
libvirt
libvirt
libvirtd
libvirt
KVM



virsh的本地模式与远程模式





VMM(Virtual Machine Manager)

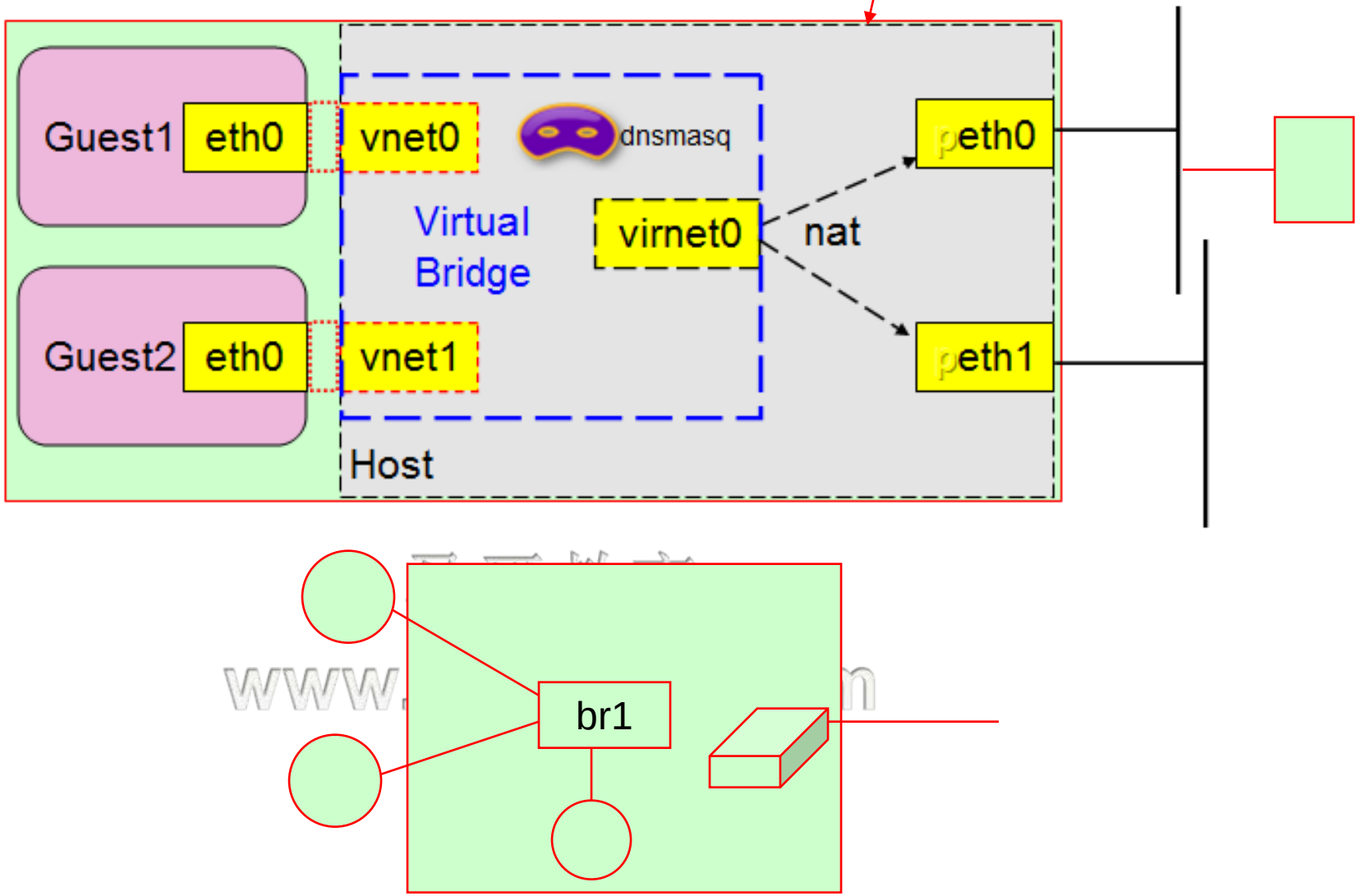
- 基于libvirt API管理工具主要有VMM和virsh。
  - “VMM（Virtual Machine Manager）”程序提供了virt-manager、virtinst和virt-viewer三个工具，均可以通过官方站点http://www.virt-manager.org获取
    - virt-manager：通过libvirt管理虚拟机的图形化工具，并可用于启动、关闭、新建或删除虚拟设备，连接至图形或串行控制台，以及查看资源占用率类的统计数据等；
    - virtinst：构建及安装虚拟的工具组件，包括virt-install（创建及安装虚拟机）、virt-clone（虚拟机克隆）、virt-convert（虚拟机格式转换）和virt-image（基于xml格式的镜像描述文件创建虚拟机）等；
    - virt-viewer：连接虚拟机的图形化客户端；
  - virsh：管理虚拟机的交互式shell，可用于创建、暂停、停止域等，也可实现虚拟设备的管理，是用于管理VKM虚拟机的最常用工具之一

QEMU主要有以下几个组成部分

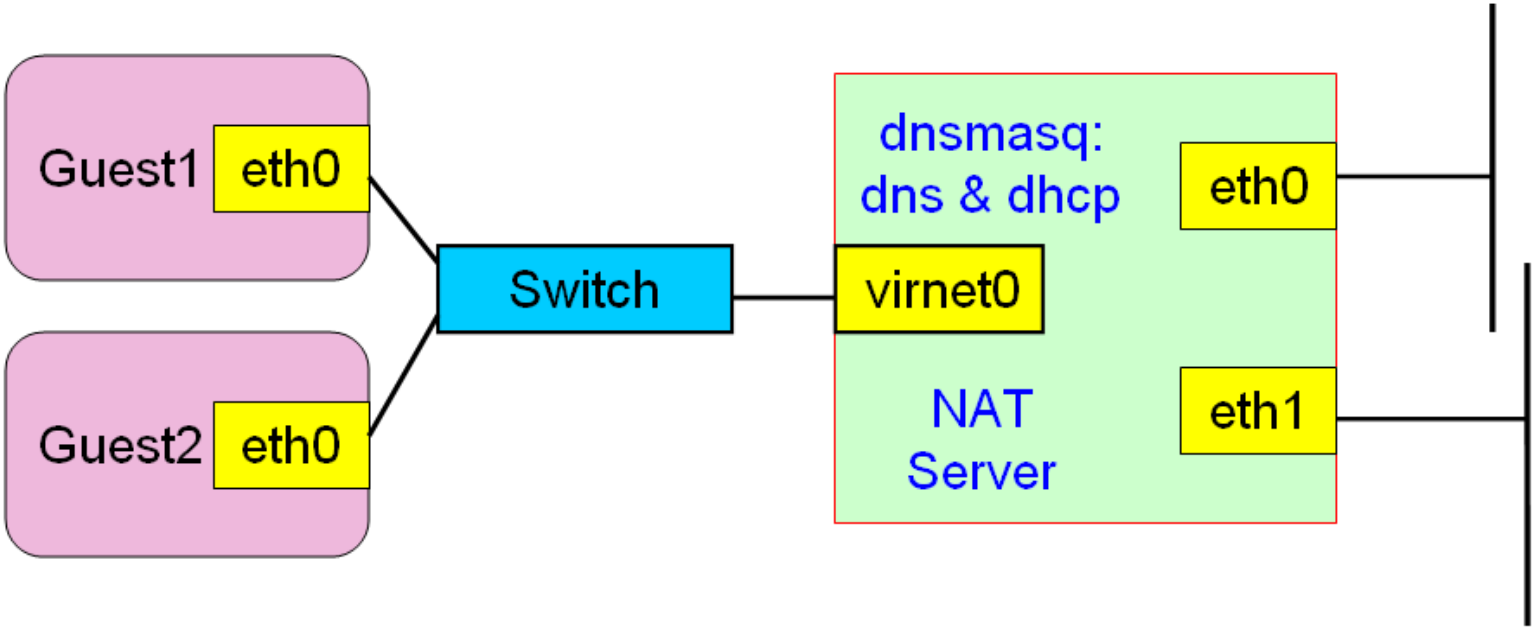
- 处理器模拟器（x86、PowerPC、Sparc等）；
- 仿真设备（显卡、网卡、硬盘等）；
- 关联仿真设备至真实设备的通用设备；
- 仿真主机的描述（如PC、Power Mac等）；
- 调试器；
- 与仿真器交互的用户接口；

Package Group	Description	Mandatory Packages	Optional Packages
Virtualization	*Provides an environment for hosting virtual machines*	*qemu-kvm*	*qemu-guest-agent, qemu-kvm-tools*
Virtualization Client	*Clients for installing and managing virtualization instances*	*python-virtinst, virt-manager, virt-viewer*	*virt-top*
Virtualization Platform	*Provides an interface for accessing and controlling virtual machines and containers*	*libvirt, libvirt-client, virt-who, virt-what*	*fence-virtd-libvirt, fence-virtd-multicast, fence-virtd-serial, libvirt-cim, libvirt-java, libvirt-qmf, libvirt-snmp, perl-Sys-Virt*
Virtualization Tools	*Tools for offline virtual image management*	*libguestfs*	*libguestfs-java, libguestfs-tools, virt-v2v*

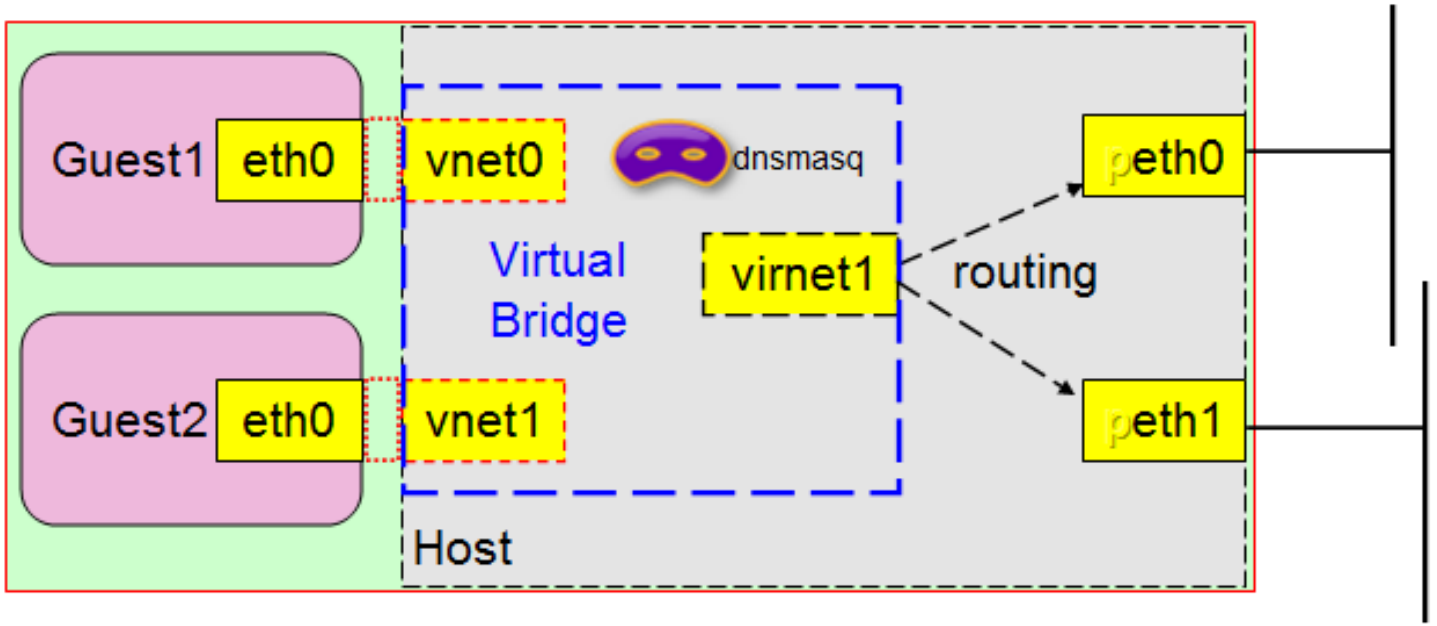
NAT模型（NAT Mode）



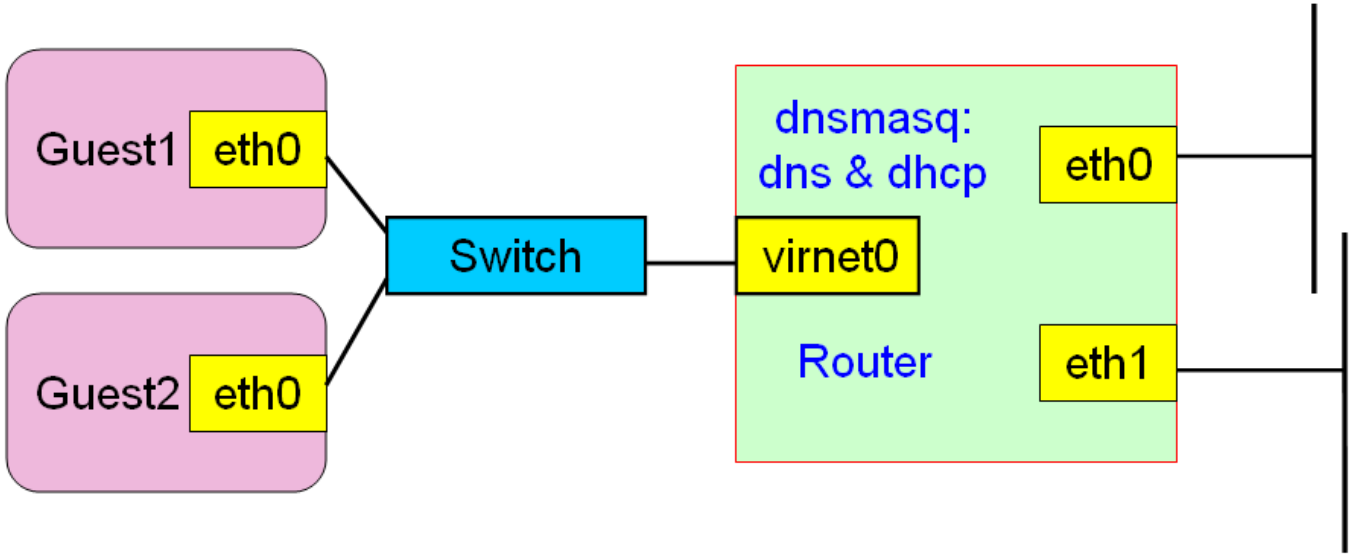
NAT实现的虚拟网络



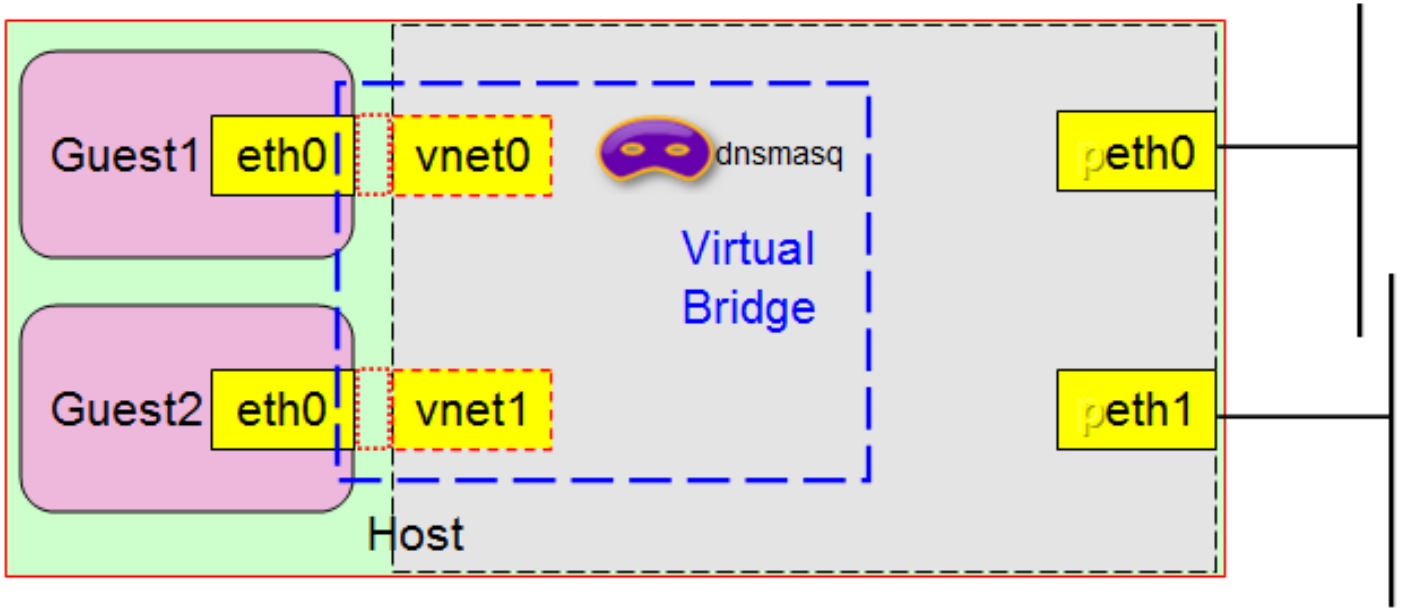
路由模型



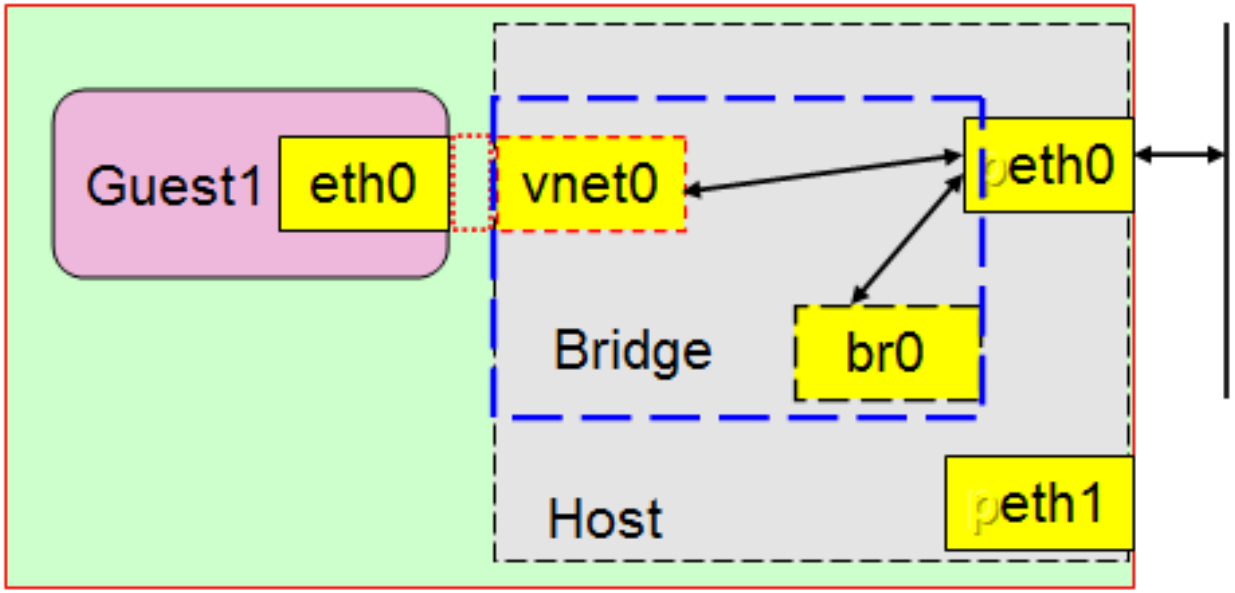
- 上图实现的网络模型



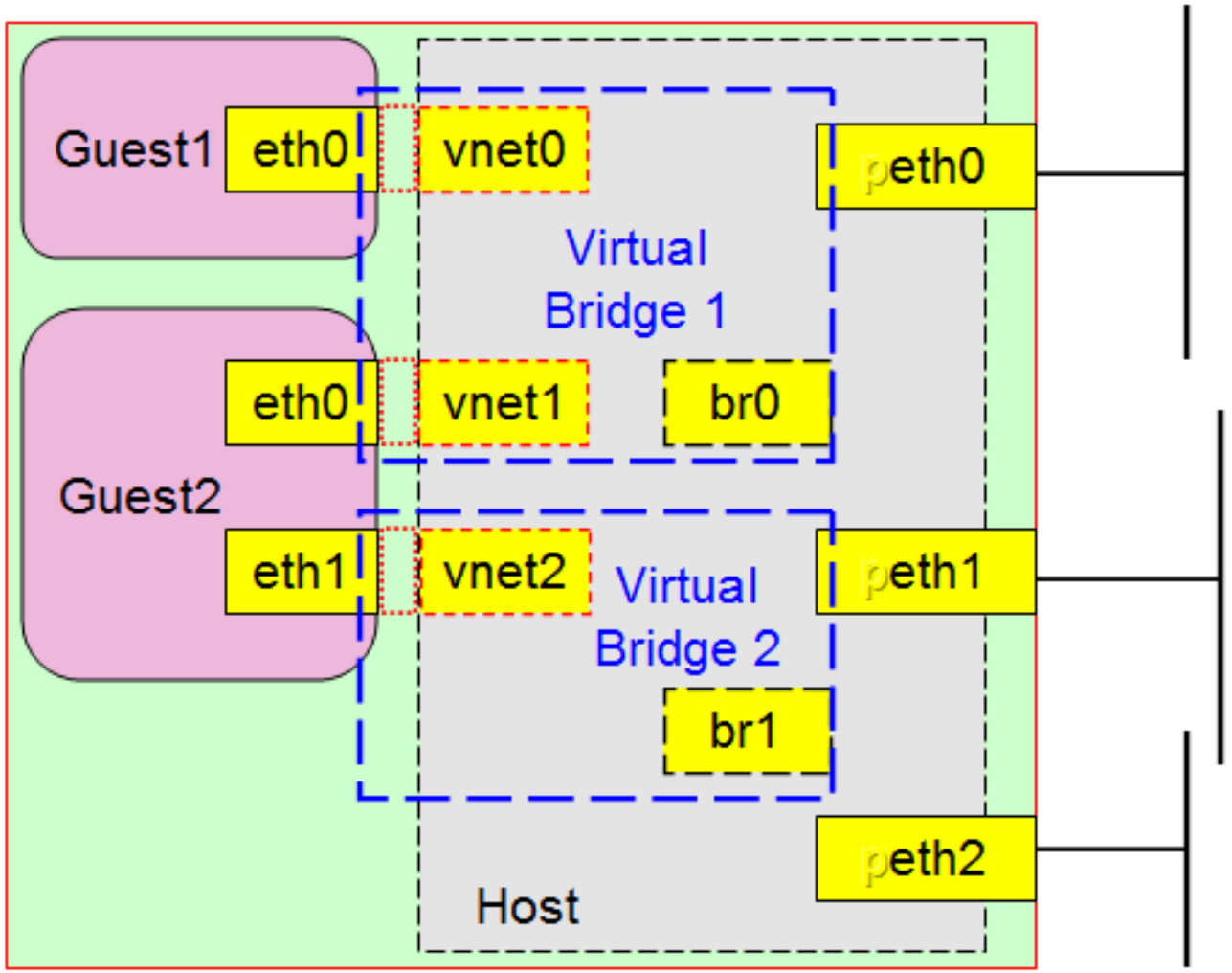
隔离模型



桥接模型



多路桥接


Guest1
eth0
vnet0
peth0
Virtual
Bridge 1
eth0
vnet1
br0
Guest2
eth1
vnet2
Virtual
Bridge 2
peth1
br1
peth2
Host

- Xen is a virtual machine monitor (hypervisor) that allows you to use one physical computer to run many virtual computers
- Virtual machine monitors provide a convenient way to use the same physical computer hardware for many different tasks
- The hypervisor controls the underlying hardware, allowing it to be used by many guest systems at once, and gives each guest system the illusion that it is running on its own private hardware

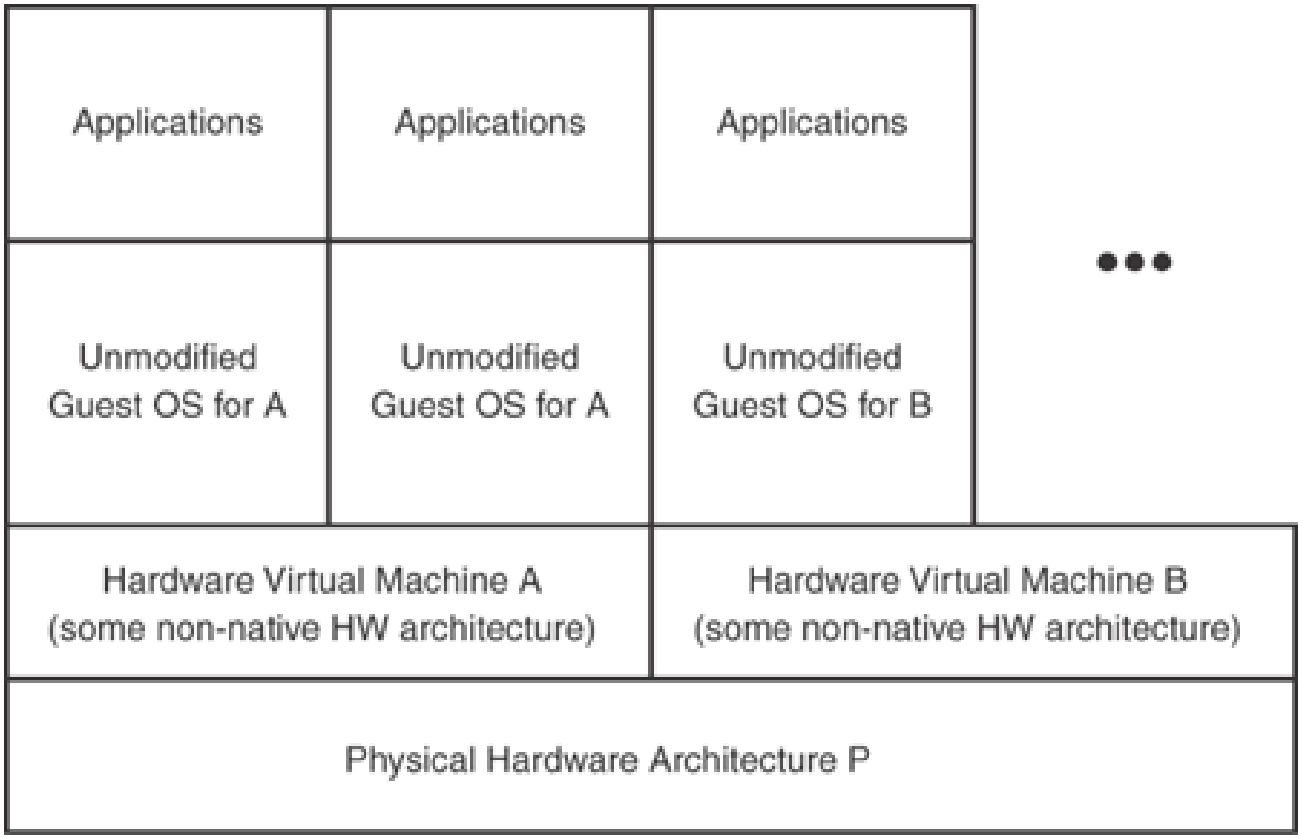
- The hypervisor abstracts the physical resources of the host computer into discrete virtual counterparts that can be allocated for use by individual guests
- Virtual guests treat their virtual hardware as if it were real, and the hypervisor ensures that this illusion is seamless
- Additionally, hypervisors must ensure some level of isolation between guests
- In a way, hypervisors act as both magician and traffic cop

- Virtual machine monitors also provide a uniform interface to the hardware
- This uniform interface shields guest systems from some lower level details of the physical computing resources and provides portability, which is another key benefit of virtualization
- In fact, many modern hypervisors allow guest systems to move from one physical machine to another without interruption
- Guest system configurations can easily be developed on one machine and then deployed on many systems
- This eases the job of managing or deploying software on a collection of machines with different hardware characteristics
- Guest systems can even migrate from one physical computer to another while running. Xen calls this live migration. Some benefits of virtualization are as follows

Emulator

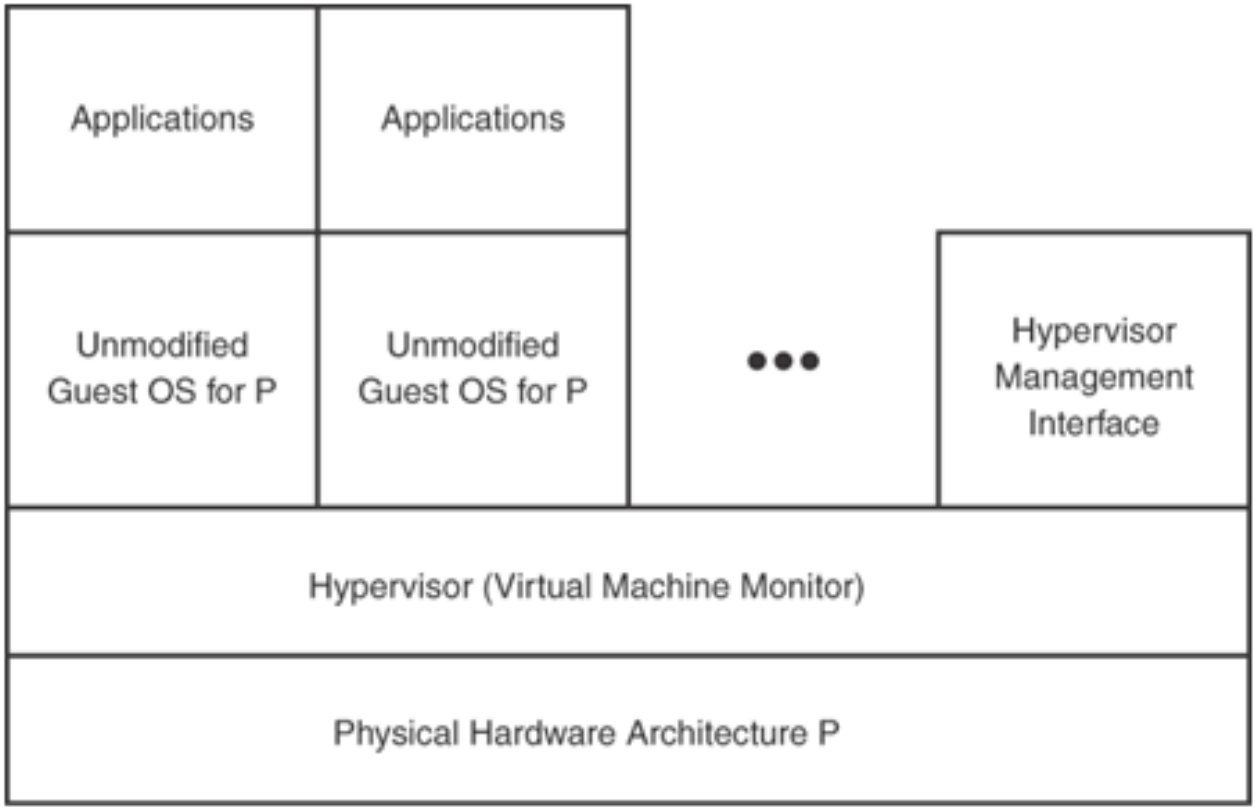
- Emulator virtual machines provide a virtual computing architecture that is not the same as the actual physical architecture of the host machine
- Operating systems intended for the emulated hardware are executed unmodified

- x86-32bit
  - power


Applications
Applications
Applications
Unmodified Guest OS for A
Unmodified Guest OS for A
Unmodified Guest OS for B
Hardware Virtual Machine A (some non-native HW architecture)
Hardware Virtual Machine B (some non-native HW architecture)
Physical Hardware Architecture P

Full Virtualization

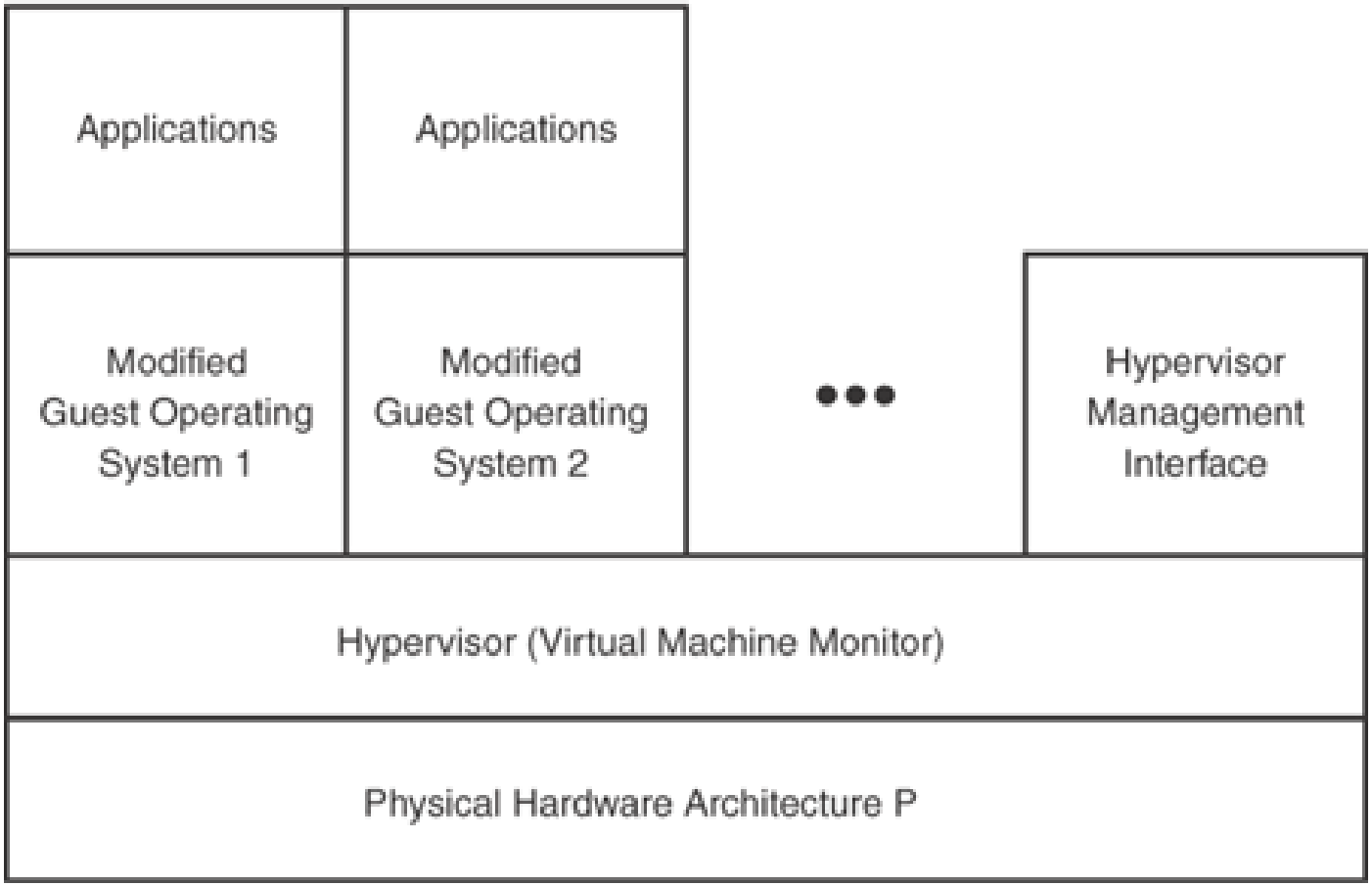
- The full virtualization hypervisor presents the actual physical hardware to each guest so that operating systems intended for the underlying architecture may run unmodified and unaware that they are being run virtualized


Applications
Applications
Unmodified Guest OS for P
Unmodified Guest OS for P
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture P

- The x86 architecture is notoriously difficult to virtualize
- Because of this, virtualization specifics (Intel's VT and AMD's AMD-V) have been added to improve performance and make running an operating system within a Xen virtual machine simpler
- Major vendors of full virtualization include VMware Workstation, VMware Server (formerly GSX Server), Parallels Desktop, Win4Lin Pro, and z/VM
- Xen supports full virtualization on basic architectures with the previously mentioned hardware support for virtualization

Paravirtualization

- In paravirtualization, the hypervisor exports a modified version of the underlying physical hardware
- The exported virtual machine is of the same architecture
- Targeted modifications are introduced to make it simpler and faster to support multiple guest operating systems
  - the guest operating system might be modified to use a special hypercall application binary interface (ABI) instead of using certain architectural features that would normally be used

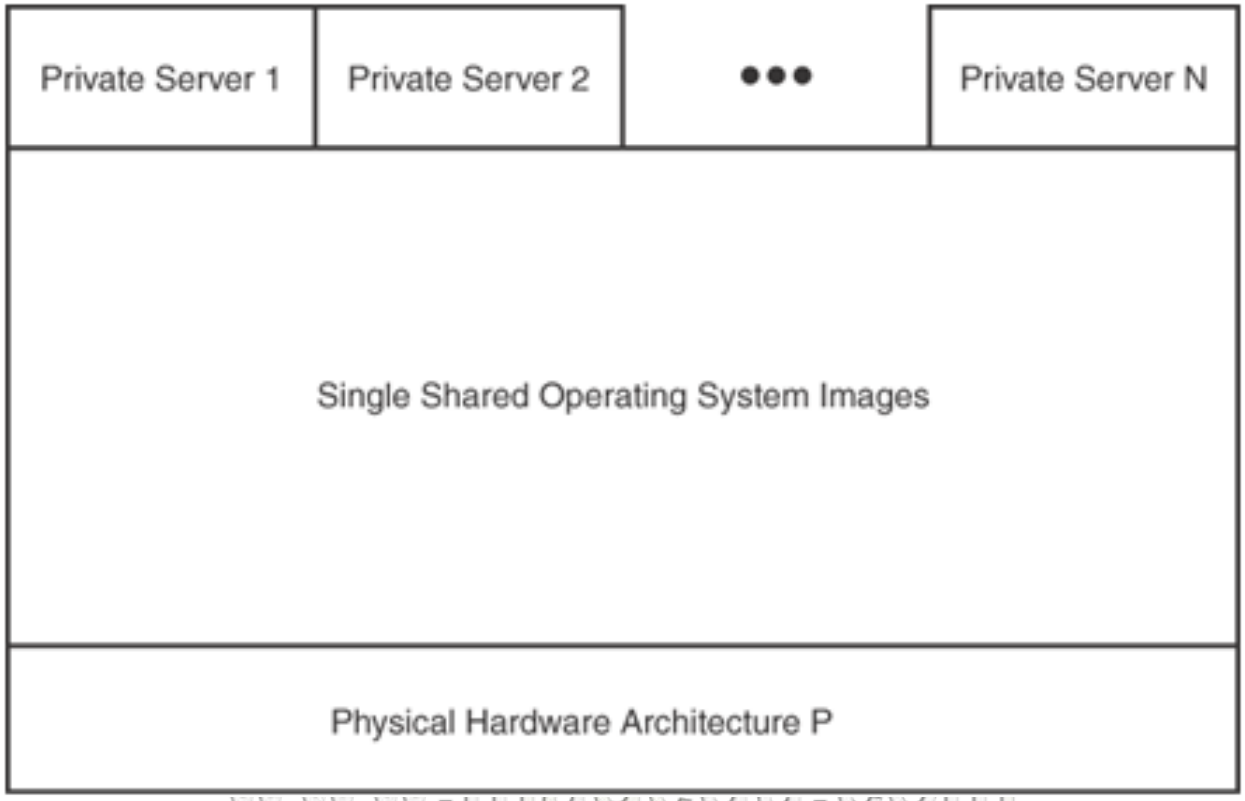

Applications
Applications
Modified Guest Operating System 1
Modified Guest Operating System 2
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture P

- KVM:
  - Kernel Virtual Machine

- Major advantages include performance, scalability, and manageability
- The two most common examples of this strategy are User-mode Linux (UML) and Xen
- The choice of paravirtualization for Xen has been shown to achieve high performance and strong isolation even on typical desktop hardware
- Xen extends this model to device I/O

Operating System Level Virtualization

- No virtual machine monitor
- The virtualization is done entirely within a traditional single operating system image
- Implementations of operating system level virtualization include Virtuozzo, Linux VServers, OpenVZ, Solaris Containers, FreeBSD jails, and HP UX 11i Secure Resource Partitions


Private Server 1
Private Server 2
●●●
Private Server N
Single Shared Operating System Images
Physical Hardware Architecture P

Other Types of Virtualization

- Library virtualization
  - emulates operating systems or subsystems via a special software library
  - An example: Wine
- Application virtualization (managed runtime)
  - the approach of running applications inside a virtual execution environment
  - The virtual execution environment provides a standard API for cross-platform execution and manages the application's consumption of local resources

Linux KVM (Kernel Virtual Machine)

- KVM is a full virtualization solution that was merged during the 2.6.20 mainline kernel development period
- KVM is a modification to the Linux kernel that actually makes Linux into a hypervisor upon insertion of an additional KVM module
- The method of KVM operation is rather interesting
  - Each guest running on KVM is actually executed in user space of the host system
  - This approach makes each guest instance (a given guest kernel and its associated guest user space) look like a normal process to the underlying host kernel

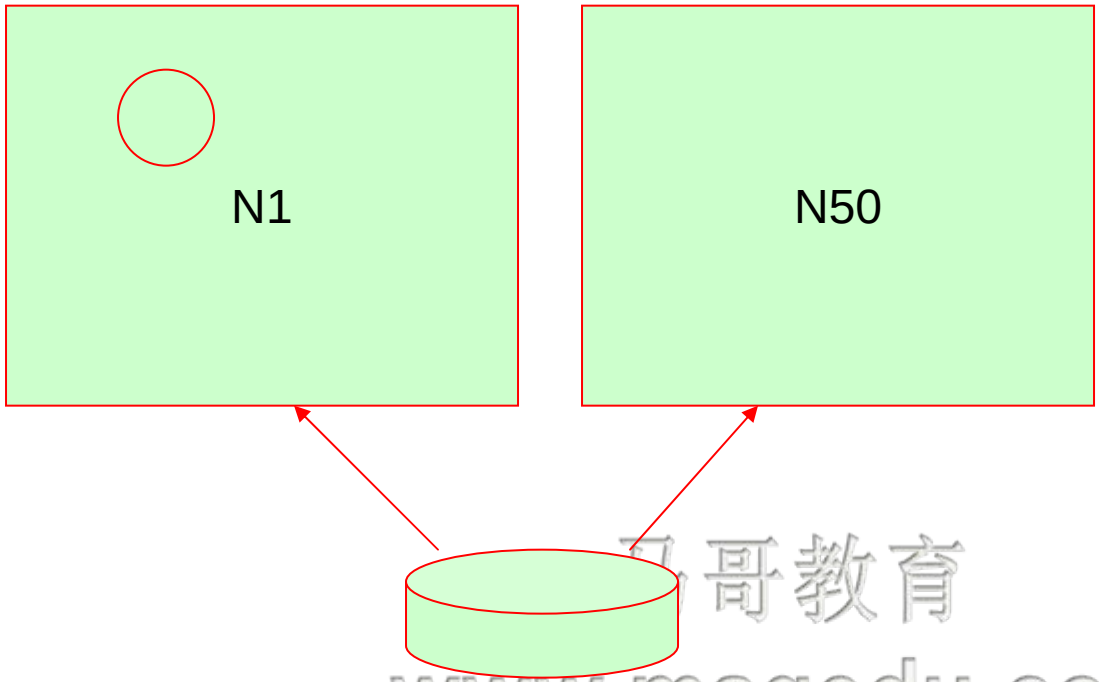
Virtualization Products at a Glance

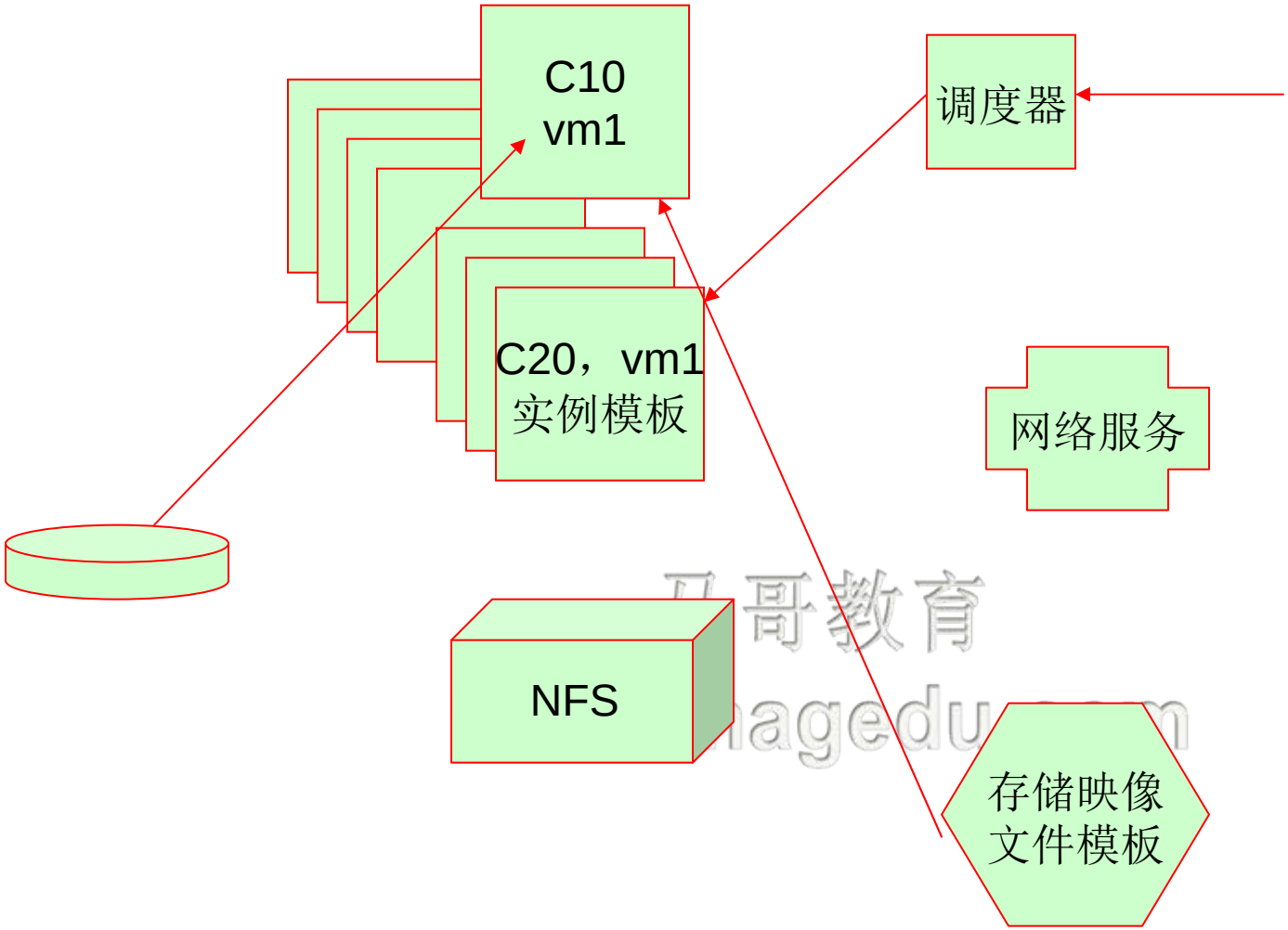
Implementation	Virtualization Type	Installation Type	License
Bochs	Emulation	Hosted	LGPL
QEMU	Emulation	Hosted	LGPL/GPL
VMware	Full Virtualization & Paravirtualization	Hosted and bare-metal	Proprietary
User Mode Linux (UML)	Paravirtualization	Hosted	GPL
Lguest	Paravirtualization	Bare-metal	GPL
Open VZ	OS Level	Bare-metal	GPL
Linux VServer	OS Level	Bare-metal	GPL
Xen	Paravirtualization or Full when using hardware extensions	Bare-metal	GPL
Parallels	Full Virtualization	Hosted	Proprietary
Microsoft	Full Virtualization	Hosted	Proprietary
z/VM	Full Virtualization	Hosted and bare-metal	Proprietary
KVM	Full Virtualization	Bare-metal	GPL
Solaris Containers	OS Level	Hosted	CDDL
BSD Jails	OS Level	Hosted	BSD
Mono	Applications level	Application Layer	Compiler and tools GPL, Runtime libraries LGPL, Class libraries MIT X11
Java Virtual Machine	Applications Level	Application Layer	GPL

马哥教育

主讲：马永亮(马哥)
QQ:113228115
客服QQ：2813150558，1661815153
http://www.magedu.com
http://mageedu.blog.51cto.com

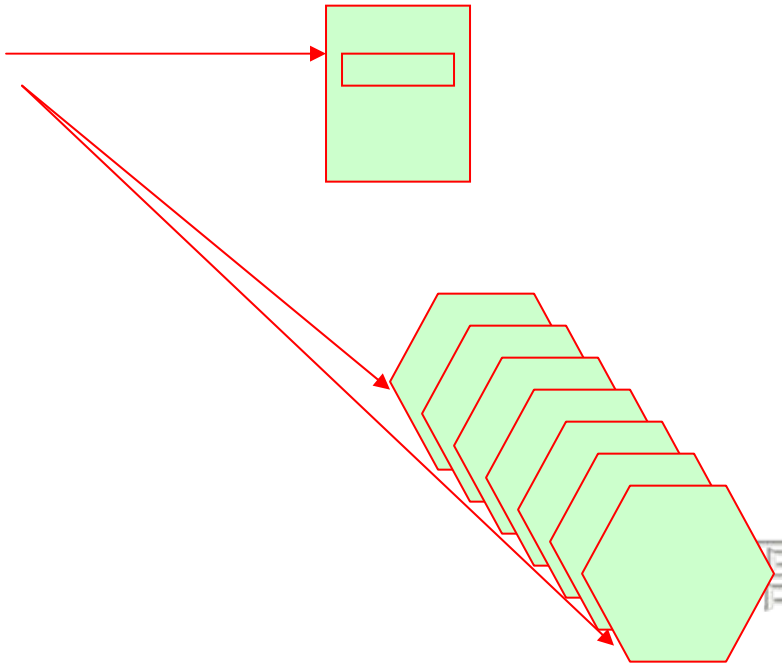
- /etc/xen/, /etc/libvirt/qemu/

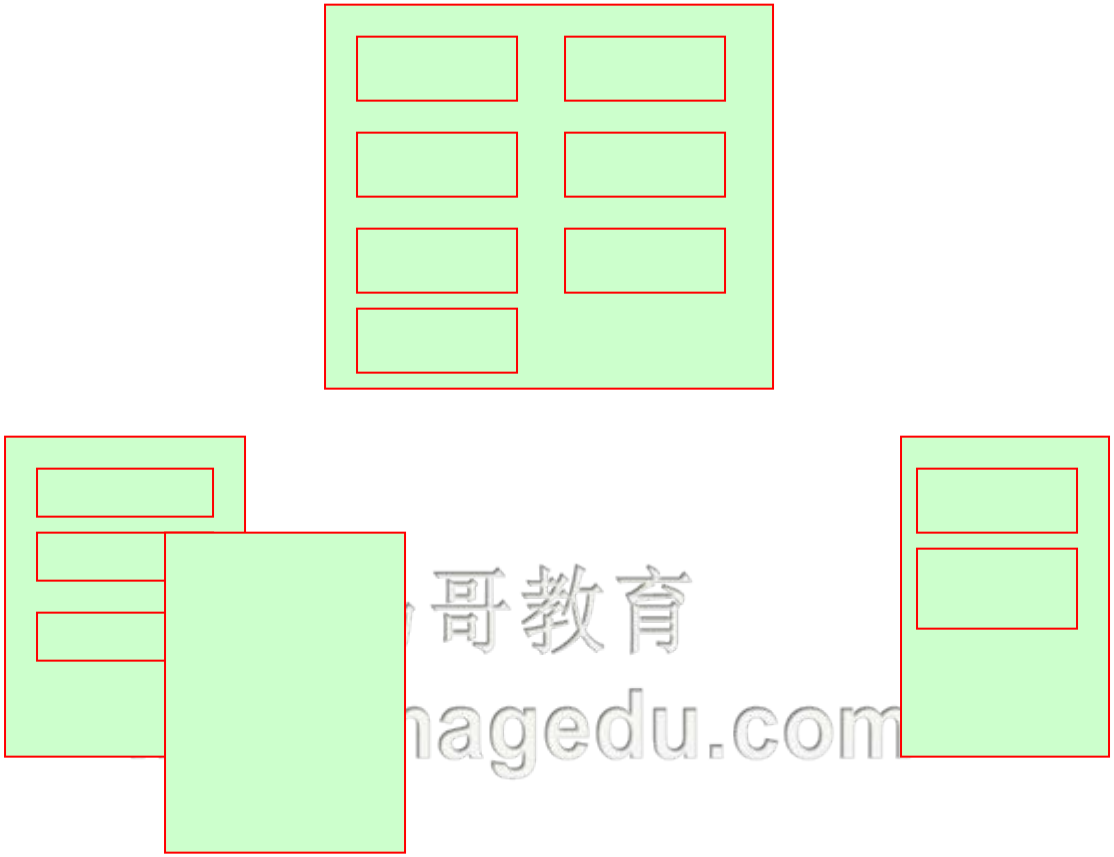


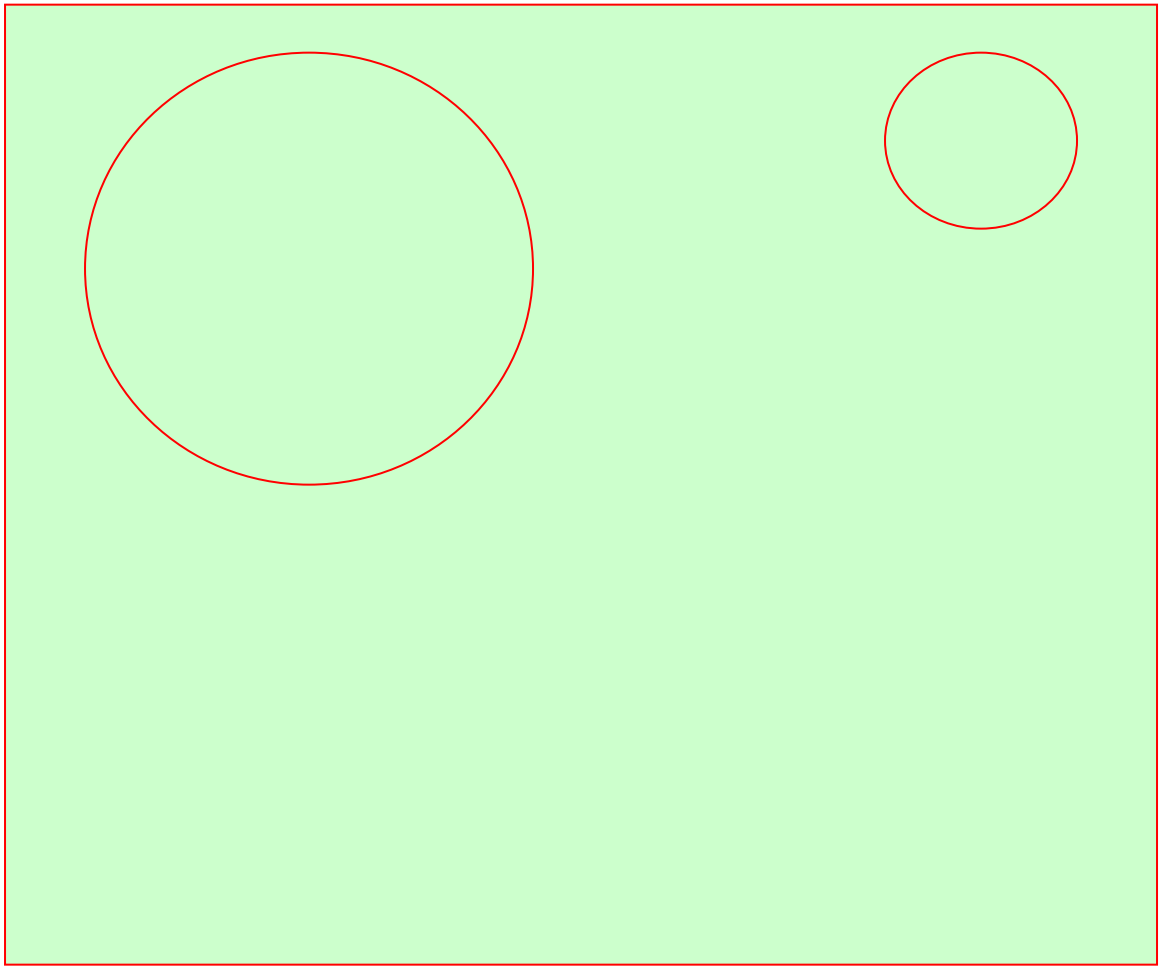


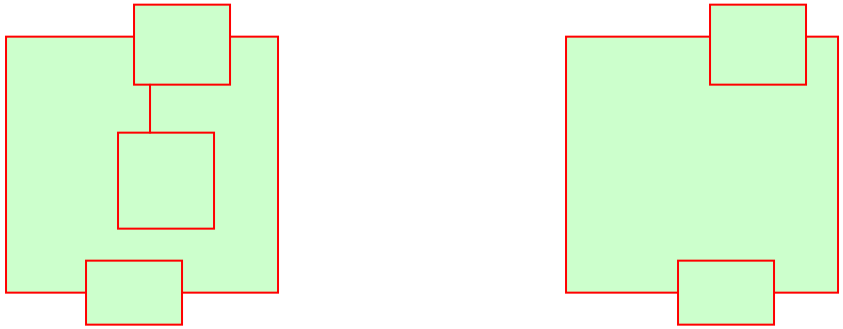
- Compute (Nova)
  - Nova-volume
  - Nova-network
  - Nova-scheduler
  - Nova-compute
- Storage (Image, Glance)
  - Glance: 映像文件元数据管理服务
- Identify (keystone)

- 分布式文件系统

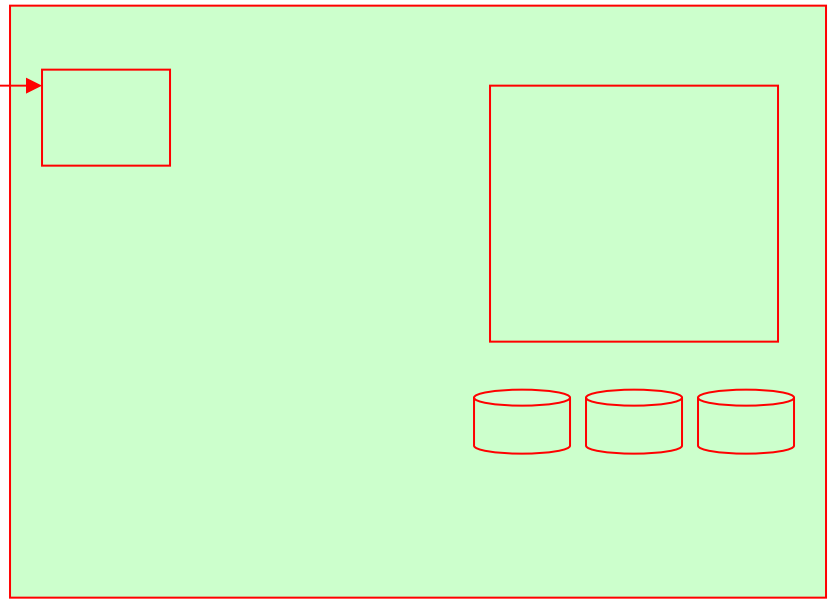


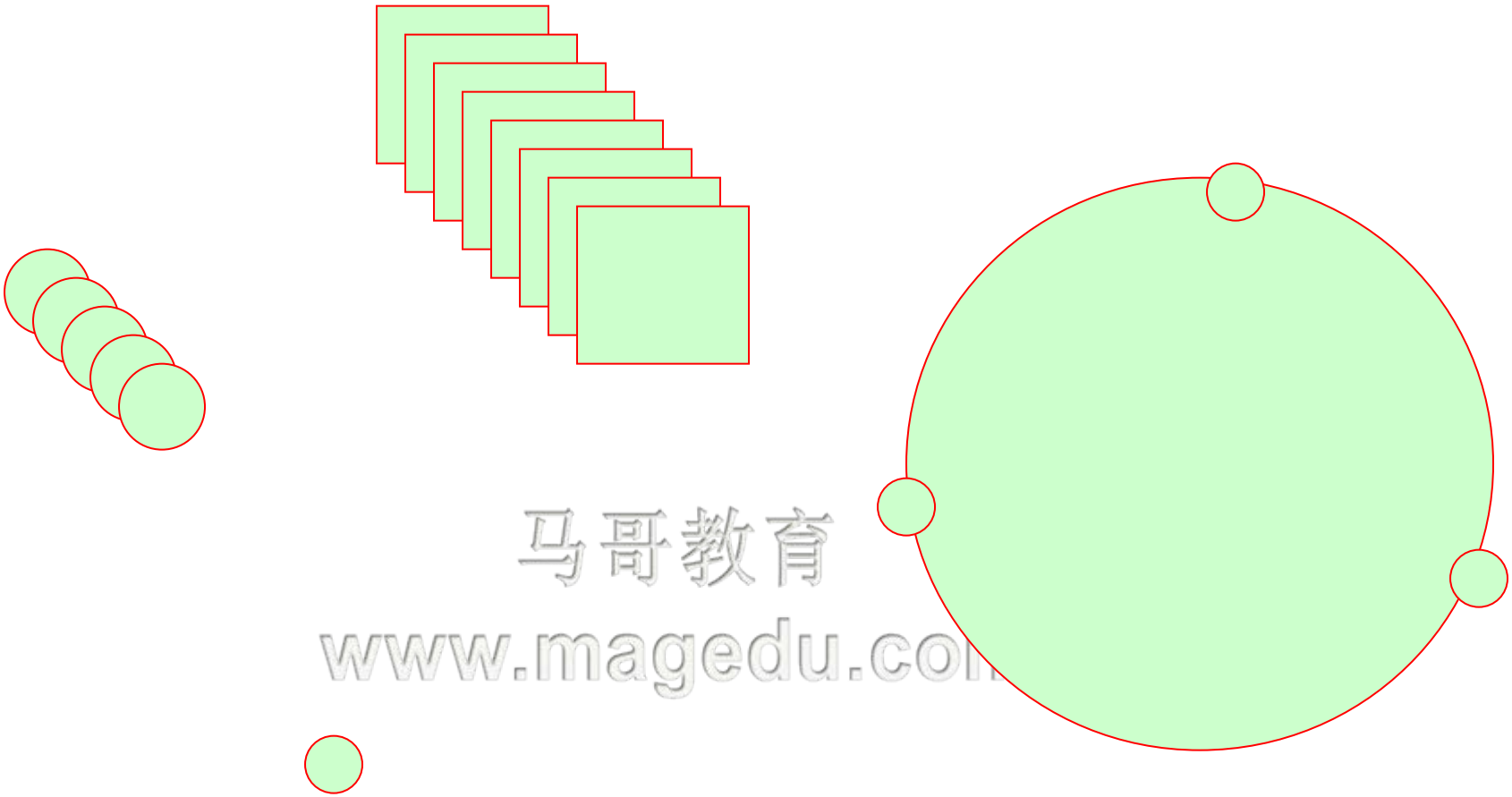






- Horizon
  - Swift
  - Essex





关于马哥教育

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：2813150558，1661815153，113228115
- QQ群：203585050，279599283

马哥教育

Thank You!