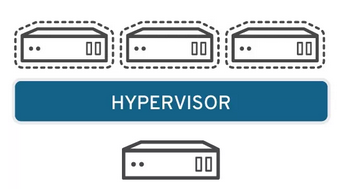
# 虚拟化技术

## 概述

虚拟化技术是将底层物理设备（CPU、内存、外设等）与上层操作系统、应用软件分离的一种解耦技术，在一台物理机上逻辑划分出多个虚拟机使用这些计算资源。

虚拟化的核心思想就是提高底层硬件资源的利用率，通过在操作系统中增加一个虚拟化层，将底层的资源池化，向上层操作系统提供一个预期一致的服务器硬件环境，并允许不同操作系统的虚拟机相互隔离并且运行在同一个物理机上。



KVM虚拟化主要包括CPU虚拟化、内存虚拟化、网络虚拟化、存储虚拟化。

其中，KVM虚拟机CPU的软件调休主要是借助NUMA技术；内存方面的调优手段主要是KSM，即相同的内存页合并、内存气球技术及大页内存的使用。这些都在后面专门详述。

## 历史

在虚拟化技术发展初始，主要应用于IBM的大型机中，随着X86服务器架构流行，VMware开发出第一个针对X86服务器的商用虚拟化技术。后来，剑桥大学一名讲师发布了开源虚拟化项目XEN。

## 分类

虚拟化类型一般分为两类：I型虚拟化和II型虚拟化。

### I型虚拟化

Hypervisor直接安装在物理机上，多个虚拟机在Hypervisor上运行。Hypervisor实现方式一般是一个**特殊定制的Linux系统**。

Xen和VMWare的ESXi都属于这个类型。

### II虚拟化

物理机上首先安装**常规的操作系统**，比如RedHat、Ubuntu和Windows。Hypervisor作为OS上的一个**程序模块**运行，并对管理虚拟机进行管理。

KVM、VirtualBox和VMWare Workstation都属于这个类型。

目前虚拟化技术，大致包括：软件模拟、虚拟化层翻译、容器虚拟化。

### 软件模拟

软件模拟是通过软件完全模拟CPU、内存、网卡、磁盘等计算机硬件设备，其优点在于可以模拟任何硬件设备，但是其效率非常低，不具备实际应用价值。

采用该技术的产品有QEMU。

### 虚拟化层翻译

Hypervisor是一种运行在物理服务器和操作系统之间的中间软件层，可允许多个操作系统和应用共享一套基础物理硬件，因此也可以看作是虚拟环境中的“元”操作系统，它可以协调访问服务器上的所有物理设备和虚拟机，也叫虚拟机监视器（Virtual Machine Monitor）。Hypervisor是所有虚拟化技术的核心，非中断地支持多工作负载迁移的能力是Hypervisor的基本功能。当服务器启动并执行Hypervisor时，它会给每一台虚拟机分配适量的内存、CPU、网络和磁盘，并加载所有虚拟机的客户操作系统。

该技术又分为三种类型：

1. 全虚拟化：通过虚拟化引擎VMM，捕获虚拟机的指令并处理。

说明：这里的处理不是完完整整的将虚拟机的指令传递给宿主机，因为虚拟机传递的是越级的指令，所以VMM需要做些额外处理工作，比如虚拟机执行reboot，宿主机不会重启。

采用该技术的产品有VMWare ESXi。

1. 半虚拟化：全虚拟化方式由于在虚拟化层做了大量的“翻译”工作，效率

较低，所以XEN提出将虚拟化层的部分功能剥离，放到虚拟机内部处理，这就要求对虚拟机内核进行改造。

采用该技术的产品有XEN、Hyper-V。

XEN包含三个基本组件：Hypervisor、Domain 0、Domain U。

其中Hypervisor运行在硬件之上，承载所有的操作系统，提供CPU和内存调度的作用。

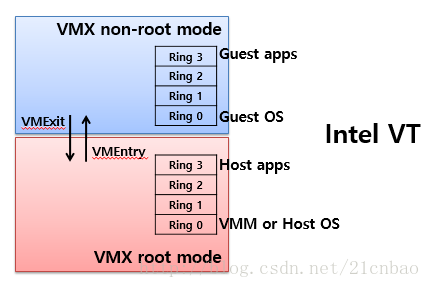
Domain 0是管理员，相当于一个大管家，主要负责IO虚拟化。

Domain U则是真正的XEN中的虚拟客户机，而Domain U又分为两类：一类全虚拟客户机，一类半虚拟客户机，后者是运行在经过修改的操作系统之上。

1. 硬件支持的全虚拟化：前面的两种虚拟化方式都是在软件层面上进行的，

后来芯片厂商Intel推出了硬件解决方案，对CPU指令进行改造，即VT-x（AMD的硬件解决方案为AMD-v），其效率非常高。

VT-x增加了两种操作模式：VMX root模式和VMX non-root 模式，虚拟化引擎VMM运行在VMX root模式，虚拟机运行在VMX non-root模式，这两种操作模式都支持ring0~ring4这4个特权级。



对于虚拟机而言，Guest OS运行在non-root模式的ring0，其上的应用程序运行在non-root模式的ring3；而对于物理机而言，host OS和VMM则运行于root模式下的ring0，物理机的应用程序运行在root模式的ring3。Guest OS下发的越级指令，会VMExit到root模式下ring0内的VMM，从root环境到non-root环境叫做VMEntry，实际使用时在二者之间不断切换。

采用该技术的产品有KVM。

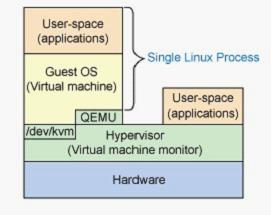
### 容器虚拟化

容器虚拟化基于CGroup和Namespace技术实现进程隔离的，每个进程有自己隔离的资源、根目录、进程号、内存空间等。但是，Docker在网络、磁盘上还受限制。

# KVM概述

KVM全称是Kernel-Based Virtual Machine，也就是说KVM是基于Linux内核实现的。

## KVM架构



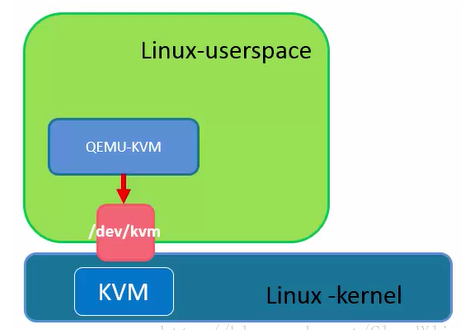
Guest：客户机系统，包括CPU（vCPU）、内存（vMem）、驱动（网卡、I/O 设备驱动等），被 KVM 置于一种受限制的CPU（即non-root）模式下运行。

KVM：运行在内核空间，提供CPU和内存的虚级化，以及客户机的I/O拦截。Guest的I/O被KVM拦截后，交给QEMU处理。

QEMU：修改过的为KVM虚机使用的QEMU代码，运行在用户空间，提供硬件I/O虚拟化，通过ioctl /dev/kvm设备和KVM交互。

## QEMU与KVM

QEMU是一个硬件模拟器，内建HDCP、DNS、TFTP等服务器，但是其速度慢实用价值不大。而KVM只是内核的一个模块，它没有用户空间的管理工具，KVM的虚拟机可以借助QEMU的管理工具管理，QEMU也可以借助KVM实现加速。现在，KVM与QEMU已经结合在一起。



/dev/kvm是KVM内核模块提供给用户空间的一个接口，这个接口被qemu-kvm调用，通过ioctl系统调用就可以给用户提供一个工具用于管理虚拟机生命周期。qemu-kvm就是通过open()、close()、ioctl()等方法打开、关闭、调用这个接口实现与KVM交互的。

## libvirt与KVM

libvirt是Redhat开源的虚拟化管理工具，除了能够管理KVM这种Hypervisor，还可以管理Xen、VirtualBox等。

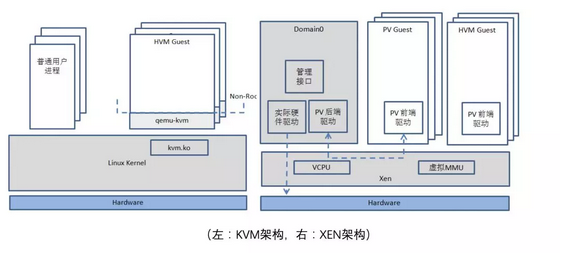
主要有3部分组成：

1. lib库：支持C、Python、Ruby等编程语言；
2. libvirtd服务
3. virsh命令行工具

libvirt可以管理不同的虚拟化引擎，但是目前实际应用环境中只有KVM在使用。可以实现虚拟机的生命周期管理以及外设的添加。

## KVM与XEN

KVM较XEN出现的晚，但是由于XEN需要对内核补丁，稍微不慎，系统崩溃，加之KVM是Linux的一部分，可以使用通常的Linux调度器和内存管理，这使得KVM更小更易使用。

 技术对比：





大多数XEN和KVM性能基准测试表明，XEN比KVM拥有更好的处理性能。

# 环境搭建

## CPU检测

egrep -c '(vmx|svm)' /proc/cpuinfo

输出结果大于0表示支持虚拟化，如果等于0则需要手动选择支持虚拟化。

## 安装KVM

yum install qemu-kvm

安装虚拟化管理工具：

yum install virt-manager libvirt libvirt-python python-virtinst bridge-utils

## 加载kvm.ko

modprobe kvm

查看驱动是否加载成功：

lsmod | grep kvm

如果加载成功会有输出信息，没有加载成功grep不到指定驱动信息。

## 启动libvirt

service libvirtd restart

chkconfig libvirtd on

## 创建系统盘

qemu-img create -f qcow2 /kvmtest/centos-7.qcow2 10G

## 启动虚拟机

virt-install --virt-type kvm --name centos-7 --ram 1024 \

--vcpus 1 \

--cdrom=/kvmtest/CentOS-7-x86\_64-minimal.iso \

--disk /kvmtest/centos-6.8.qcow2,format=qcow2 \

--network network=default \

--graphics vnc,listen=0.0.0.0 --noautoconsole \

--os-type=linux --os-variant=rhel6

各参数的含义如下：

-n NAME 指定虚拟机的名称

-r MEMORY 指定虚拟机的内存用量

-u UUID 指定虚拟机的唯一通用标识符（UUID）。

省略这个参数时，virt-install 将会自动产生

–vcpus=VCPUS 指定虚拟机的虚拟 CPU （Virtual CPU，VCPU） 数量

-f DISKFILE 指定虚拟磁盘的文件路径名称

-s DISKSIZE 用来指定虚拟磁盘的大小，这个参数需配合-f 使用。

DISKSIZE为虚拟磁盘的大小，单位是GB

-m MAC 指定虚拟机的网络卡之硬件地址。

这个参数可以省略，省略时virt-install 将自动产生

-p 以半虚拟化的方式建立虚拟机

-l LOCATION 指定安装来源

## VNC连接虚拟机

netstat -natp

# CPU虚拟化

## 概述

从计算层面来说，X86物理服务器上的CPU、内存资源都可提供给虚拟机使用。现在的高性能X86服务器一般都是多CPU多核系统，NUMA架构会越来越受欢迎，因为这个架构可解决多处理器多核和非统一内存架构之间的交互带来的新的CPU、内存资源分配方法的挑战，并改善占用大量内存的工作负载的性能。

通俗来讲，CPU虚拟化就是用单个CPU模拟出多个并行的CPU，允许一个平台同时运行多个操作系统，并且应用程序都可以在相互独立的空间内运行而互不影响，从而显著提高计算机的工作效率。

CPU的虚拟化技术是一种硬件方案，原因是纯软件虚拟化解决方案存在很多限制 。“客户”操作系统很多情况下是通过VMM(Virtual Machine Monitor，虚拟机监视器)来与硬件进行通信，由VMM来决定其对系统上所有虚拟机的访问。(注意，大多数处理器和内存访问独立于VMM，只在发生特定事件时才会涉及VMM，如页面错误)。在纯软件虚拟化解决方案中，VMM在软件套件中的位置是传统意义上操作系统所处的位置。如处理器、内存、存储、显卡和网卡等)的接口，模拟硬件环境。这种转换必然会增加系统的复杂性。支持虚拟技术的CPU带有特别优化过的指令集来控制虚拟过程，通过这些指令集，VMM会很容易提高性能，相比软件的虚拟实现方式会很大程度上提高性能。

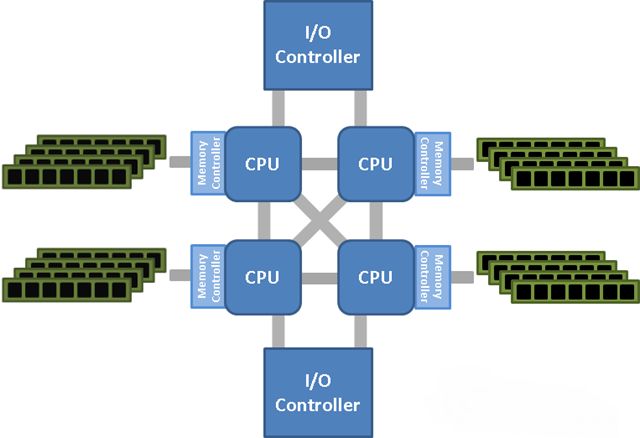
虚拟化技术可提供基于芯片的功能，借助兼容VMM软件能够改进纯软件解决方案。由于虚拟化硬件可提供全新的架构，支持操作系统直接在上面运行，从而无需进行二进制转换，减少了相关的性能开销，极大简化了VMM设计，进而使VMM能够按通用标准进行编写，性能更加强大。另外，在纯软件VMM中，目前缺少对64位客户操作系统的支持，而随着64位处理器的不断普及，这一严重缺点也日益突出。而CPU的虚拟化技术除支持广泛的传统操作系统之外，还支持64位客户操作系统。

虚拟化技术是一套解决方案。完整的情况需要CPU、主板芯片组、BIOS和软件的支持，例如VMM软件或者某些操作系统本身。即使只是CPU支持虚拟化技术，在配合VMM的软件情况下，也会比完全不支持虚拟化技术的系统有更好的性能。

## 硬件虚拟化

Intel处理器产品线中应用英特尔Virtualization Technology(Intel VT)虚拟化技术，而AMD支持AMD Virtualization Technology(AMD VT)虚拟化技术。

## NUMA



传统的服务器架构下是把内存放到单一的存储池中，这对于单处理器或单核心的系统工作良好。但是这种传统的统一访问方式，在多核心同时访问内存空间时会导致资源争用和性能问题。而NUMA是服务器CPU和内存设计的新架构，它改变了内存对CPU的呈现方式，这是通过对服务器每个CPU的内存进行分区来实现的。每个分区（或内存块）称为NUMA节点，而和该分区相关的处理器可以更快地访问NUMA内存，而且不需要和其它的NUMA节点争用服务器上的资源（其它的内存分区分配给其它处理器）。NUMA也支持任意一个处理器访问服务器上的任何一块内存区域，某个处理器当然可以访问位于不同区域上的内存数据，但是却需要更多本地NUMA节点之外的传输，并且需要目标NUMA节点的确认。这增加了整体开销，影响了CPU和内存子系统的性能。

例如一台配置了两颗八核处理器以及128GB内存的服务器.在NUMA架构中，每个处理器能够控制64GB的物理内存，每个处理器的八个核心中的每个核心将对应一个8GB的NUMA节点.这将会如何影响虚拟机性能？由于每个处理器核心访问NUMA节点内内存的速度要比其他节点快，因此当虚拟机内存大小少于或者等于NUMA节点的内存大小时，虚拟机在理论上能够获得最好的性能。所以我们在这台物理服务器上分配虚机时，不要给每台虚拟机分配超过8GB的内存，如果给虚拟机分配更多的内存，则虚拟机必然要访问其NUMA节点之外的部分内存，这样或多或少会影响其性能.如果应用能够感知NUMA，那就更好了。vSphere使用vNUMA可以创建能够感知NUMA的虚拟机，该虚拟机将会被分割为虚拟NUMA节点，每个vNUMA节点将会被放置到一个不同的物理NUMA节点。尽管虚拟机仍旧在两个NUMA节点之间扩展，但虚拟机内的操作系统和应用能够感知NUMA，资源使用将会得到优化。

NUMA已经对在数据中心服务器上安装及选择内存的方式带来了很多改变。在给服务器增加物理内存时，我们需要注意增加的内存要在NUMA节点之间进行平衡及匹配以使主板上的每个处理器拥有相同的内存。如果在我们所举例的服务器上配置更多的内存，那么必须在处理器之间平衡这些内存模块，如果增加64GB的内存，那么每个处理器将分配到32GB的内存（每个处理器可支配的内存将增加到96GB，服务器总内存数将达到192GB），每个NUMA节点的内存大小将从8GB增加到12GB。

结合VMWARE的最佳实践，VMware一般给CPU建议，最大支持64个vCPU，一般不超过32个，最好不要超配置；内存一般不给建议，根据不同的业务对内存大小会有不同的要求，当然最好不要跨NUMA单元去进行调用。

另外还要注意一点NUMA架构只针对物理CPU(Socket)，不针对核(Core)。由于每个Socket控制的内存插槽是不同的，因此要确保内存插槽是均匀的.例如128G内存分为8个16G的内存条，那么应该4个插在一个Socket的内存插槽中，另4个插在另一个socket的内存插槽中.在为虚机分配vCPU资源时，也尽可能按照Socket/Core的倍数分配，比如1X1, 1X2, 1X 4, 1X8, 2X1, 2X2, 2X4, 2X8等组合，但不要使用2X3, 2X5, 2X7这种组合。后面的组合会引起跨Socket的内存调用，从而容易导致性能下降。

# 内存虚拟化

## 概述

通过内存虚拟化共享物理系统内存，动态分配给虚拟机。现在所有的X86 CPU都包括了一个称为内存管理的模块MMU（Memory Manager Unit）和TLB（Translation Lookaside Buffer），通过MMU和TLB来优化虚拟内存的性能。

KVM 中，虚机的物理内存即为 qemu-kvm 进程所占用的内存空间。KVM 使用 CPU 辅助的内存虚拟化方式。在 Intel 和 AMD 平台，其内存虚拟化的实现方式分别为：

AMD 平台上的 NPT （Nested Page Tables） 技术

Intel 平台上的 EPT （Extended Page Tables）技术

EPT 和 NPT采用类似的原理，都是作为 CPU 中新的一层，用来将客户机的物理地址翻译为主机的物理地址。

## KSM

## 内存气球

## 内存限制

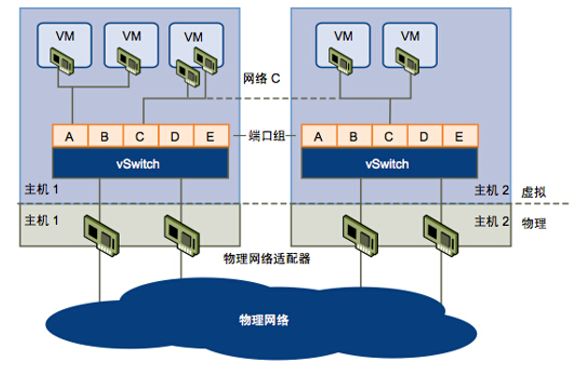
## 巨页

# 服务器虚拟化

# 网络虚拟化

## 概述

从网络层面来说，X86物理机使用的是物理的网卡，连接的是物理的交换机。在一台X86被划分成多个VM虚机后，就诞生了虚拟网卡和虚拟交换机，这样在虚拟和物理网络之间就产生了流量传输与交互。



同一台物理机上划分的VM有同一网段和不同网段之分，再根据虚机之间的网络流量是否经过物理网卡，具体可分为四种不同的情况：

第一种情况，比如某业务系统的VM在同一主机的同一网段，则虚拟机之间网络流量不经过主机物理网卡，其最大网络流量为7.6GB。（测试方法：在testvm1上启用jperf server作为网络数据接收端,在testvm2中启用jperf client连接jperf server发送网络数据包，同时加压网络流量.X86主机为双万兆网卡）

第二种情况，比如某业务系统的VM在同一主机的不同网段，则虚拟机之间网络流量经过主机物理网卡，其最大网络流量为5.6GB。测试方法同上。

第三种情况，比如某业务系统的VM在不同主机的同一网段，则虚拟机之间网络流量经过主机物理网卡，其最大网络流量为6.5GB。测试方法同上。

第四种情况，比如某业务系统的VM在不同主机的不同网段，则虚拟机之间网络流量经过主机物理网卡，其最大网络流量为4.6GB.测试方法同上。

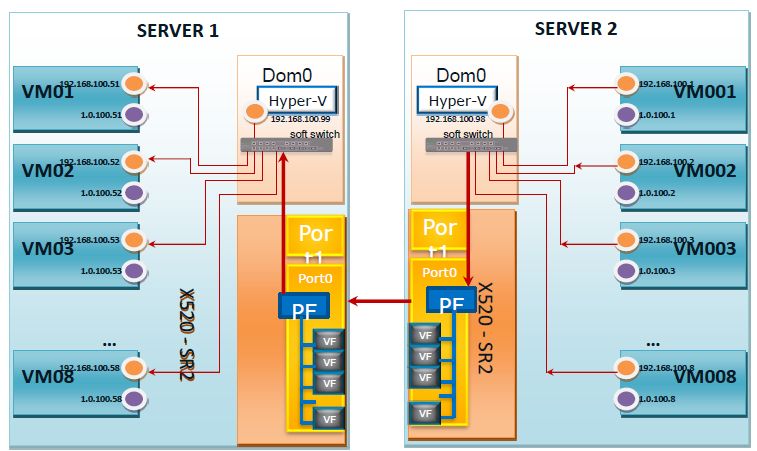
测试的几种场景对比表：



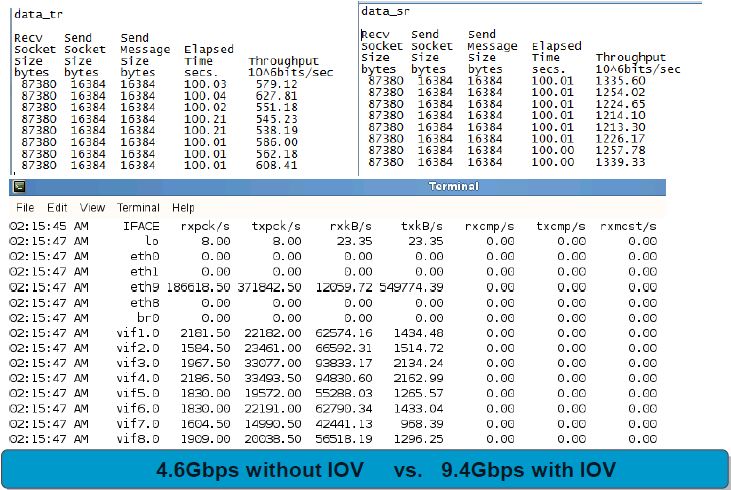
## SR-IOV

在一台X86物理服务器上进行VM的网络虚拟化，还有一种技术就是SR-IOV.SR-IOV 技术是INTEL提出的一种基于硬件的虚拟化解决方案，可提高性能和可伸缩性。SR-IOV 标准允许在虚拟机之间高效共享 PCIE（Peripheral Component Interconnect Express，快速外设组件互连）设备，并且它是在硬件中实现的，可以获得能够与本机性能媲美的网络 I/O 性能.比如我们把一台X86物理服务器上的万兆网卡通过SR-IOV技术分成4个虚拟网卡给4个VM使用，那么其网络传输性能将比虚拟化网卡给VM使用高很多。

测试方法：在一台X86物理服务器上，4个VM启用jperf server作为网络数据接收端；在另一台X86物理服务器上，4个VM启用jperf client连接jperf server发送网络数据包，同时加压网络流量.两台X86主机为双万兆网卡。



网络传输的数据量对比：



普通的虚拟化传输的最大数据量为4.6Gbps，而SR-IOV的直接硬件虚拟化可达到9.4Gbps。

具有 SR-IOV 功能的设备还有以下优点：节能、减少了适配器数量，同时简化了布线，减少了交换机端口。SR-IOV有很多优点，但是也有很多限制，比如VMWARE里的原有的很多功能将对SR-IOV 的虚拟机不可用，比如Vmotion、Storage Vmotion、Vshield、NetFlow、High Availability、FT、DRS、DPM、挂起和恢复、快照、热添加和删除虚拟设备、加入到群集环境.

因此，我们在考虑x86网络虚拟化的时候，更多的需要结合性能、业务特点、基础设施来综合考虑。如果一个业务要求比较高的性能同时不需要更多的灵活性，可以考虑SR-IOV技术，反之则选择X86常见的网络虚拟化技术，结合VMWARE来进行部署。

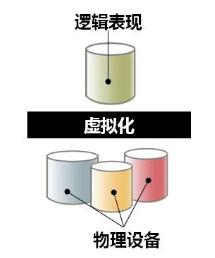
# 存储虚拟化

## 概述

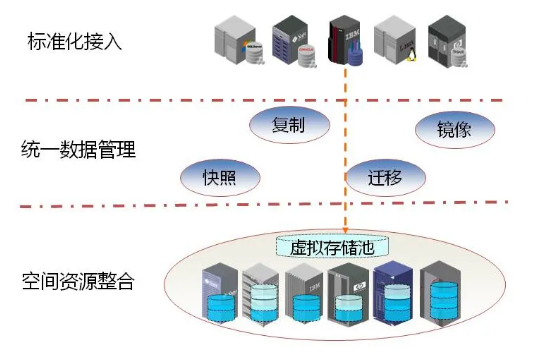
所谓虚拟存储，就是把多个存储介质模块（如硬盘、RAID）通过一定的手段集中管理起来，所有的存储模块在一个存储池（Storage Pool）中得到统一管理，从主机和工作站的角度，看到就不是多个硬盘，而是一个分区或者卷，就好象是一个超大容量（如1T以上）的硬盘。这种可以将多种、多个存储设备统一管理起来，为使用者提供大容量、高数据传输性能的存储系统，就称之为虚拟存储。

存储虚拟化是在物理存储系统和服务器之间增加一个虚拟层，它管理和控制所有存储并对服务器提供存储服务。

服务器不直接与存储硬件打交道，存储硬件的增减、调换、分拆、合并对服务器层完全透明。



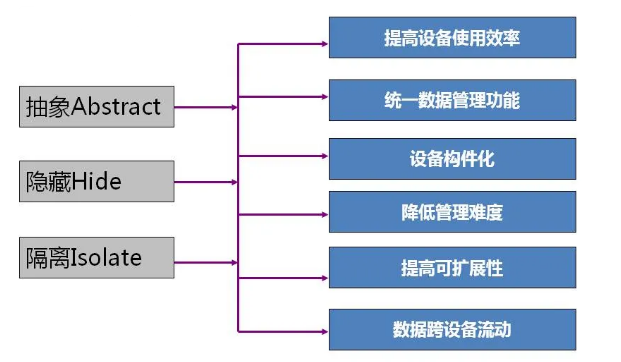
**存储虚拟化的原动力：**



**存储虚拟化的优点：**

1. 隐藏了复杂程度
2. 允许将现有的功能集成使用
3. 摆脱了物理容量的局限

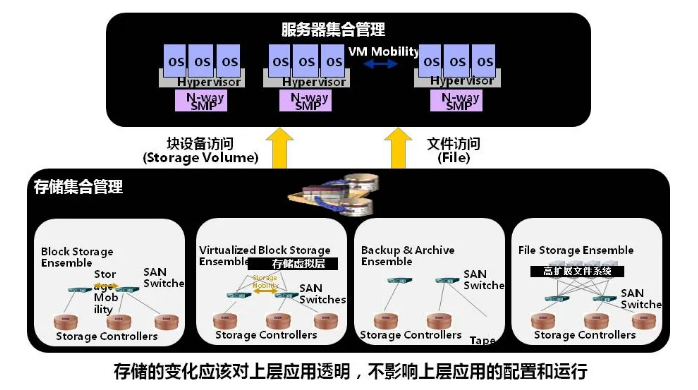
**存储虚拟化目的：**



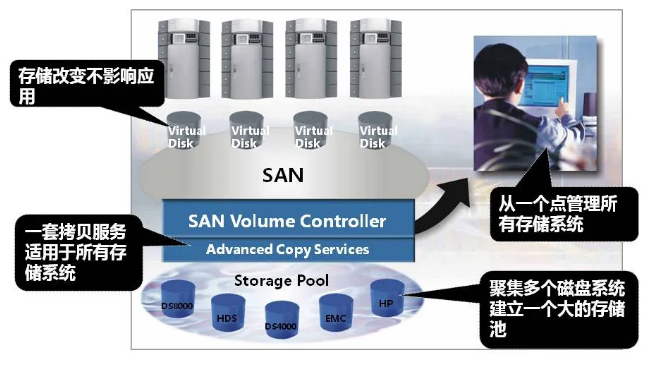
QEMU作为磁盘设备的软件模拟，后来结合KVM技术，加上Virtio半虚拟化技术的支持，性能得到很大改善。

KVM虚拟机可以指定IDE、SATA、Virtio、Virtio-SCSI几种磁盘格式，CentOS6.x只支持IDE和VIrtio磁盘类型，CentOS7.x增加支持SATA和Virtio-SCSI磁盘类型。IDE和SATA是全虚拟化的磁盘，性能较低；Virtio和Virtio-SCSI是半虚拟化磁盘，通过改造虚拟机的驱动达到目标。

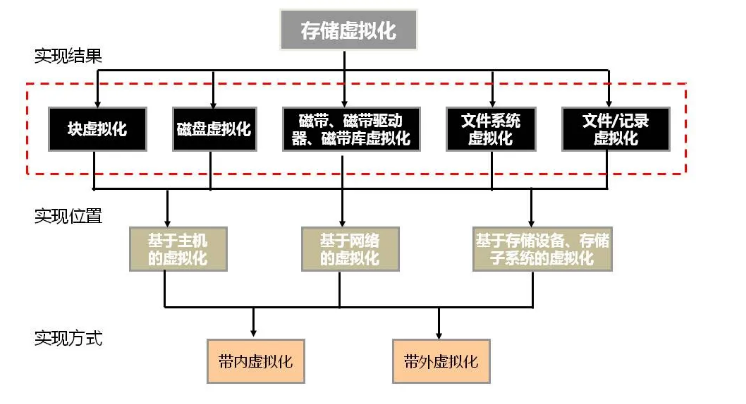
## 架构



异构环境构建存储虚拟化：

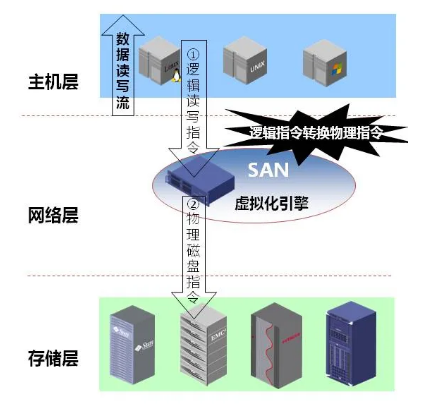


## 实现



### 实现方式

#### 带内虚拟化



用途：

1. 异构存储系统整合
2. 统一数据管理，在业务运行同时完成复制、镜像、CDP等各种数据管理功能

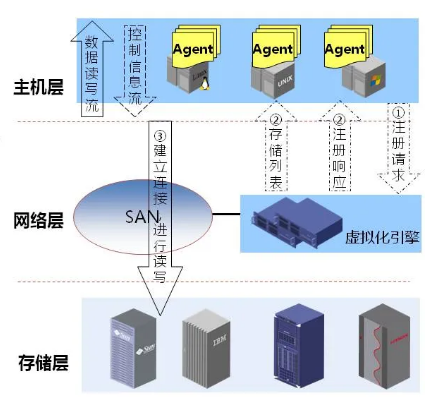
优点：

1. 服务器、存储设备的兼容性好
2. 虚拟化和数据管理功能由专用硬件实现，不占用主机资源
3. 丰富的数据管理功能
4. 配置简单，易于实施

缺点：

1、虚拟化设备发生故障，整个系统将中断

#### 带外虚拟化



用途：一般用于不同存储设备之间的数据复制

优点：

1. 虚拟化设备发生故障，整个系统将不会中断

缺点：

1. 主机资源占用较大
2. 大部分产品缺乏数据管理功能
3. 主机和存储系统需要严格的兼容性认证
4. 数据初始化同步复杂
5. 配置复杂，实施难度高

### 实现位置

#### 基于主机虚拟化

#### 基于网络虚拟化



主要用途：异构存储系统整合和统一数据管理

实现方式：通过在存储域网（SAN）中添加虚拟化引擎实现

优点：

1. 与主机无关，不占用主机资源
2. 能够支持异构主机，异构存储设备
3. 使不同存储设备的数据管理功能统一
4. 构建统一管理平台，可拓展性好

缺点：

1. 部分厂商数据管理功能弱，难以达到虚拟化统一数据管理的目睹
2. 部分厂商产品成熟度较低，仍然存在和不同存储和主机的兼容性问题

常见产品：

1. H3C IV系列
2. IBM SVC
3. EMC VPLEX
4. NetApp GFiler系列

#### 基于存储设备/存储子系统虚拟化



主要用途：在同一存储设备内部，进行数据保护和数据迁移

实现方式：在存储控制器上添加虚拟化功能，常见于中高端存储设备

优点：

1. 与主机无关，不占用主机资源
2. 数据管理功能丰富

缺点：

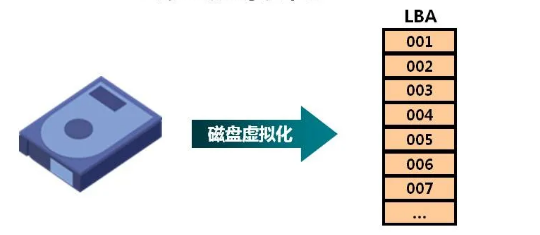
1. 一般只能实现对本设备内磁盘的虚拟化
2. 不同厂商间的数据管理功能不能互操作
3. 多套存储设备需配置多套数据管理软件，成本较高

常见产品（老存储产品）：

1. H3C IX/EX系列
2. HP EVA
3. HDS TagmaStor
4. SUN 6920系列

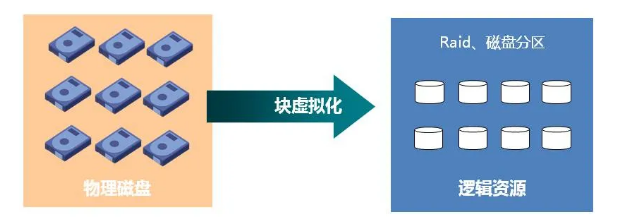
### 实现结果

#### 磁盘虚拟化



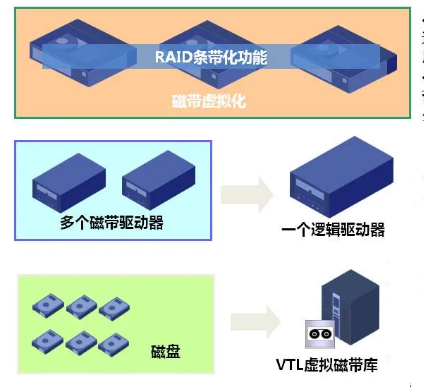
1. 磁盘扇区的物理地址一般用C-H-S（柱面号-磁头号-扇区号）表示
2. 磁盘虚拟化就是将扇区地址用LBA（逻辑块地址）表示，屏蔽底层物理磁盘的概念
3. 磁盘虚拟化由磁盘自身固件完成
4. 磁盘虚拟化的结果是使磁盘的使用者无须了解磁盘的内部硬件细节，通过块地址就可以访问磁盘

#### 块虚拟化



1. 块虚拟化是指对多块硬盘建立RAID，划分逻辑卷（LUN）；
2. 每个逻辑卷对于使用者都完全等同于一块物理硬盘。物理上来说，这个逻辑卷的所有数据块，都是通过RAID处理，分布在不同的物理硬盘上；
3. 块虚拟化的结果是使存储的使用者无须关系RAID实现的具体过程，只需要像读写普通硬盘一样读写这个逻辑卷，就能获取RAID对数据的保护功能。

#### 磁带/磁带驱动器/磁带库虚拟化



磁带虚拟化：

1. 采用类似RAID的技术对多盘磁盘进行条带化和校验，以期提高磁带使用的可靠性和性能；
2. 由于磁带读写缓慢，难以满足条带和校验的需要，因此RAIT技术很少使用

磁带驱动器虚拟化：

1. 提高磁带驱动器使用效率
2. 加快数据备份的速度

磁带库虚拟化：

VTL将磁盘阵列虚拟为磁带库，使用户在无需改变备份习惯和现有备份软件配置的情况下，具有以下优点：

1. 提高性能，缩短数据备份和恢复窗口；
2. 获得磁盘阵列的RAID保护功能，避免磁带介质故障；
3. 持续扩展备份的性能和容量。

#### 文件系统虚拟化

#### 文件/记录虚拟化

### 技术对比



## 存储池

### 概述

存储池是一个由libvirt管理的文件、目录或存储设备，提供给虚拟机使用。存储池被分为存储卷，这些存储卷保存虚拟镜像或连接到虚拟机作为附加存储。libvirt通过存储池的形式对存储进行统一管理、简化操作。对于虚拟机操作来说，存储池和卷并不是必需的。

### 指令

**virsh中和存储池相关的命令：**

pool-autostart 自动启动某个池

pool-build 建立池

pool-create-as 从一组变量中创建一个池

pool-create 从一个 XML 文件中创建一个池

pool-define-as 在一组变量中定义池

pool-define 在一个XML文件中定义（但不启动）一个池或修改已有池

pool-delete 删除池

pool-destroy 销毁（删除）池

pool-dumpxml 将池信息保存到XML文档中

pool-edit 为存储池编辑 XML 配置

pool-info 查看存储池信息

pool-list 列出池

pool-name 将池 UUID 转换为池名称

pool-refresh 刷新池

pool-start 启动一个（以前定义的）非活跃的池

pool-undefine 取消定义一个不活跃的池

pool-uuid 把一个池名称转换为池 UUID

**virsh中的和存储卷相关命令：**

vol-clone 克隆卷

vol-create-as 从一组变量中创建卷

vol-create 从一个 XML 文件创建一个卷

vol-create-from 生成卷，使用另一个卷作为输入。

vol-delete 删除卷

vol-download 将卷内容下载到文件中

vol-dumpxml 保存卷信息到XML文档中

vol-info 查看存储卷信息

vol-key 根据卷名或路径返回卷的key

vol-list 列出卷

vol-name 根据给定卷key或者路径返回卷名

vol-path 根据卷名或key返回卷路径

vol-pool 为给定密钥或者路径返回存储池

vol-resize 重新定义卷大小

vol-upload 将文件内容上传到卷中

vol-wipe 擦除卷

### 分类

KVM存储模式：

1. 基于文件系统的存储

在安装KVM宿主机时，可选文件系统为dir(directory)或fs(formatted block storage)作为初始KVM存储格式。默认选项为dir，用户指定本地文件系统中的一个目录用于创建磁盘镜像文件。fs选项可以允许用户指定某个格式化文件系统的名称，把它作为专用的磁盘镜像文件存储。两种KVM存储选项之间最主要的区别在于：fs文件系统不需要挂载到某个特定的分区。

两种选项所指定的文件系统，都可以是本地文件系统或位于SAN上某个物理宿主机上的网络文件系统。后者具备一定的优势，因为SAN可以很轻易地实现多个主机同时访问，而本地磁盘或文件系统则无法实现。

还有一种基于文件的磁盘存储方式是netfs，用户可以指定一个网络文件系统的名称，如Samba.用这种方式作为KVM存储很方便，因为这样很容易访问到位于其它服务器上的文件系统，同时用户也可以通过多台宿主机访问磁盘文件。

所有的这些基于文件的KVM存储方式都有一个缺点：文件系统固有缺陷。因为虚拟机的磁盘文件不能直接读取或写入KVM存储设备，而是写入宿主机OS之上的文件系统。这也就意味着在访问和写入文件时中间增加了额外一层，这通常会降低性能。所以，如果您希望寻找KVM虚拟化性能最优方案，最好考虑基于设备的存储。

目录（dir）：Filesystem Directory

文件系统（FS）：Pre-Formatted Block Device

网络文件系统（netfs）：Network Exported Directory

1. 基于设备的存储

共支持四种不同的物理存储：磁盘、iSCSI、SCSI和lvm逻辑盘。

磁盘方式指直接读写硬盘设备。

iSCSI和SCSI方式可选，取决于用户采取SCSI或iSCSI地址把磁盘设备连接。这种KVM存储方式的优势在于，磁盘的名称是固定的，而不需要取决于宿主机OS搜索到磁盘设备的顺序。

这种连接磁盘的方式也有缺点：灵活性不足。虚拟磁盘的大小很难改变，而且（LVM可以改变卷大小）基于设备的KVM存储不支持快照。

#### 基于目录的存储池

#### 基于文件系统的存储池

#### 基于NFS的存储池

命令行创建存储池：

[root@kvm001 ~]# virsh pool-define-as --name kvmnfs --type netfs \

> --source-host 192.168.2.20 \

> --source-path /nfsshare \

> --target /nfstest //需要提前创建目录

#### 基于磁盘的存储池

#### 基于分区的存储池

#### 基于LVM的存储池

如果要优化KVM存储的灵活性，可以使用LVM(Logical Volume Manager)。LVM的优势在于可以使用快照，而快照并不是KVM虚拟化自带的功能。

LVM可以把所有存储放到一个卷组里，从而轻松创建一个逻辑卷。该卷组是物理磁盘设备的一个抽象，所以如果超出可用磁盘空间最大值，还可以向卷组中添加新的设备，从而极大简化了增加存储空间的过程，增加的空间在逻辑卷中直接可以使用。使用LVM使得磁盘空间分配更加灵活，而且增加和删除存储也更为容易。

最后，LVM无论是在单宿主机或多宿主机环境中都可以很好工作。在多宿主机环境中，您可以在SAN上创建逻辑卷。如果使用Cluster LVM，可以很容易的配置成多个主机同时访问某个逻辑卷。

#### 基于iSCSI的存储池

**创建iSCSI的存储池：**

1、命令创建

[root@kvm001 ~]# virsh pool-define-as --name storage01 --type iscsi \

> --source-host 192.168.2.20 \ //存储服务器的地址

> --source-dev iqn.2017-10.com.cloud.www:storage001 \ //target目标

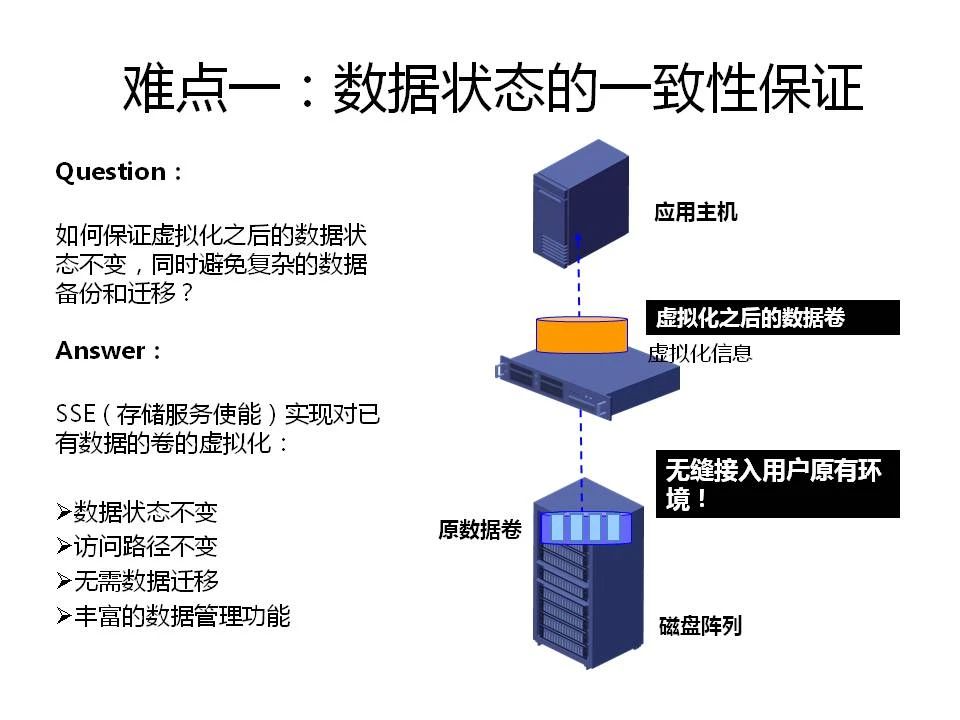
> --target /dev/disk/by-path

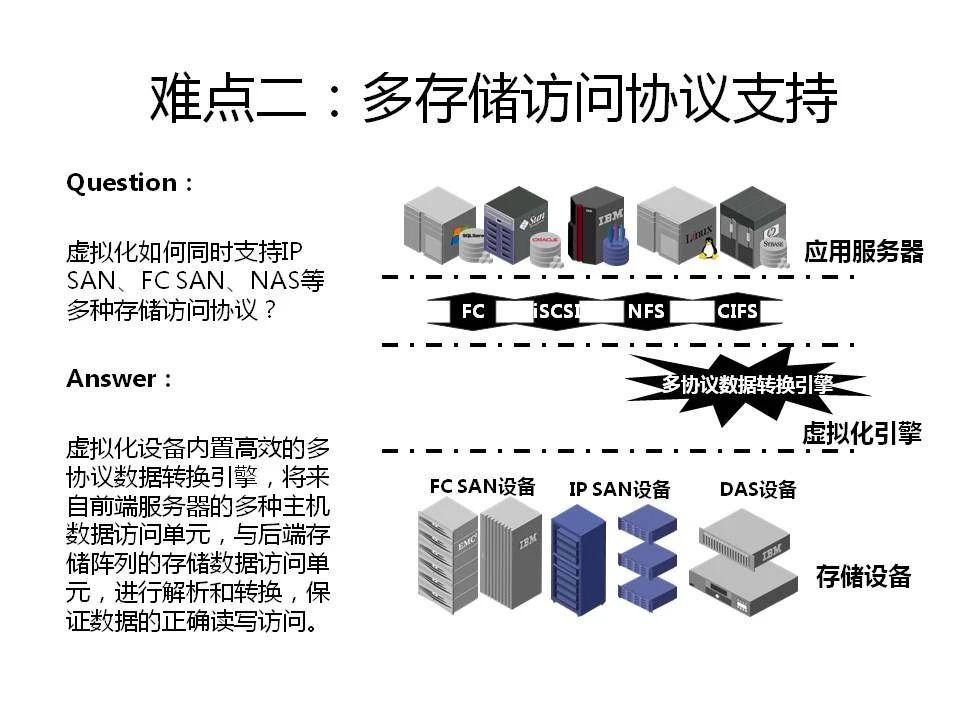
[root@kvm001 ~]# virsh pool-start storage01

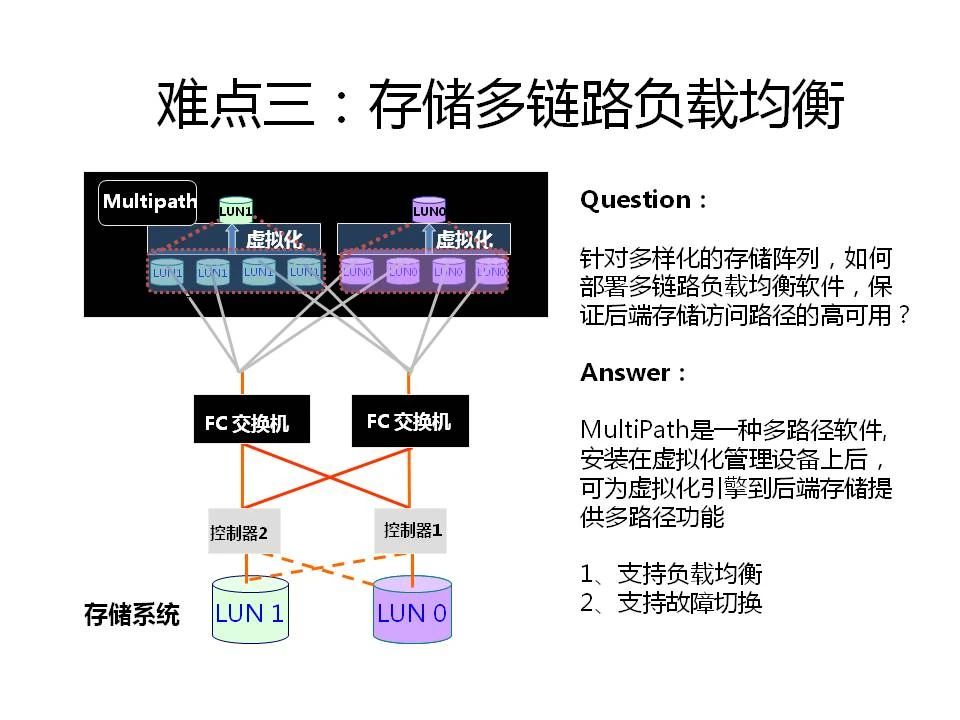
[root@kvm001 ~]# virsh pool-autostart storage01

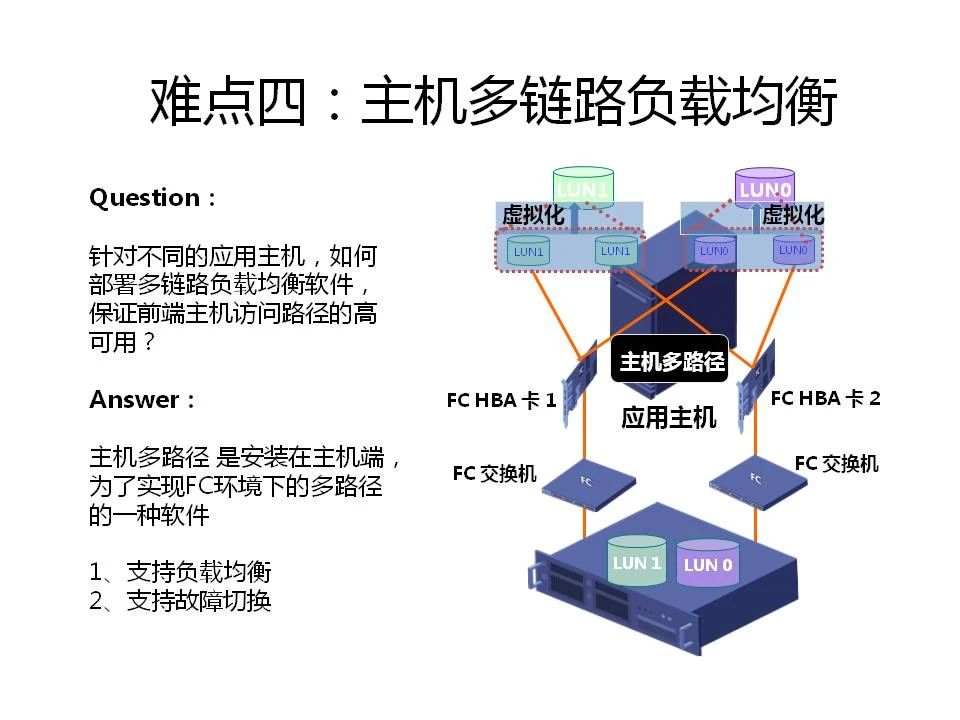
2、virt-manager创建

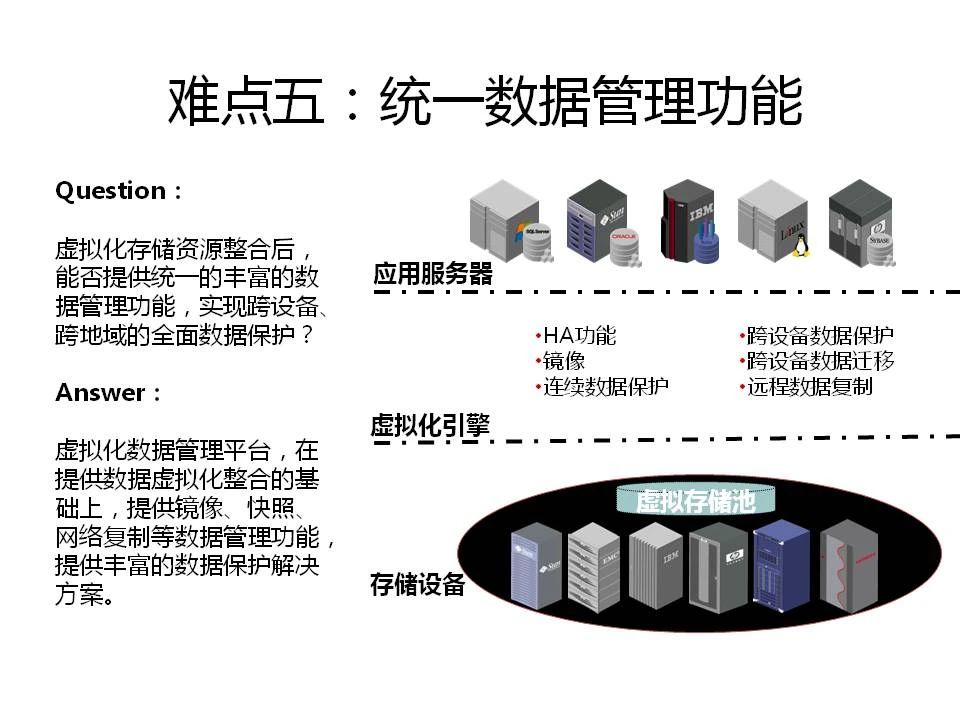
## 技术难点











## 磁盘缓存

### 概述

在配置磁盘的时候，可以指定缓存模式。每个虚拟机的磁盘接口可以配置成writethrough、writeback、none、direstsync或者unsafe缓存模式。

### 性能

### 应用

## 磁盘镜像格式

### 磁盘镜像文件

### 裸设备

### 性能对比

### 应用场景

# 案例分析

例如一个linux下的postfix邮件系统，包含邮件服务器，数据库和网络。从磁盘看邮件系统的一个最大的问题是：不是很多大文件的读写，而是很多小文件的读写，而且这些读写请求是来自同一时间的多个进程或者线程。对这种很多小文件的读写应用服务，在分配邮件用户所在的磁盘时，建议使用Thin provision模式.这样既避免了初始空间的大量占用，也可以做到随需增长。

从内存上看，对于postfix而言，它的每一个进程不会消耗太多的内存，我们期望的是大量的内存被自动使用到磁盘缓存中来提高磁盘I/O速率，当然这个我们不需要操作，linux帮我们完成了！Linux虚拟内存管理默认将所有空闲内存空间都作为硬盘缓存，因此在拥有数GB内存的生产性Linux系统中，经常可以看到可用的内存只有20MB。从处理器上看邮件系统，不管是smtp、imap对CPU的占用都不是很大.这样我们在分配CPU和内存资源时，就可以按照NUMA架构来配置固定大小的单元。比如一台配置了两颗八核处理器以及128GB内存的服务器，虚拟化成4台邮件服务器，就可以每台分配成4核32G。

从网络上看，邮件系统会频繁的使用网络子系统，但是邮件系统的瓶颈还是磁盘的吞吐而不是网络的吞吐，对应这种应用不要求强交互，延迟是允许的，所以网卡是虚拟的还是SR-IOV的影响都不大。

对于邮件系统的数据库服务器，因为小文件随机读写比较多，数据库的磁盘可以选择厚置备模式，提高小数据块的IO。

对于不同的业务系统，具体的问题还需要具体分析，性能优化不是一朝一夕的事，随着业务的发展变化，优化的技术手段和方式都会相应的随之改变.

# virsh指令

## 创建虚机

如果希望借助virt-manager通过复制的方式创建虚机，可以采用如下操作：

1. 通过virt-manager命令打开KVM虚拟机管理图形化界面
2. 选择要复制的虚拟机，右键shut down-force off，将虚拟机关闭
3. 选择要复制的虚拟机，右键clone，弹出窗口，修改虚机名称
4. 修改镜像名称
5. 创建好虚拟机
6. 在virt-manager图形化界面上选中新建的虚拟机，点击open
7. 依次选中Detail-Overview，然后可以修改虚机名称，点击Apply
8. 依次点击Add Hardware，选中Storage，选中SELECT managed or other existing storage，点击浏览选中已经复制的镜像文件，点击close volume，点击完成
9. 修改网卡信息

## 删除虚机

通过virt-manager方式删除KVM虚拟机：

1. 通过virt-manager命令打开KVM虚机图形化管理界面
2. 选择要删除的KVM虚拟机，右键shutdown-force off关闭虚机
3. 选择要删除的虚机，右键delete

如果想彻底删除10.47.181.1这个虚拟机，需要执行以下操作：

1. virsh destroy 10,47,181,1停止虚拟机
2. virsh list –all确认该虚拟机已经处于关闭态
3. virsh undefine 10.47.181.1解除标记
4. 通过查询/etc/libvirt/qemu/10.47.181.1.xml配置文件，找到镜像文件位置/var/lib/libvirt/images/10.47.181.1.qcow2
5. 删除虚机文件：

rm –rf /var/lib/libvirt/images/10.47.181.1.qcow2

rm –rf /etc/libvirt/qemu/10.47.181.1.xml

1. virsh list –all观察10.47.181.1已经被删除

## 修改虚机名

如果想要修改虚拟机名称，可以采取如下操作：

1. 关闭虚拟机：virsh shutdown vm-1
2. 导出xml文件：virsh dumpxml vm-1 > vm-2.xml
3. 编辑vm-2.xml，修改vm名称
4. 重定义：

virsh undefined vm-1

virsh define vm-2.xml

1. 启动虚拟机：virsh start vm-2
2. 查看：virsh list

## 打开虚机问题

在使用virt-manager打开KVM虚拟机输入用户名时键盘无法使用的问题，解决办法如下：

1. 使用virt-manager打开虚机管理界面，右键open，点击run虚机，进入登录界面，点击Not Found进入输入用户界面
2. 此时出现无法输入问题，先将此窗口关闭，然后再关闭的虚拟机上右键Shut down Force Off，将虚机关闭
3. 在图形化界面上选中关闭的虚拟机，点击打开Detail Display Spice，Type选择VNC Server，Keymap选择en-us，点击应用即可