IB00109 云计算技术

授课教师：姜婧妍

jiangjingyan@sztu.edu.cn

2023年

第二章

虚拟化技术

授课教师：姜婧妍

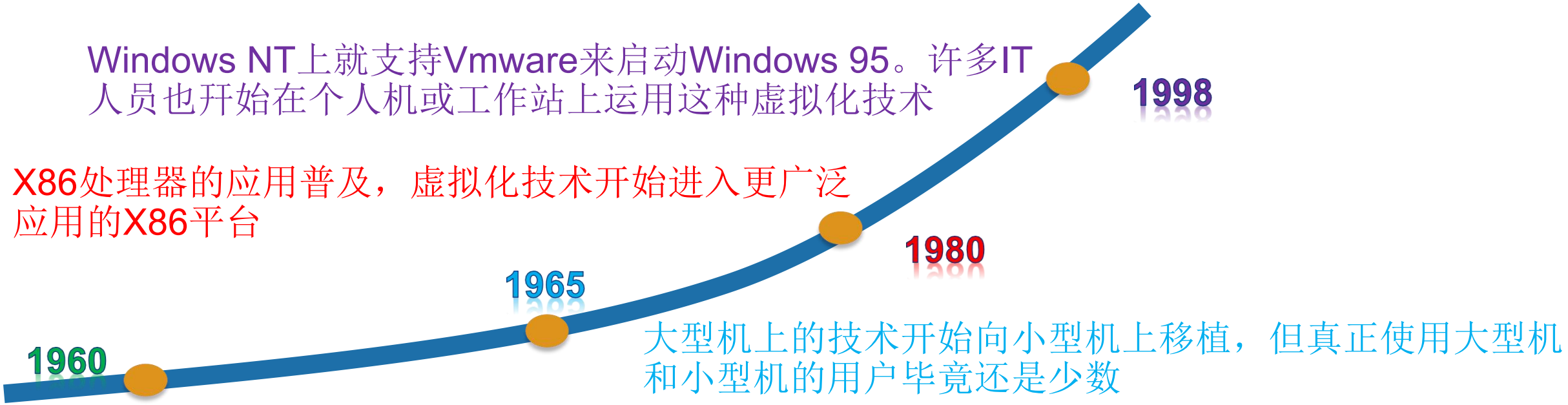
jiangjingyan@sztu.edu.cn

2023年

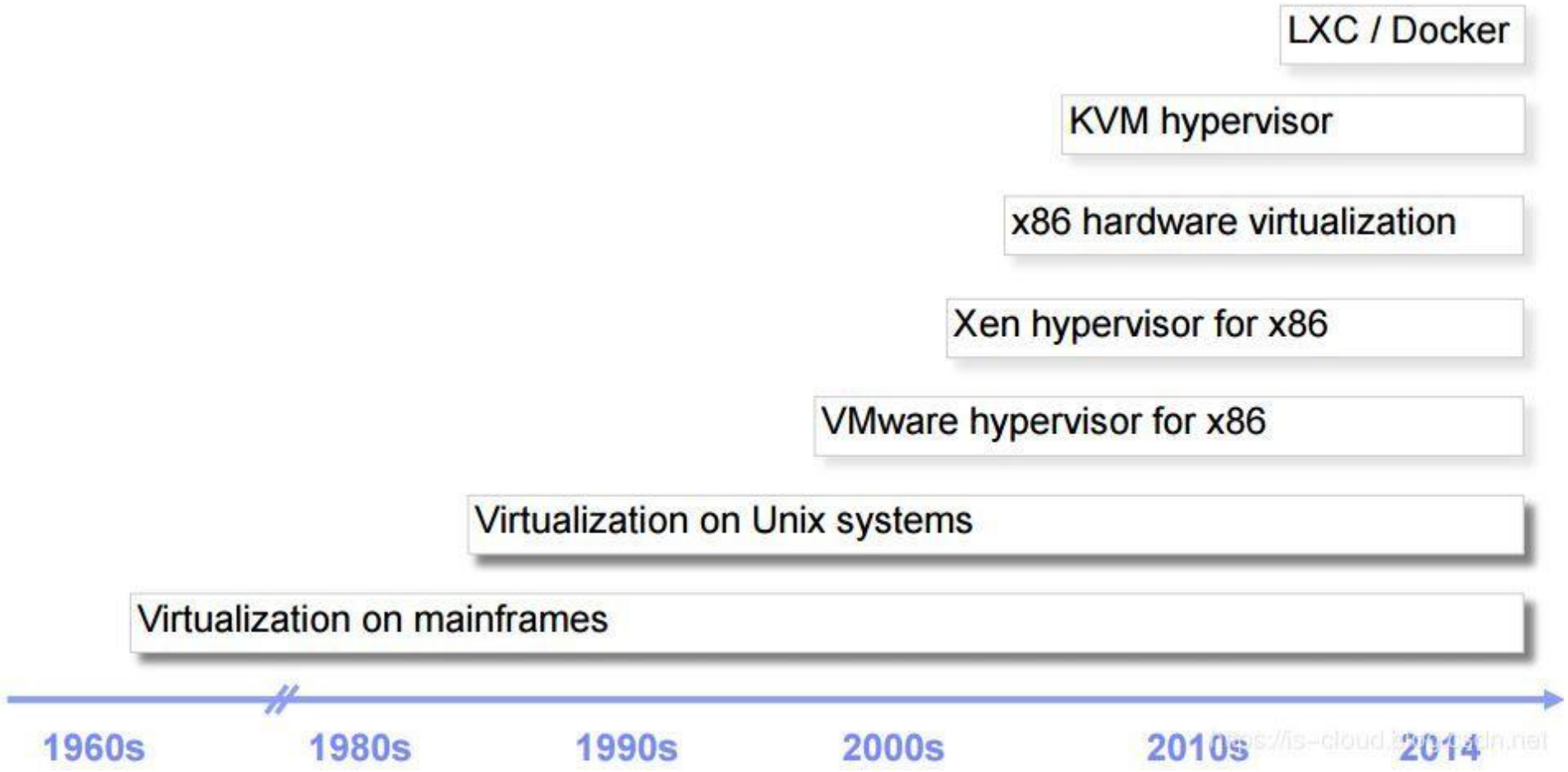
目录 CONTENTS

虚拟简介

- **概念：** 虚拟化，是指将一台计算机虚拟为多台逻辑计算机，每个逻辑计算机可相互独立运行而互不影响，从而显著提高计算机的工作效率。
- **历史：**

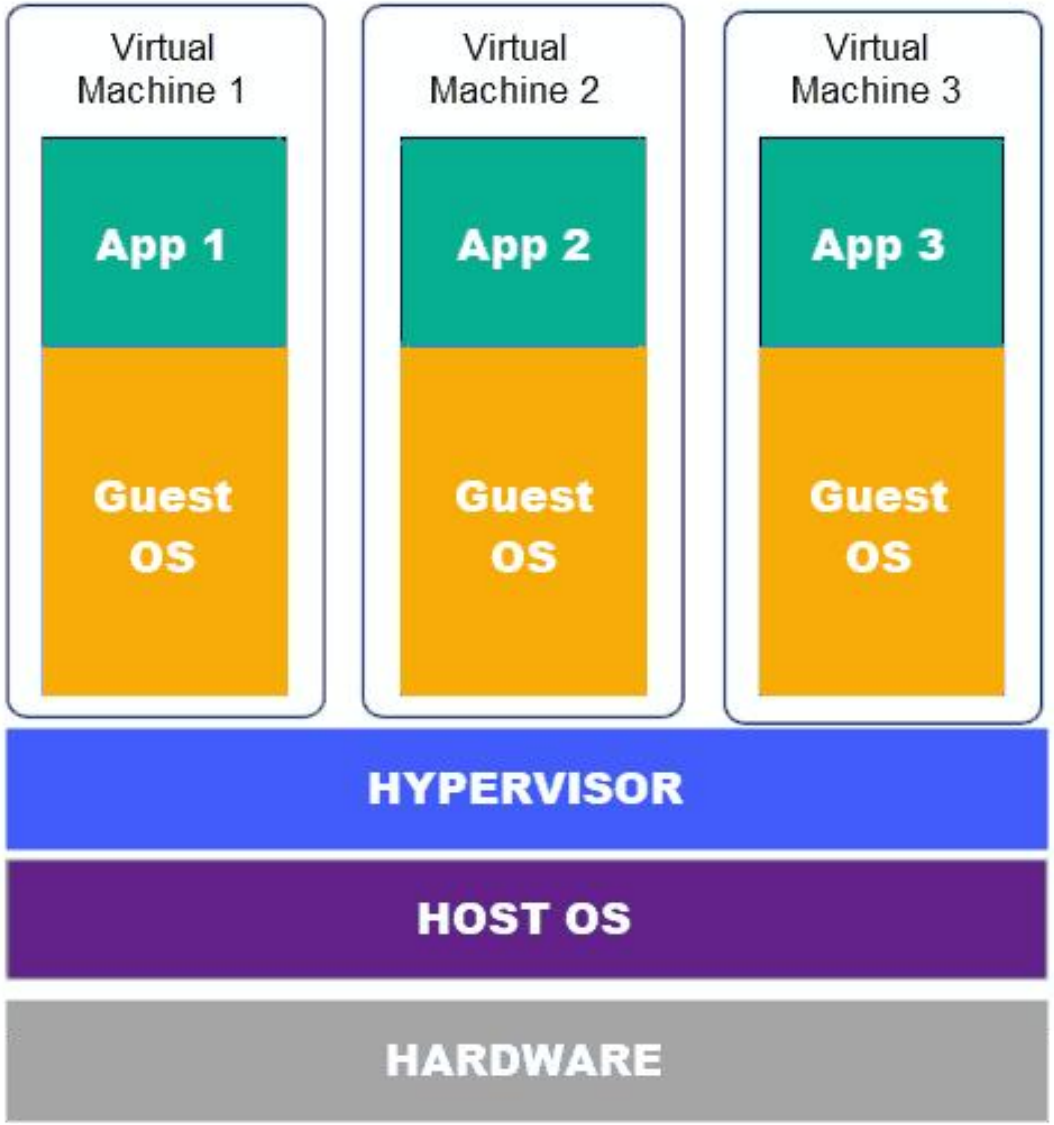


虚拟简介


LXC / Docker
KVM hypervisor
x86 hardware virtualization
Xen hypervisor for x86
VMware hypervisor for x86
Virtualization on Unix systems
Virtualization on mainframes
1960s
1980s
1990s
2000s
2010s
2014

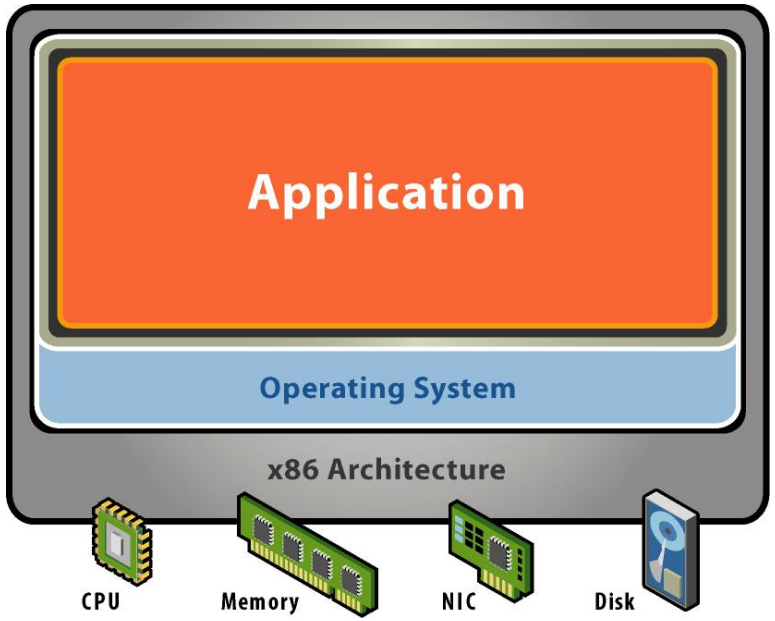
虚拟化中的几个概念

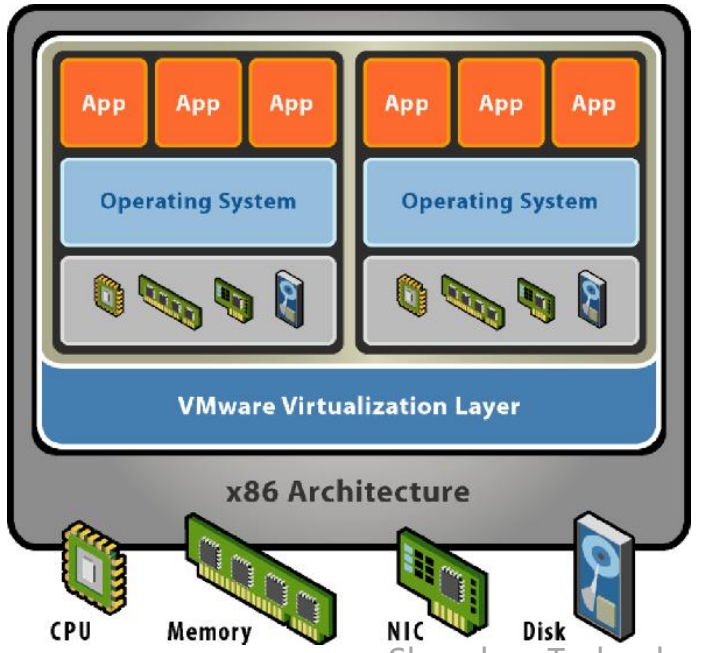
- Host Machine：物理机
- Host OS：运行在物理机之上的OS
- Hypervisor ：又称虚拟机监控器（Virtual Machine Monitor，VMM）
- Virtual Machine：虚拟出来的虚拟机
- Guest OS：运行在虚拟机之上的OS



虚拟化简介

- 虚拟化：是将物理IT资源转换为虚拟IT资源的过程。
- 作用：通过该技术将一台计算机虚拟为多台逻辑计算机。在一台计算机上同时运行多个逻辑计算机，每个逻辑计算机可运行不同的操作系统，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。

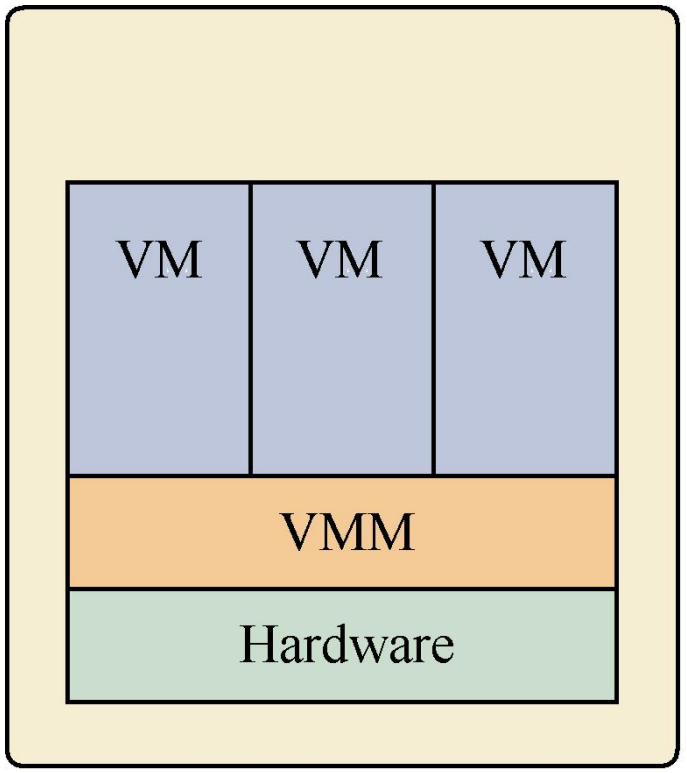




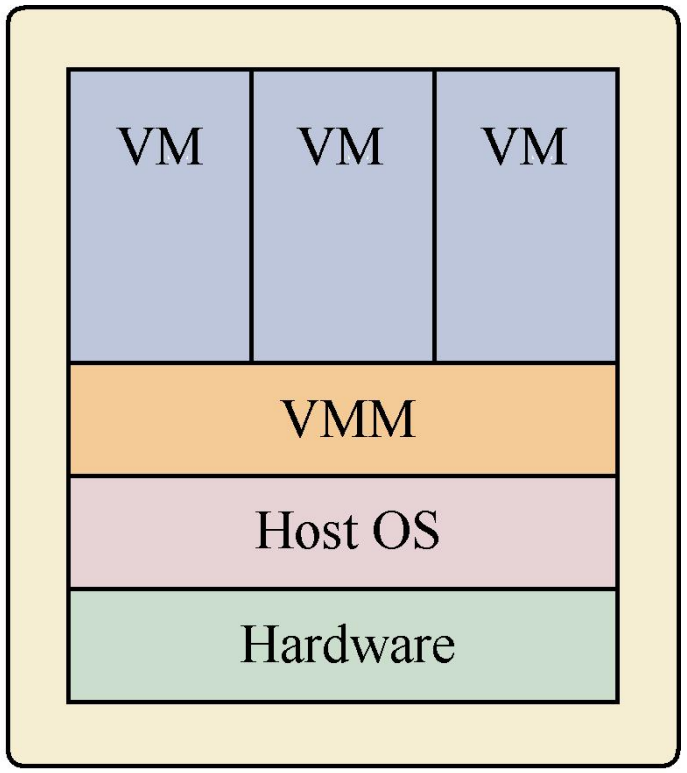
虚拟化简介

波佩克与戈德堡虚拟化需求（Popek and Goldberg virtualization requirements）即：虚拟化系统结构的三个基本条件。满足这些条件的控制程序才可以被称为虚拟机监控器（Virtual Machine Monitor，简称VMM）：

- **资源控制**（Resource Control）。控制程序必须能够管理所有的系统资源。
- **等价性**（Equivalence）。在控制程序管理下运行的程序（包括操作系统），除时序和资源可用性之外的行为应该与没有控制程序时的完全一致，且预先编写的特权指令可以自由地执行。
- **效率性**（Efficiency）。绝大多数的客户机指令应该由主机硬件直接执行而无需控制程序的参与。



（a）裸金属架构



（b）寄居架构

虚拟化软件层所处的位置

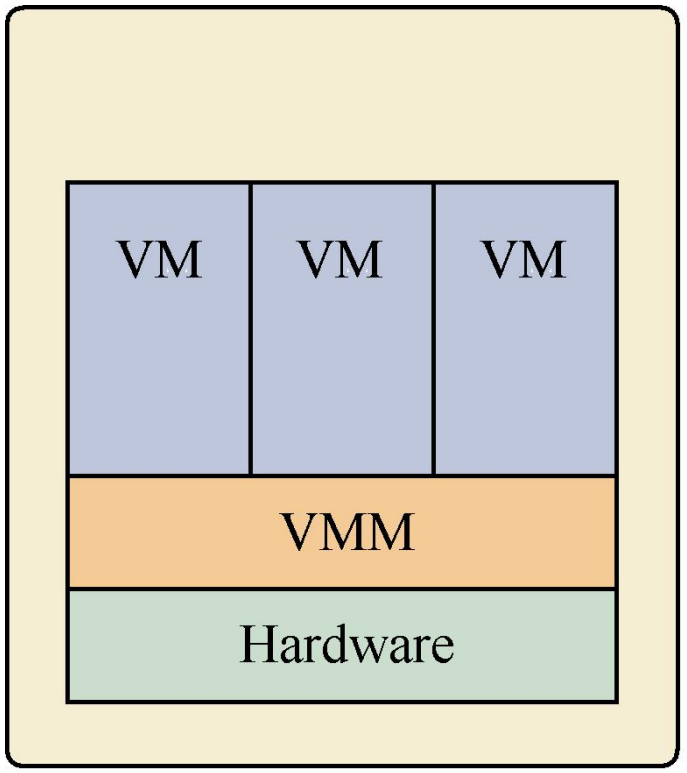
虚拟化简介

类型 I（原生或裸机 Hypervisor）：这些虚拟机管理程序直接运行在宿主机的硬件上来控制硬件和管理客户机操作系统。

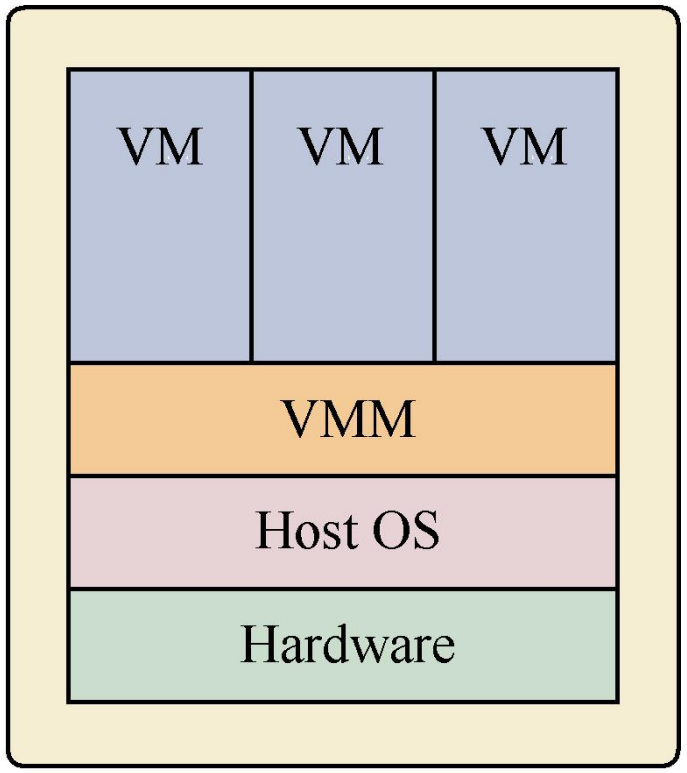
- **需要硬件支持**
- VMM 作为主操作系统
- 运行效率高

类型 II（寄居或托管 Hypervisor）：VMM 运行在传统的操作系统上，就像其他计算机程序那样运行。

- VMM 作为应用程序运行在主操作系统环境内
- 运行效率一般较类型 I 低



（a）裸金属架构



（b）寄居架构

虚拟化软件层所处的位置

虚拟化简介

定价　云市场　合作伙伴　支持与服务　了解阿里云　域名

< 查看全部产品

弹性裸金属服务器

弹性裸金属服务器（ECS Bare Metal Server）是一种可弹性伸缩的高性能计算服务，计算性能与传统物理机无差别，具有安全物理隔离的特点。分钟级的交付周期将提供给您实时的业务响应能力，助力您的核心业务飞速成长。

裸金属七代实例已发布，点此查看。

立即购买　产品控制台　产品文档　产品价格

产品规格　产品优势　产品功能　产品动态　文档与工具

产品规格

通用型　计算型　内存型　高主频型

通用型弹性裸金属服务器 ebmg5s

基于自研的新一代软硬一体化虚拟化技术架构（神龙架构）而打造的创新型计算产品，融合了虚拟机的弹性资源、分钟级交付、全自动运维和物理机的性能无损、完整特性、硬件级强隔离

适用场景

✓ 提供均衡的计算、存储以及网络配置，满足对资源独享、安全隔离、性能有较高要求的业务场景，如容器、数据库、企业核心业务、大数据计算等

CPU内存比	网络带宽能力	网络收发包能力
1：4	32 Gbit/s	450万PPS

实例　96核384G

地域　华北2（北京）

系统盘　40G ESSD云盘

带宽　5M

购买时长　1 个月

¥12405.00 /月

立即购买

虚拟化带来的好处

01

提高资源利用率

通过整合服务器可以将共用的基础架构资源聚合到资源池中，打破原有的一台服务器一个应用程序的模式。

02

降低成本，节能减排

通过使用虚拟化，可以使所需的服务器及相关 IT 硬件的数量变少

03

统一管理

虚拟化系统将资源整合，在管理上十分方便

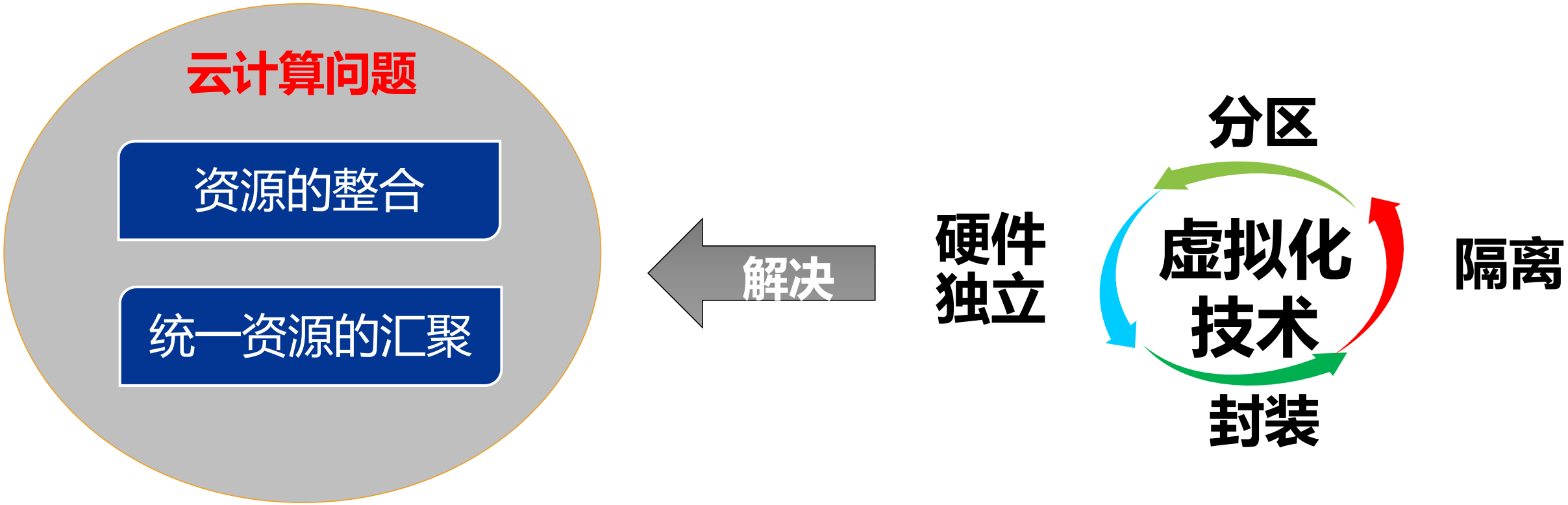
04

提高安全性

用户可以在一台计算机上模拟出多个不同的操作系统，在虚拟系统下的各个子系统相互独立

虚拟化技术

和云计算之间的关系：



虚拟化技术

- 虚拟化技术特点：

分区：大型的、扩展能力强的硬件可被用来作为多台独立的服务器使用；在一个单独的物理系统上，可以运行多个虚拟的操作系统和应用；计算资源可以被放置在资源池中，并能够被有效地控制

虚拟化技术

- 虚拟化技术特点：

隔离：虚拟化能够提供理想化的物理机，每个虚拟机互相隔离；数据不会在虚拟机之间泄露；应用只能在配置好的网络上进行通讯。

虚拟化技术

- 虚拟化技术特点：

封装：虚拟单元的所有环境被存放在一个单独文件中；为应用展现的是标准化的虚拟硬件，确保兼容性；整个磁盘分区被存储为一个文件，易于备份、转移和拷贝

虚拟化技术

- 虚拟化技术特点：

硬件独立：可以在其他服务器上不加修改的运行虚拟机。虚拟技术支持高可用性、动态资源调整，极大地提高系统的可持续运行能力

虚拟化技术

- 虚拟化技术是一种思想：

IT界的所有硬件或者软件都可以一种“服务组合”的抽象思想来处理，即形成一个可被用户灵活调用的资源池，从而实现外部用户业务系统和IT软硬件环境的解耦。

这意味着，外部用户业务系统无须了解软硬件的实现细节，就能方便地使用各式各样的软硬件资源，就好像这些资源放在一个黑箱里一样，只需通过接口就能访问，感受不到其真正的实体和虚体的区别，而这也通常被称为“用户透明化”。

虚拟化分类

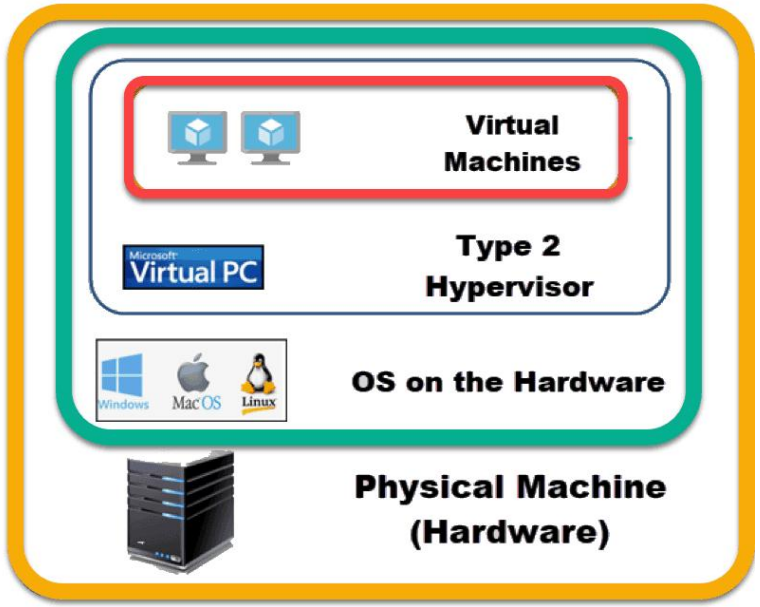
①从虚拟平台角度划分

- **全虚拟化**：虚拟的操作系统,与底层的硬件完全隔离,由中间的Hypervisor层转化。典型的代表有Vmware WorkStation, Microsoft Virtrual Server。
- **半虚拟化**：在虚拟机的操作系统当中加入特定的虚拟化指令,可以直接通过Hypervisor调用硬件资源,免除了Hypervisor层转换指令的开销。典型代表有Xen, Hyper-V。

②从虚拟化的层次划分

- **软件辅助的虚拟化技术**：通过软件的方法,让客户机的特权指令陷入异常,从而触发宿主机进行虚拟化。如Hyper-V等。
- **硬件支持的虚拟化技术**：在X86处理系统架构中加入新的指令机以及运行模式完成虚拟化操作,进而对硬件资源进行直接调用,进行虚拟化。如AMD-V等。

虚拟化分类



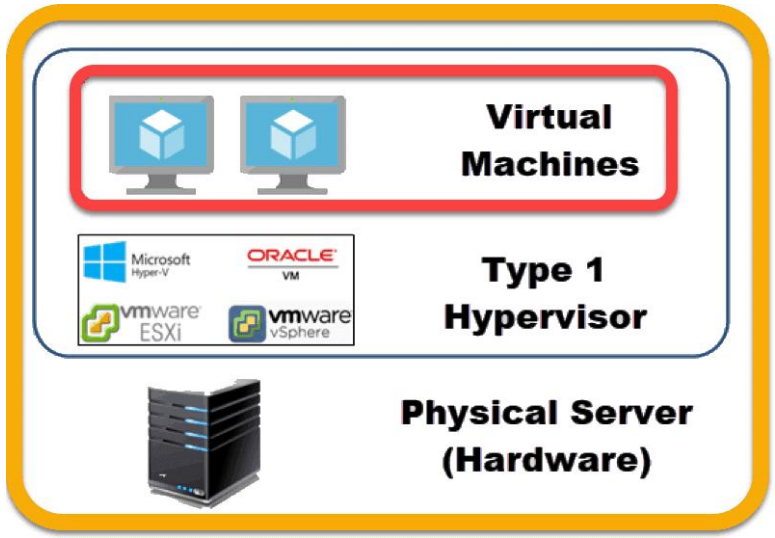
寄居架构

③从虚拟化的实现结构划分

- **基于操作系统的虚拟化**：在一个已存在的操作系统上安装虚拟化软件
- 特点：
  - 简单、易于实现
  - 安装和运行虚拟化程序依赖于主机操作系统对设备的支持
  - 有两层OS，管理开销较大，性能损耗大
  - 虚拟机对各种物理设备的调用，都通过虚拟化层和宿主机的OS一起协调才能完成
- 例子：Vmware Workstation和VirtualBox等

虚拟化分类

时代的选择



裸金属架构

③从虚拟化的实现结构划分

- **基于硬件的虚拟化**：将虚拟化软件直接安装在物理主机硬件上。
- 特点：
  - 不依赖于主机操作系统
  - 支持多种操作系统，多种应用
  - 依赖虚拟化层进行管理
  - 需要对虚拟层内核进行开发
- 例子：VMvare ESX、Xen等

虚拟化分类

④从虚拟化在云计算的应用领域进行划分

- **服务器虚拟化**：将一台服务器虚拟成多台服务器进行使用
- **存储虚拟化**：将整个云系统的存储资源进行统一整合管理
- **应用程序虚拟化**：把应用程序对底层硬件和系统的依赖抽取出来，从而解耦
- **平台虚拟化**：集成各种开发资源虚拟出一个面向开发人员的统一接口，如监控视频平台、消息平台、短信平台
- **桌面虚拟化**：将用户的桌面以使用的终端进行分离，进行解耦

虚拟化技术

①完全虚拟化

- 最流行的虚拟化方法使用名为hypervisor的一种软件,在虚拟服务器和底层硬件之间建立一个抽象层。VMware和微软的VirtualPC是代表该方法的两个商用产品,而基于核心的虚拟机(KVM)是面向Linux系统的开源产品。
- hypervisor可以捕获CPU指令,为指令访问硬件控制器和外设充当中介。因而,完全虚拟化技术几乎能让任何一款操作系统不用改动就能安装到虚拟服务器上,而它们不知道自己运行在虚拟化环境下。主要缺点是, hypervisor给处理器带来开销。

虚拟化技术

②准虚拟化

- 完全虚拟化是处理器密集型技术,因为它要求hypervisor管理各个虚拟服务器,并让它们彼此独立。减轻这种负担的一种方法就是,改动客户操作系统,让它以为自己运行在虚拟环境下,能够与hypervisor协同工作。这种方法就叫准虚拟化(para-virtualization)。
- Xen是开源准虚拟化技术的一个例子。操作系统作为虚拟服务器在Xen hypervisor上运行之前,它必须在核心层面进行某些改变。因此, Xen适用于BSDLinux, Solaris及其他开源操作系统,但不适合对像Windows这些专有的操作系统进行虚拟化处理,因为它们无法改动。
- 准虚拟化技术的优点是性能高。

2

ubuntu/images-testing/hvm-ssd/ubuntu-cosmic-daily-amd64-server-20190318 - ami-0033f98233eadd8fd

Canonical, Ubuntu, 18.10, UNSUPPORTED daily amd64 cosmic image build on 2019-03-18

Root device type: ebs | Virtualization type: hvm | ENA Enabled: Yes

Select

64-bit (x86)

ubuntu/images-testing/ebs-ssd/ubuntu-trusty-daily-amd64-server-20190424 - ami-00365845c96ac739f

Canonical, Ubuntu, 14.04 LTS, UNSUPPORTED daily amd64 trusty image build on 2019-04-24

Root device type: ebs | Virtualization type: paravirtual | ENA Enabled: No

Select

64-bit (x86)

Windows_Server-2019-English-Full-Base-2021.09.15 - ami-0428fc1ee1bde045a

Microsoft Windows Server 2019 with Desktop Experience Locale English AMI provided by Amazon

Root device type: ebs | Virtualization type: hvm | ENA Enabled: Yes

Select

64-bit (x86)

Windows_Server-2019-English-Full-ContainersLatest-2021.09.15 - ami-0e8900c6c58aa0b89

Microsoft Windows Server 2019 with Containers Locale English AMI provided by Amazon

Root device type: ebs | Virtualization type: hvm | ENA Enabled: Yes

Select

64-bit (x86)

Windows_Server-2019-English-Full-SQL_2017_Standard-2021.09.15 - ami-02698d804aed02860

Microsoft Windows Server 2019 Full Locale English with SQL Standard 2017 AMI provided by Amazon

Root device type: ebs | Virtualization type: hvm | ENA Enabled: Yes

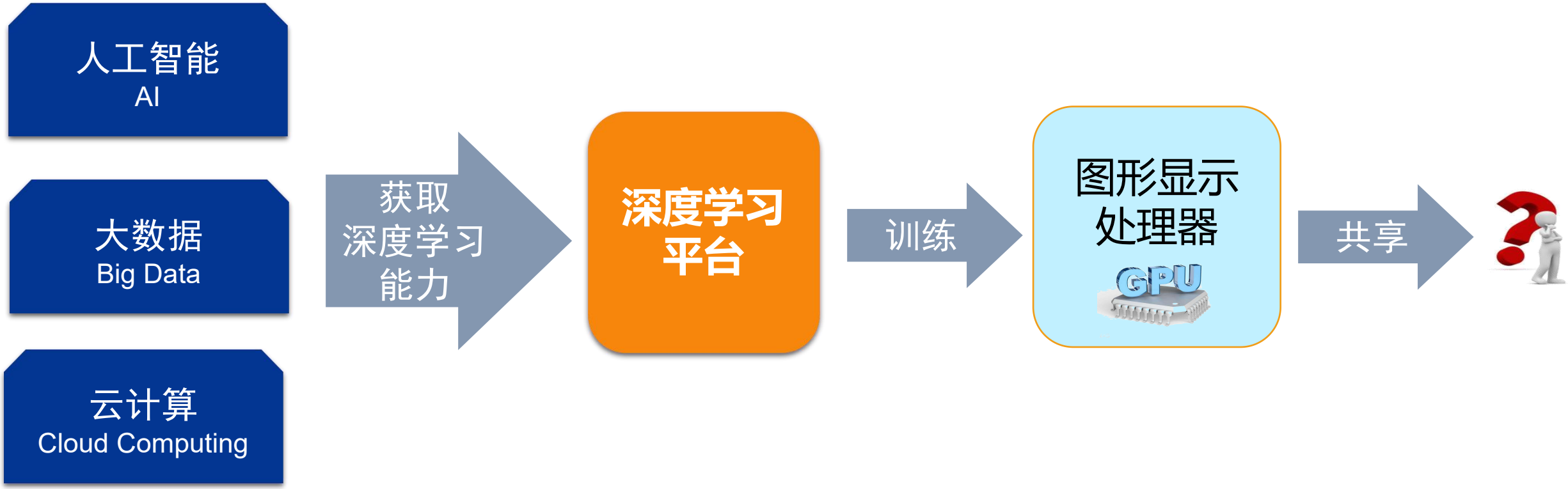
Select

64-bit (x86)

02 计算虚拟化

计算虚拟化

- 场景：公有云计算的深度学习平台



计算虚拟化

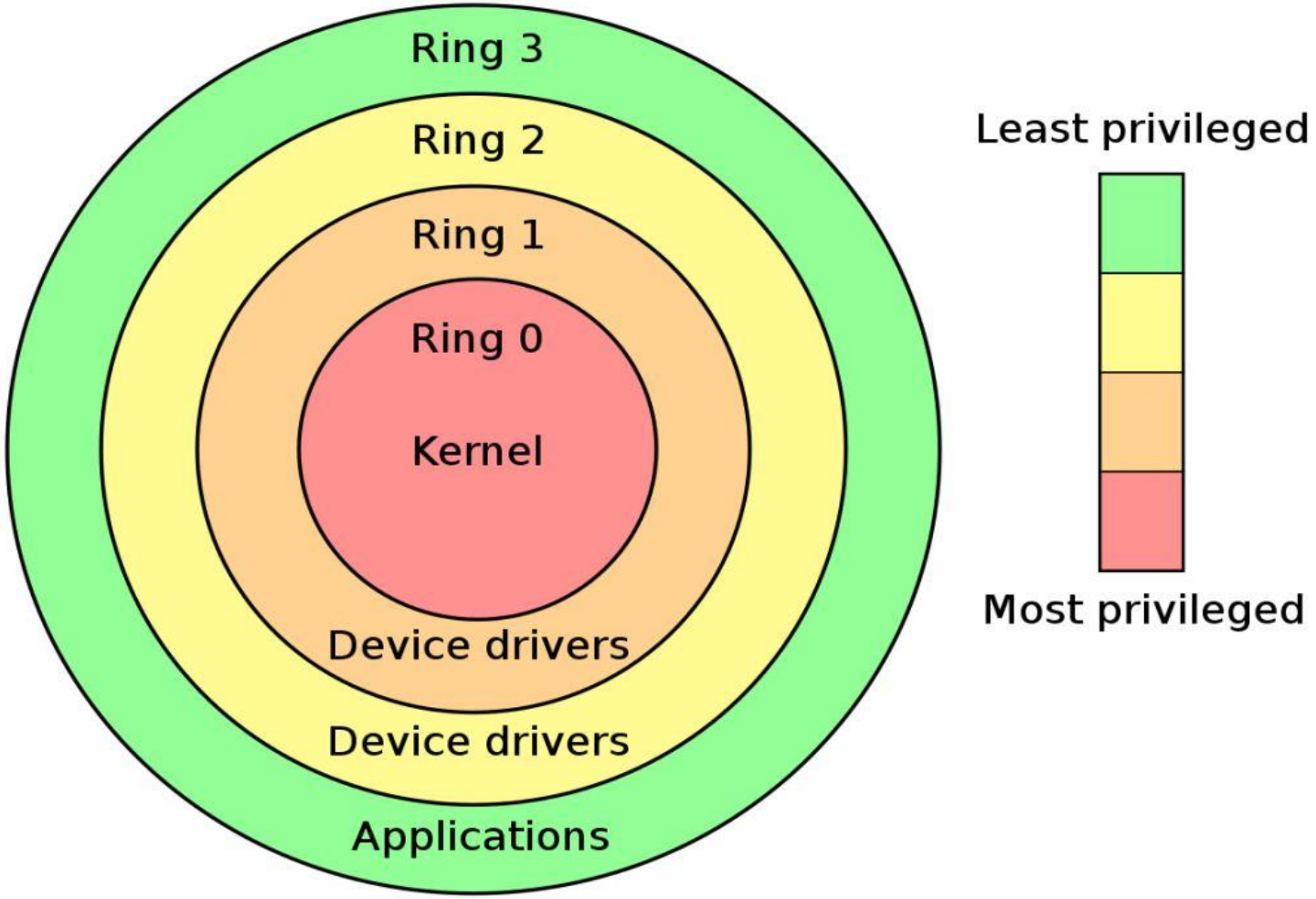
- 定义：

  计算虚拟化是指在物理服务器的宿主机操作系统(Host OS)中加入一个虚拟化层(Hypervisor)，在虚拟化层之上可以运行多个客户端操作系统(Guest OS)。资源利用率和灵活性。

- 开源计算虚拟化软件：KVM（Kernel-based Virtual Machine）
  - CPU虚拟化
  - 内存虚拟化
  - GPU虚拟化

计算虚拟化

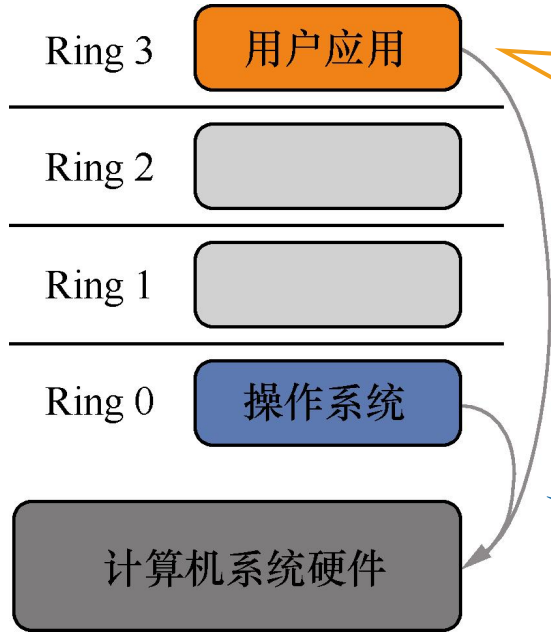
- CPU 为了保证程序代码执行的安全性，多用户的独立性以及保证操作系统的稳定性，提出了 **CPU 执行状态的概念**。
- 它有效的限制了不同程序之间的数据访问能力，避免了非法的内存数据操作，同时也避免了应用程序错误操作计算机的物理设备。
- 一般的，CPU 都会划分为用户态和内核态，而 x86 CPU 更是细分为了 **Ring 0~3 四种执行状态。**



计算虚拟化 – CPU虚拟化

- CPU的虚拟化技术可以单CPU模拟多CPU并行，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。

？在虚拟化中，因为宿主操作系统是工作在ring0的，客户操作系统就不能也在ring0了，怎么办?



x86架构下指令执行方式

用程序的代码运行在最低运行级别上ring3上，不能做受控操作。如果要做，比如要访问磁盘，写文件，那就要通过执行系统调用（函数），执行系统调用的时候，CPU的运行级别会发生从ring3到ring0的切换，并跳转到系统调用对应的内核代码位置执行，这样内核就为你完成了设备访问，完成之后再从ring0返回ring3。

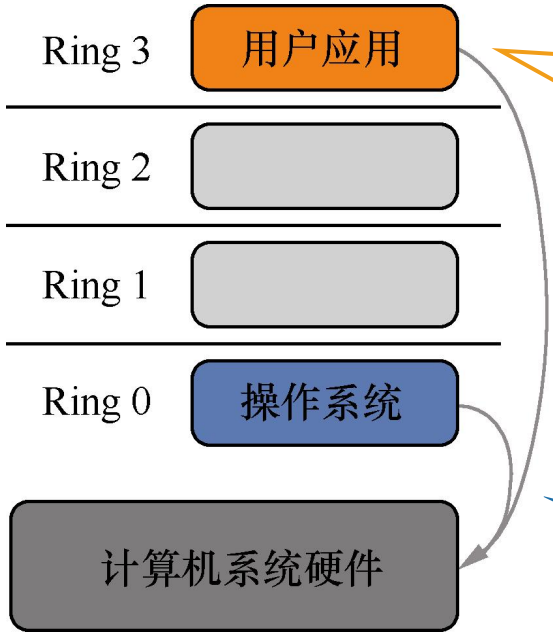
操作系统（内核）的代码运行在最高运行级别ring0上，可以使用特权指令，控制中断、修改页表、访问设备等等。

计算虚拟化 – CPU虚拟化

- CPU的虚拟化技术可以单CPU模拟多CPU并行，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。

？在虚拟化中，因为宿主操作系统是工作在ring0的，客户操作系统就不能也在ring0了，怎么办?

- 通过虚拟机监控器（VMM）可实现客户操作系统对硬件的访问，根据其原理不同分为以下3种技术：
  - 全虚拟化；
  - 半虚拟化；
  - 硬件辅助虚拟化



x86架构下指令执行方式

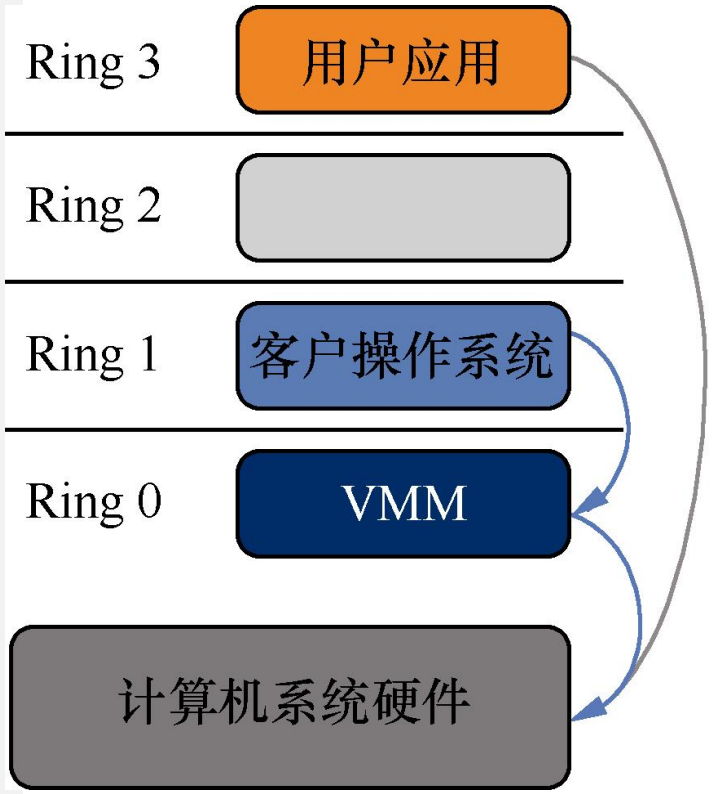
用程序的代码运行在最低运行级别上ring3上，不能做受控操作。如果要做，比如要访问磁盘，写文件，那就要通过执行系统调用（函数），执行系统调用的时候，CPU的运行级别会发生从ring3到ring0的切换，并跳转到系统调用对应的内核代码位置执行，这样内核就为你完成了设备访问，完成之后再从ring0返回ring3。

操作系统（内核）的代码运行在最高运行级别ring0上，可以使用特权指令，控制中断、修改页表、访问设备等等。

计算虚拟化 – CPU虚拟化

3.2 CPU全虚拟化

- 全虚拟化指的是虚拟机完完全全的模拟了计算机的底层硬件，包括处理器，物理内存，时钟，各类外设等等。这样呢，就不需要对原有硬件和操作系统进行改动。
- 在虚拟机软件访问计算机的物理硬件就可以看做软件访问了一个特定的接口。这个接口是由VMM（由Hypervisor技术提供）提供的既VMM提供完全模拟计算机底层硬件环境，并且这时在计算机（宿主机）上运行的操作系统（非虚拟机上运行的操作系统）会被降级运行（Ring0变化到Ring1）。

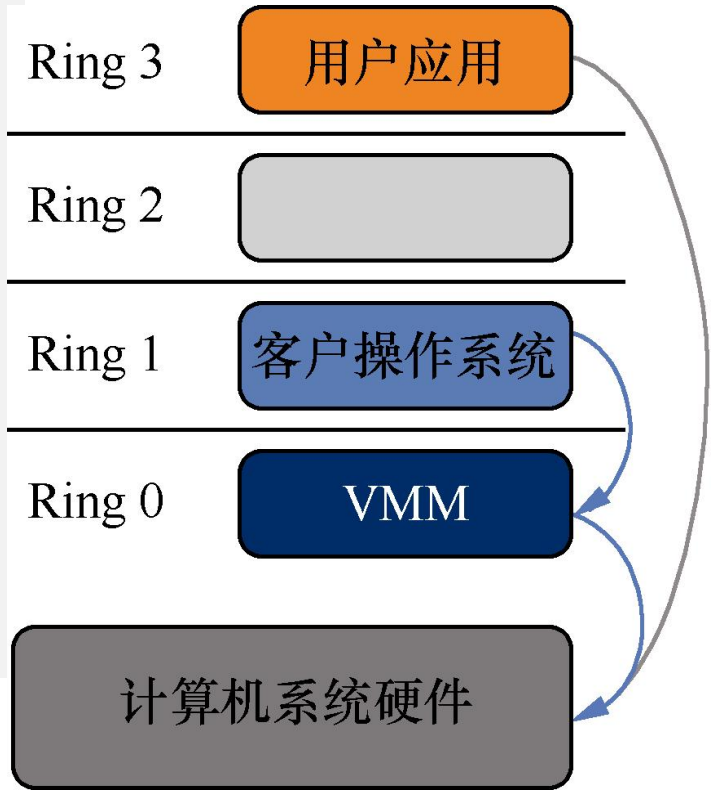


使用VMM二进制翻译客户操作系统的请求

计算虚拟化 – CPU虚拟化

3.2 CPU全虚拟化

- 二进制翻译技术简称BT，是一种直接翻译可执行二进制程序的技术，能够把一种处理器上的二进制程序翻译到另一种处理器上执行。
- 虚拟化软件层将操作系统的指令翻译并将结果缓存供之后使用，而用户级指令无须修改就可以运行，具有和物理机一样的执行速度。



直接执行用户态请求

对内核态请求进行二进制翻译

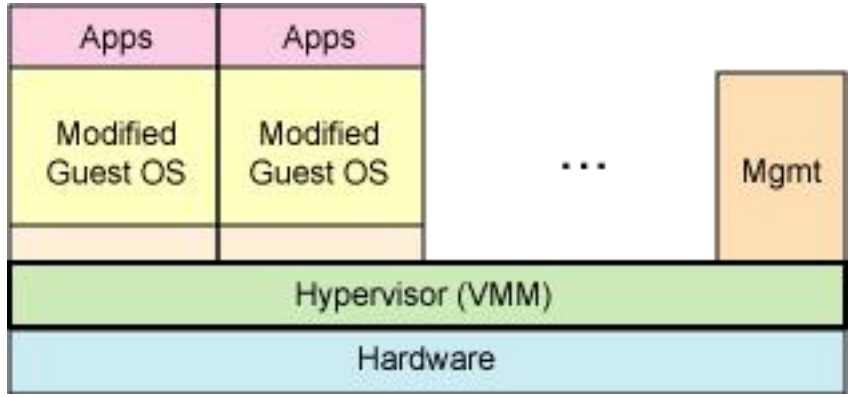
“捕获异常-翻译-模拟”性能损耗大

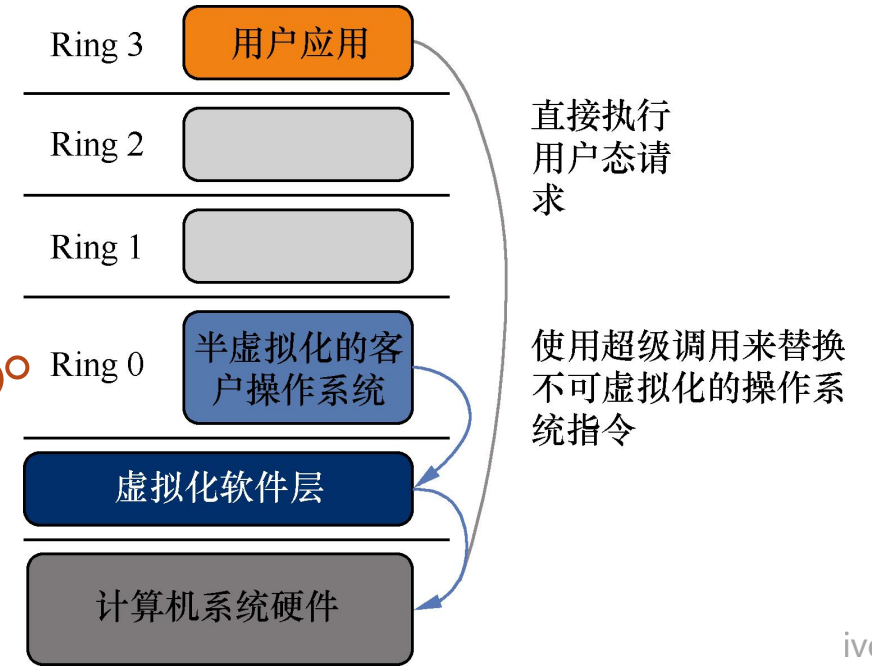
使用VMM二进制翻译客户操作系统的请求

计算虚拟化 – CPU虚拟化

3.2 CPU半虚拟化

- 主要采用Hypercall技术。Guest OS的部分代码被改变，从而使Guest OS会将和特权指令相关的操作都转换为发给VMM的Hypercall (超级调用)，由VMM继续进行处理。而Hypercall支持的批处理和异步这两种优化方式，使得通过Hypercall能得到近似于物理机的速度。





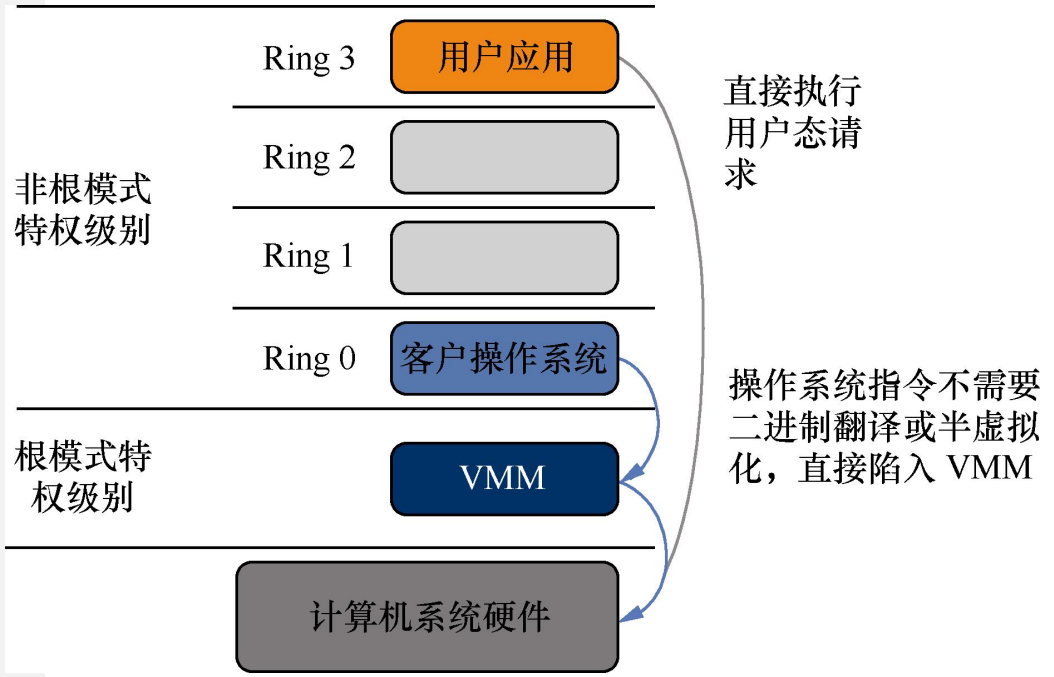
？ 对于 Linux 而言自然是改了就改了，但 Windows 你要怎么改?

大幅度提高性能

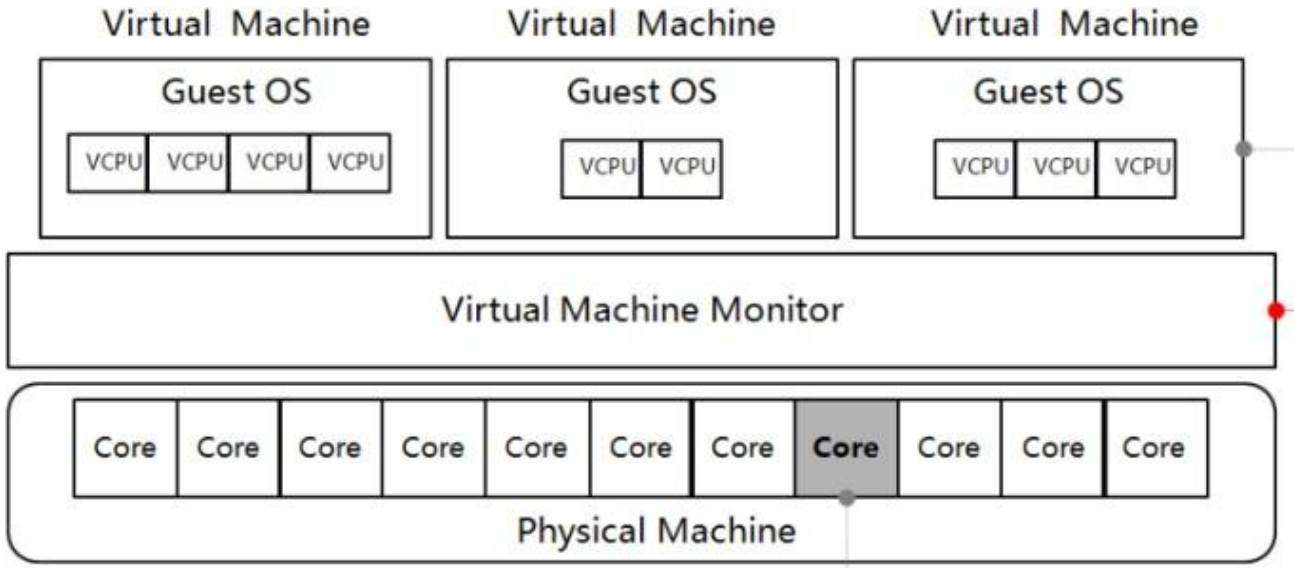
计算虚拟化 – CPU虚拟化

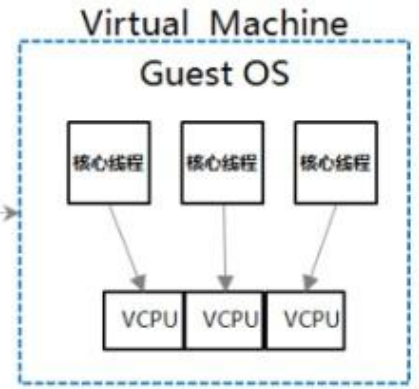
3.3 CPU硬件辅助虚拟化技术

- 目前主要有**Intel的VT-x和AMD的AMD-V**这两种技术。其核心思想都是通过引入新的指令和运行模式,使VMM和Guest OS分别运行在不同模式(**ROOT模式**和**非ROOT模式**)下,且Guest OS运行在Ring 0下。
- 通常情况下, Guest OS的核心指令可以直接下达到计算机系统硬件执行,而不需要经过VMM,当Guest OS执行到特殊指令的时候,系统会切换到VMM,让VMM来处理特殊指令。

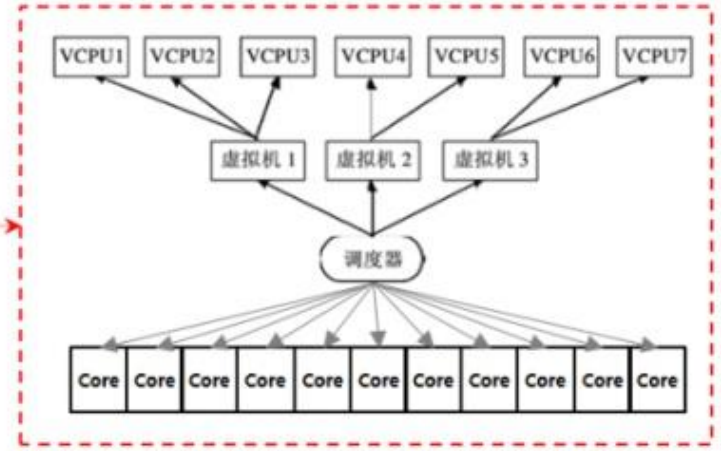


计算虚拟化 – CPU虚拟化

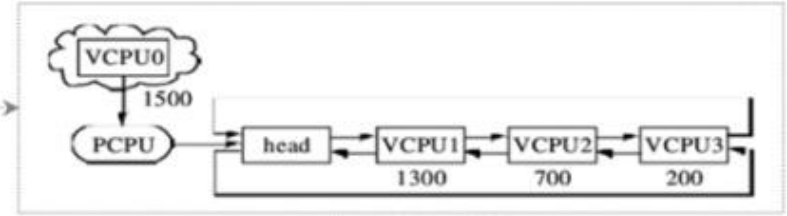




2级调度：客户操作系统负责线程在VCPU上的调度



1级调度：虚拟机监视器负责VCPU在物理CPU上的调度



物理CPU：每个物理CPU关联着一个VCPU执行队列

计算虚拟化 – CPU虚拟化

CPU资源池中的资源是什么？ 算力（用主频表达）

- **什么是主频？** CPU的主频代表CPU的一个“核”每秒计算的次数（如：2.9GHz主频的CPU，可以每秒计算2.9G次，即29亿次）
- **一颗CPU的算力是多少（未开启超线程）？** CPU核心数*主频=整个CPU的“算力”
- **一颗CPU的算力是多少（开启超线程）？** CPU核心数*2*主频=整个CPU的“算力” （超线程是将一颗物理CPU通过时分复用的方式变为2个逻辑CPU，操作系统识别到的就是逻辑CPU）
- **一台服务器的算力是多少？** CPU个数*CPU核心数*2*主频=整个服务器的“算力”
- **一个服务器集群的算力是多少？** 服务器1+服务器2 的“算力” =整个集群的“算力”

示例： 一个集群中包含3台双路服务器，所有的CPU均采用Intel Xeon Gold 6226R

问题：这个集群总算力是多少？556.8GHz=2*16*2*2.9GHz*3

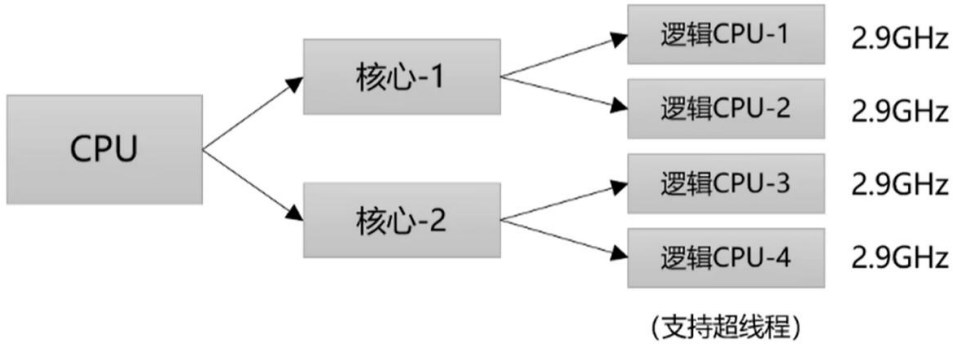
- **2（第一个）：** 代表每个服务器2颗服务CPU
- **16 ：** 一颗CPU有16个核心
- **2（第二个）：** 代表支持超线程，使得每个物理CPU可以变为2个逻辑CPU
- **2.9GHz：** 代表每个逻辑CPU的主频

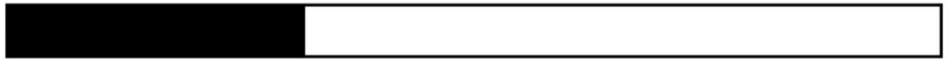
（算出单台服务器算力=185.6GHz）

- **3：** 代表3台服务器（因为三台服务器配置相同故直接乘3即可）

（算出单台服务器算力=556.8GHz）

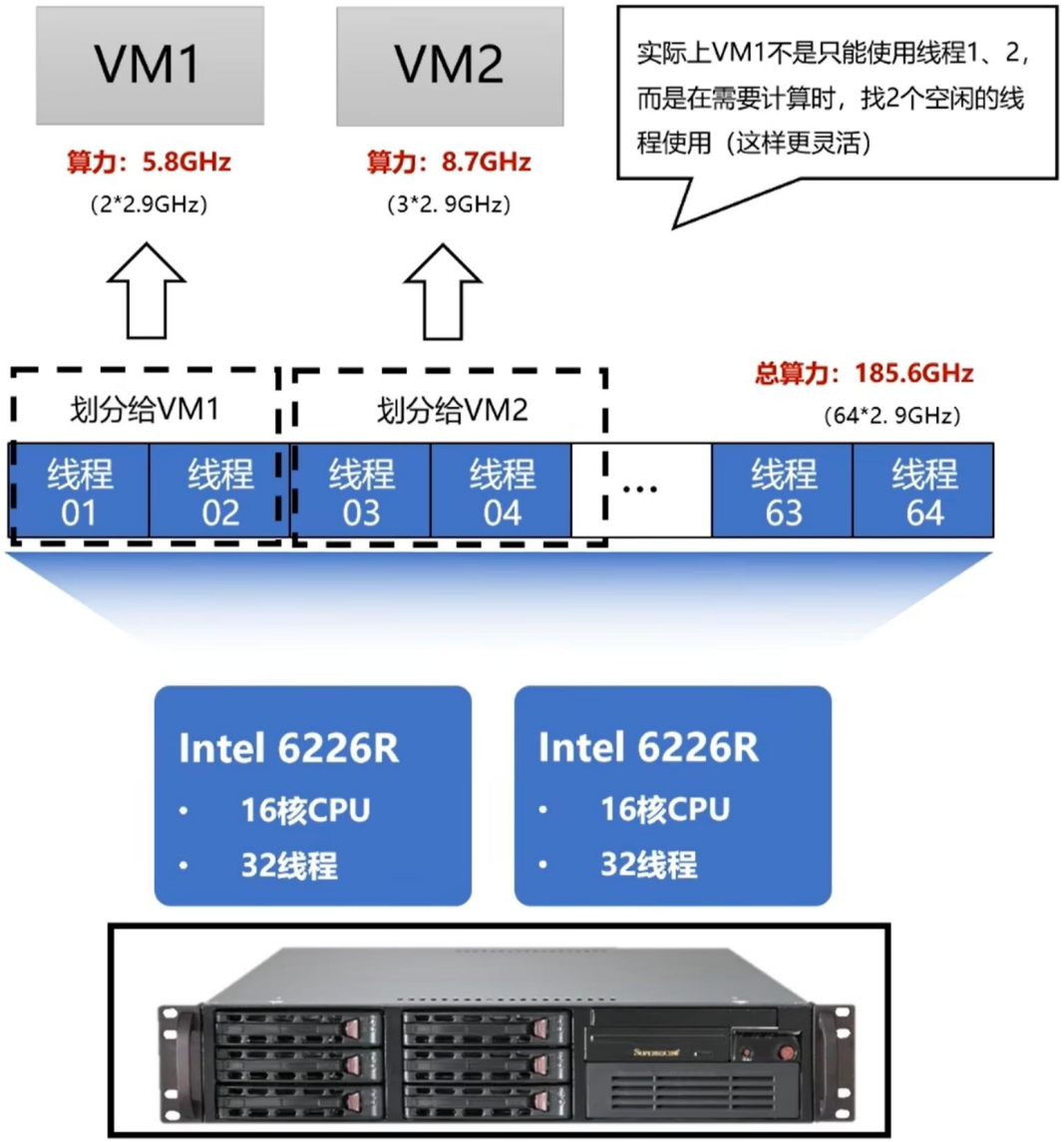
（备注：主频其实并不等于CPU的计算能力，但也是描述CPU计算能力的主要参数，云厂家大都采用该参数描述算力）





- 总容量：556.8GHz
- 已用：50GHz
- 可用：506.8GHz

计算虚拟化 – CPU虚拟化



VM如何申请资源?

1、所有VM划分“总算力”

- 让每个VM可以使用总算力的一部分，划分颗粒度是基于逻辑CPU划分的（即线程）
- 比如：某台服务器总算力185.6GHz，其中VM1从总算力中划分出2个线程使用，VM2从总算力中划分出3个线程使用。VM1就会获得5.8GHz的算力；VM2获得8.7GHz的算力

2、VM通过配置的vCPU“划分”算力

- VM在创建时，通过配置vCPU来控制为期分配的线程数，通“vCPU”表达
- 一个“vCPU”代表要划分一个线程（即，逻辑CPU）
- 比如：创建一个VM，CPU部分的配置为6vCPU，代表要从总算力池中划分6个线程给这个VM使用

限制：一个VM的vCPU个数不能大于物理线程数

- 比如：一台服务器共64个线程，一个VM最大可以配置的vCPU个数就是64，不能配置65个vCPU

问题：“线程”被VM划分完了怎么办?

- 具体问题：32个VM每个VM 2线程，用完了64线程，当我想创建第几33个VM时是不是无法创建?
- 答案：可以创建，可以通过vCPU的超分配实现

思考：为什么要支持超分配?

- *因为直接划分线程，每个VM的物理资源也不会一直100%，依然存在资源浪费的问题*
- *超分配可以让资源的复用更极致*

计算虚拟化 – 内存虚拟化

- 定义：

  通过内存虚拟化来对物理系统内存进行共享，并将其动态分配给虚拟机，操作系统保持着虚拟页到物理页的映射

- 相关概念：物理页和虚拟页

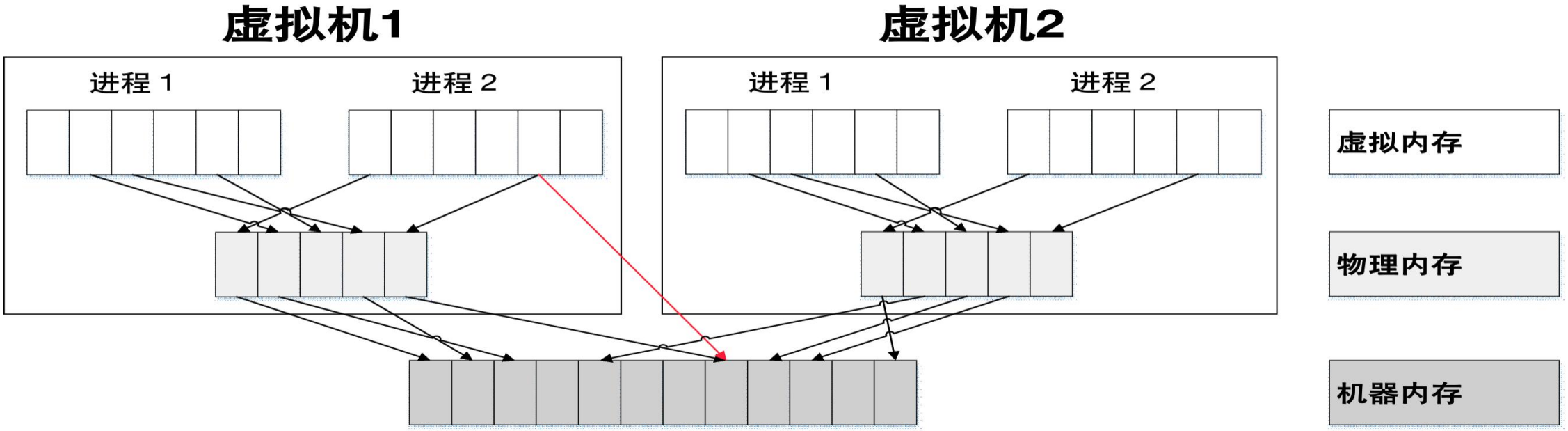
  为便于管理，物理内存被分页，就像一本书里面的好多页纸，每张纸上记录了不同的信息。对于32位的CPU来说，每个物理页大小是4K。与之对应的，虚拟页指的是虚拟内存中的分页。

- 内存虚拟化：

  让客户机使用一个隔离的、从零开始且具有连续的内存空间，KVM 引入一层新的地址空间，只是宿主机虚拟地址空间在客户机地址空间的一个映射。

计算虚拟化 – 内存虚拟化

- 图示：KVM 为了运行多台虚拟机于一台物理机器上，需要实现虚拟内存到物理内存到 机器内存直接的地址转换。



常见的虚拟化产品

- Hyper-V虚拟化
- Xen虚拟化
- Vmware虚拟化
- VirtualBox虚拟化
- KVM虚拟化
- Docker虚拟化

06 容器技术

容器技术

- 场景：平台虚拟化管理

某公司的平台上，一台 16 核 32G 内存的虚拟机上，需要跑 500+ 个用户的应用，有两个事情很重要： ①资源隔离：比如限制应用最大内存使用量，或者资源加载隔离等；②低消耗：虚拟化本身带来的损耗需要尽量的低。不可能在一台机器上开 500 个虚拟机，虽然可以在资源隔离方面做的很好，但这种虚拟化本身带来的资源消耗太严重。

如何在平台上，进行虚拟化的有效管理呢?

容器技术

•传统虚拟化技术的缺陷：

1.每一个虚拟机都是一个完整的操作系统，所以需要给其分配物理资源，当虚拟机数量增多时，操作系统本身消耗的资源势必增多

2.开发环境和线上环境的通常存在区别，所以开发环境与线上环境之间无法达到很好的桥接，在部署上线应用时，依旧需要花时间去处理环境不兼容的问题

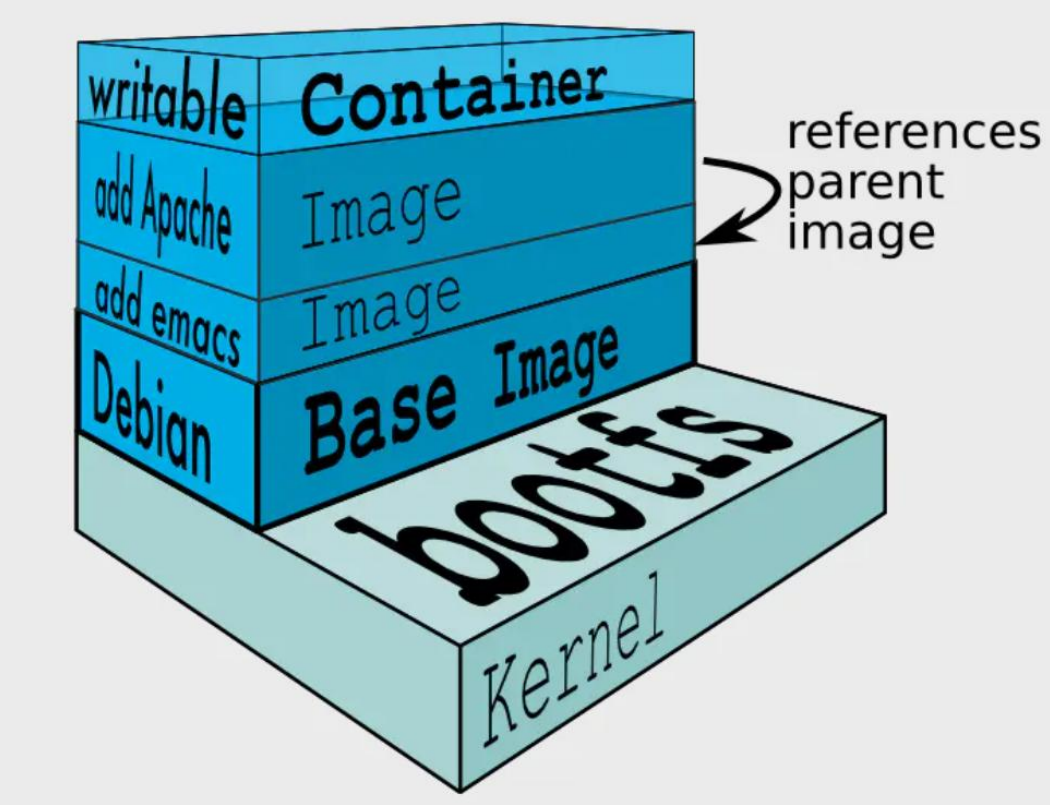
•新型虚拟化技术：容器

容器可以把开发环境及应用整个打包带走，打包好的容器可以在任何的环境下运行，这样就可以解决开发与线上环境不一致的问题了

•代表应用：Docker

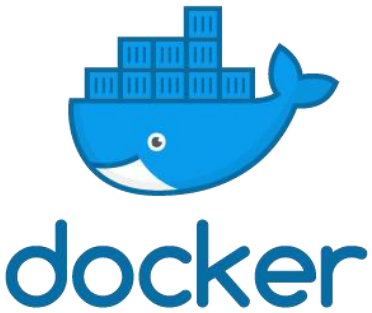
容器简介

- 容器：利用一个开源的容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任一Linux或Windows机器上，也可以实现虚拟化。
- 镜像：可执行的独立软件包，包含软件运行的内容：代码，运行时环境，系统工具，系统库和设置。



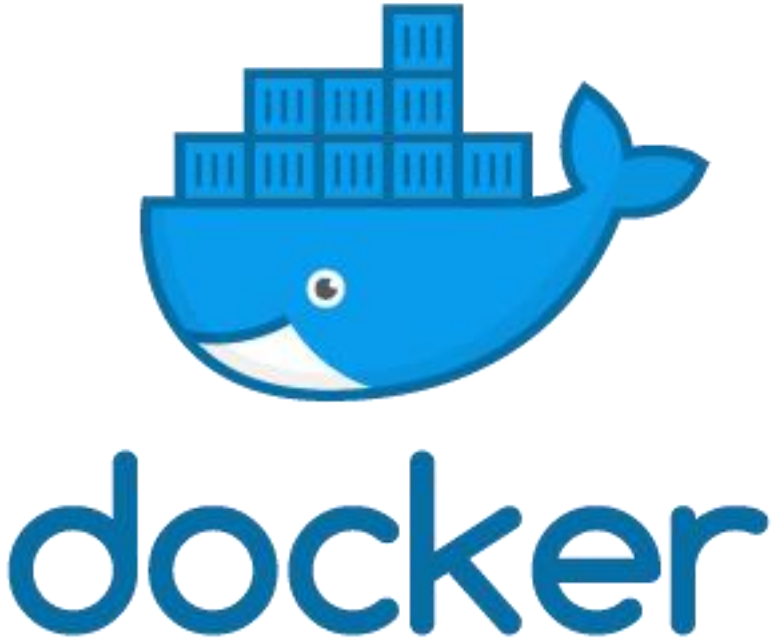
主流容器技术

- Docker：基于容器技术的轻量级虚拟化解决方案。Docker是容器引擎，把Linux的cgroup、namespace等容器底层技术进行封装抽象，为用户提供了创建和管理容器的便捷界面（包括命令行和API）
- 作用：将应用程序与该程序的依赖，打包在一个文件里，运行这个文件，就会生成一个虚拟容器。程序在这个虚拟容器里运行，就好像在真实的物理机上运行一样。有了Docker，就不用担心环境问题。
- 核心：实现应用与运行环境整体打包以及打包格式统一。



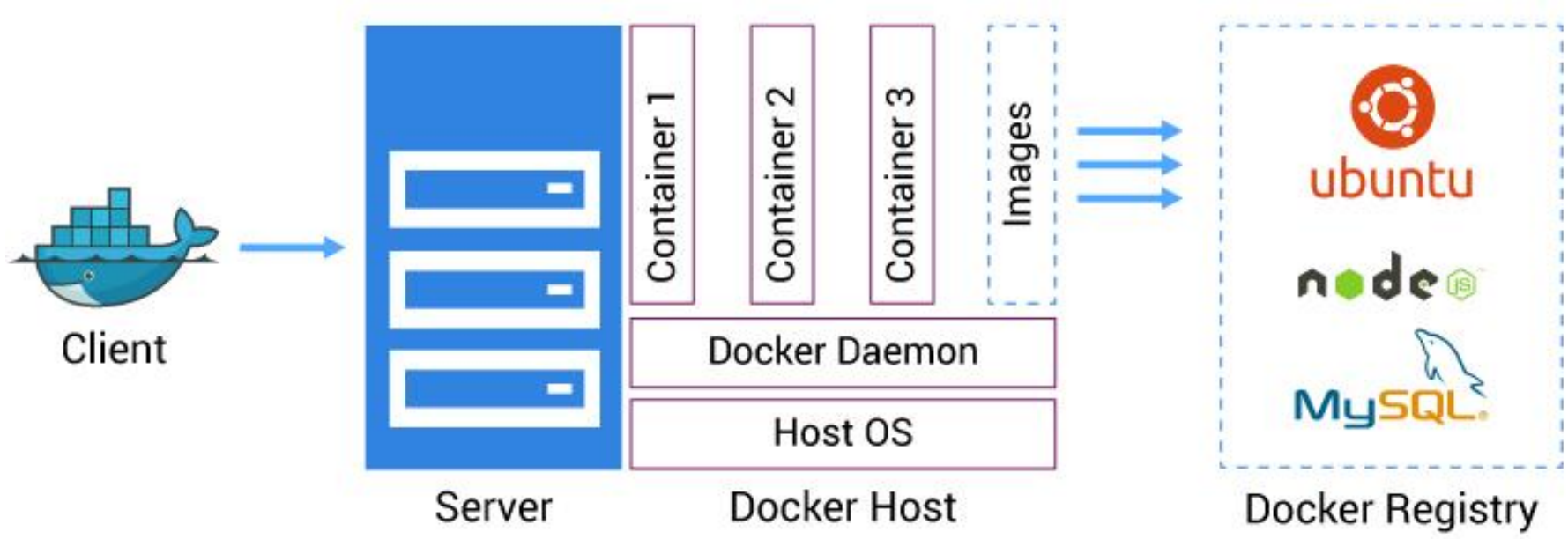
Docker带来的好处

- 秒级的交付和部署
- 保证环境一致性
- 高效的资源利用
- 弹性的伸缩
- 动态调度迁移成本低



Docker核心组件

- 一个完整的Docker有以下几部分组成：
  - 客户端（Docker Client）、守护进程（Docker Daemon）、镜像（Docker Image）、容器（Docker Container）、仓库（Docker Registry）



Docker核心组件-镜像

- 镜像（Docker Image）：Docker 镜像 是一个特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的一些配置参数（如匿名卷、环境变量、用户等）。镜像 不包含 任何动态数据，其内容在构建之后也不会被改变。
- 因为镜像包含操作系统完整的 root 文件系统，其体积往往是庞大的，因此在 Docker 设计时，就充分利用 Union FS 的技术，将其设计为分层存储的架构。所以严格来说，镜像并非是像一个 ISO 那样的打包文件，镜像只是一个虚拟的概念，其实际体现并非由一个文件组成，而是由一组文件系统组成，或者说，由多层文件系统联合组成。
- 镜像构建时，会一层层构建，前一层是后一层的基础。每一层构建完就不会再发生改变，后一层上的任何改变只发生在自己这一层。（详细请查阅dockerfile相关资料）

Docker核心组件-容器

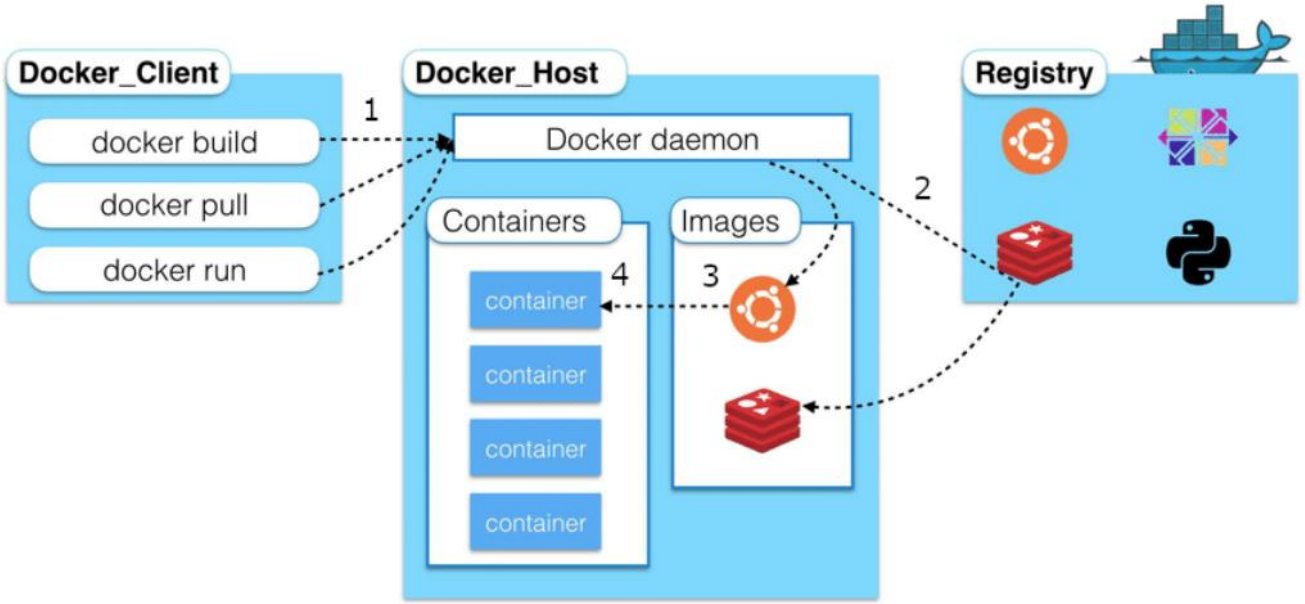
- 容器（Docker Container）：镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的 类 和 实例 一样，**镜像是静态的定义，容器是镜像运行时的实体**。容器可以被创建、启动、停止、删除、暂停等。
- 容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的 命名空间。因此容器可以拥有自己的 root 文件系统、自己的网络配置、自己的进程空间，甚至自己的用户 ID 空间。容器内的进程是运行在一个隔离的环境里，使用起来，就好像是在一个独立于宿主的系统下操作一样。

Docker核心组件-容器

- 每一个容器运行时，是以镜像为基础层，在其上创建一个当前容器的存储层，我们可以称这个为容器运行时读写而准备的存储层为 容器存储层。
- 容器存储层的生存周期和容器一样，容器消亡时，容器存储层也随之消亡。因此，任何保存于容器存储层的信息都会随容器删除而丢失。
- 按照 Docker 最佳实践的要求，**容器不应该向其存储层内写入任何数据，容器存储层要保持无状态化。**所有的文件写入操作，都应该使用 数据卷（Volume）、或者 绑定宿主目录，在这些位置的读写会跳过容器存储层，直接对宿主（或网络存储）发生读写，其性能和稳定性更高。
- 数据卷的生存周期独立于容器，容器消亡，数据卷不会消亡。因此，使用数据卷后，容器删除或者重新运行之后，数据却不会丢失。

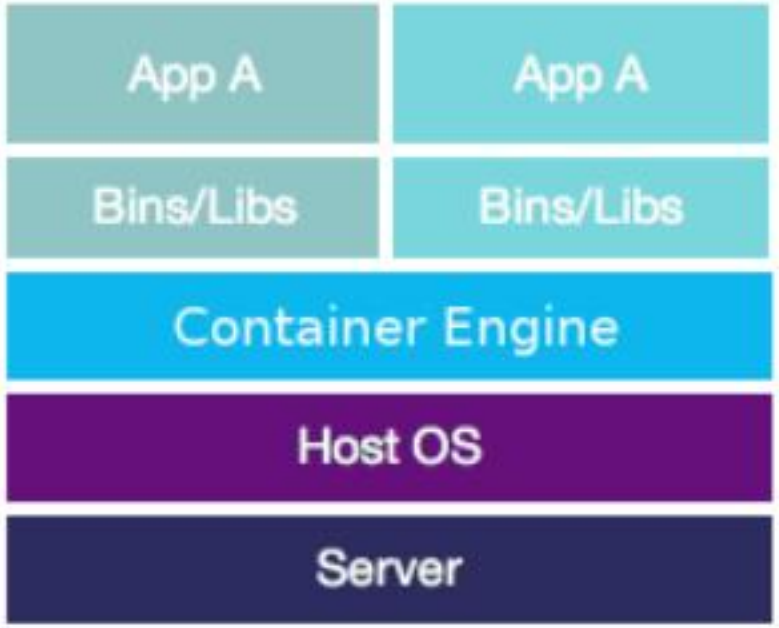
Docker核心组件

- Docker采用C/S架构，Docker Daemon作为服务端接受来自客户的请求，并处理这些请求（创建、运行、分发容器）。客户端和服务端既可以运行在一个机器上，也可通过socket或者RESTful API来进行通信。
- Docker daemon一般在宿主主机后台运行，等待来自客户端的消息。Docker客户端则为用户提供一系列可执行命令，用户用这些命令实现跟Docker daemon交互。



虚拟机与容器的对比

Virtual Machines



Containers

虚拟机与容器的对比

	容器技术	虚拟机技术
占用磁盘空间	小，甚至几十KB	非常大，上GB
启动速度	块，几秒钟	慢，几分钟
运行形态	直接运行在宿主机的内核上，不同容器共享同一个Linux内核	运行在Hypervisor上
并发性	一台宿主机可以启动成百上千个容器	最多几十个虚拟机
性能	接近宿主机本地进程	逊于宿主机
资源利用率	高	低

虚拟机与容器的对比

？ linux上可以托管windows容器吗

参考资料

强烈建议课下翻阅了解以下材料，深入了解docker命令和k8s 概念

https://www.docker.com/
https://yeasy.gitbook.io/docker_practice/
https://kubernetes.io/

深圳技术大学
SHENZHEN TECHNOLOGY UNIVERSITY
SZTU

Thank You!

感谢您的时间！