华为认证云计算系列教程

HCIA-Cloud Computing

华为云计算工程师认证

学习指导

配套版本: V4.0

华为技术有限公司

商标声明

 和其它华为商标均为华为技术有限公司的商标。

本文档提及的其它所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其它原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://e.huawei.com

目录

1 什么是云计算

1.1 云计算就在身边

2008 年，《钢铁侠 1》上映，开启了超级英雄类型电影的盛宴，超级英雄电影的代表——漫威，已经开始准备完成第一阶段的故事，《复仇者联盟 4》马上上映，我在这里蹭一下超级英雄的热度，使用超级英雄来介绍云计算。

在众多的超级英雄中，钢铁侠深受众多漫威粉喜爱，在《钢铁侠 3》中，钢铁侠去营救总统和小辣椒时，面对众多的变种人，他召唤出了钢铁军团，将电影推向了高潮，同时这也是漫威系列电影中钢铁军团第一次出现在大屏幕上，以后的电影中，我们更是见识到他召唤出各种型号的钢铁战衣。

我们来分析一下整个实现过程，首先，钢铁军团需要提前被造出来，这个在影片中处处有暗示，它们大多被存放在 Stark 大厦，还有的存放在别的地方，比如在《复仇者联盟 2》中，钢铁侠从太空中召唤来了“反浩克装甲”。第二步，使用某种技术打通钢铁侠和钢铁军团之间以及钢铁军团相互之间的通信，前者是为了将它们召唤出来，给小辣椒揉肩或者远程操作去救飞机上掉下来的人，再或者和绿巨人 PK，后者是为了能让钢铁军团之间互相配合，这个场景只有在《钢铁侠 3》中出现过一回。第三步，面对不同的需求，召唤出不同型号的战甲。钢铁侠战甲的升级换代一直被影迷们津津乐道，最初用来逃生的是 Mark1，拥有全新反应堆的是 Mark6，对付绿巨人时召唤的是 Mark44，在《复仇者联盟 3》中，我们更是见识到了最新的、用来和灭霸正面硬刚的 Mark50。

像这种提前将资源准备好，通过特定技术随时随地使用这些资源去执行特定任务的方式基本就属于云计算类型。

在电影中，使用钢铁战甲可以飞天遁地、惩奸除恶，在现实生活中呢？云计算能为我们提供什么样的服务呢？是不是也可以像钢铁战甲一样召之即来，挥之即去呢？打开华为公有云网站来看一下，如图 1-1。

图 1-1 华为公有云

在“产品”>“基础服务”中，我们可以看到计算、存储、网络、容器服务等等大的类型，大的类型下面又可以分为数量不等的小类型，我们看一个 HOT 的服务——弹性云服务器 ECS，如图 1-2：

图 1-2 弹性云服务器规格

我们会发现云服务器的规格和我们买电脑时的配置差不多，都会有 CPU、内存及硬盘的参数，后面也会给云服务器安装需要的操作系统，配置 IP 地址等等。其实，购买一台云服务器和购买一台电脑一样，云服务器就是把一台电脑放在云端，使用云服务器可以完成绝大部分在电脑上可以完成的工作，比如编辑文档、发送邮件或者协同办公等。除了这些，云服务器还具备一些电脑不具备的特性，比如，云服务器可以通过手机或者 Pad 使用，感觉和真实的电脑一样，甚至，我们可以随时修改自己云服务器的配置，比如把内存从 1G 扩容成 2G 等等。

除了云服务器，在华为公有云上，我们还可以购买到很多其它的服务，比如，需要网站，就可以购买云速建站服务，可以很快完成网站的搭建；需要硬盘来存储数据，就可以购买对象存储服务或者云硬盘。往更高的层次讲，在云计算中，我们可以完成人脸识别、语音识别、图像识别或者文字识别等 AI 方面的功能。

总之，云计算可以让我们像使用水电一样使用 IT 服务。大家想一下我们平常是如何使用水电的？是不是打开水龙头、打开开关就可以使用了？那是因为水和电都已经连在了水网电网上。以此类推，水网电网放到云计算中就是互联网，IT 服务就是我们要使用的水电，而水龙头和开关可以是网页界面，也可以是某一个 APP。

说了这么多，您可能有这样的疑问，我有自己的电脑，需要的大多数应用都已经被安装在本地硬盘中，我也不会用到什么人脸识别、语音识别等服务，IT 服务又不是水电，离开了它我也能正常地生活，云计算和我没有多大关系。其实不然，您可能已经在使用云计算了。

前面我们讲过，云计算的水龙头和开关可能是网页，也可能是某个 APP，那么在这里，我列举了几个最常用的 APP——百度网盘、华为手机的自动备份、有道笔记和网易云音乐。

百度网盘，相信大家都使用过，百度把它定义成一项云存储服务，通过百度网盘的客户端，我们可以轻松地将照片、视频、文档等文件进行备份、同步和分享。其中“备份”、“同步”、“分享”就属于 IT 服务，如果没有云计算，我们要实现这些服务就需要手动将文件拷贝到其它的硬盘上，然后再通过这个硬盘分享给别人或者恢复数据。而现在，我们通过客户端，无论它是被安装在手机上还是 PC 上，只要连接互联网并指定需要备份的文件夹，数据就会被自动上传到共享

资源池中来代替硬盘，这种模式就是云计算。使用了云计算后，资源池是共享的，所以通过分享，其它人很容易就可以将这些数据下载下来，另外，通过一定的技术手段，还可以自动进行数据同步。

图 1-3 云计算类 APP

百度网盘需要我们自己安装，而华为手机的备份与恢复服务是出厂自带的，其它品牌的手机也有类似的服务，比如 IPHONE 的 iCLOUD，它们都可以将手机端的文件备份到远端的数据中心，更换手机后，使用自己的账号密码就可以将自己的数据还原到新手机上。

人生总是处处有惊喜，时时有感悟，某个时刻灵感大爆发需要记下来时，却发现身边没有纸和笔，这时你就需要有道云笔记，它是网易推出的一款产品，提供了文档在线创建和编辑的功能，获得了 5000 万用户的青睐。文档的创建和编辑也是我们非常常用的 IT 服务，使用有道云笔记，就相当于使用由云计算提供的服务，无论是 PC 端、移动端还是网页端，用户都可以随时随地对线上资料进行编辑、分享以及协同，并且每次编辑都可以立刻同步到云端。

最后一款网易云音乐，使用该 APP，所有的歌曲都可以在线收听，随时打开随时播放。

1.2 云计算的特点

从上面的例子中我们发现，无论是钢铁侠——万众瞩目的超级英雄，还是我们自己，一个代表未来人群，一个代表当下人群，都不约而同地选择了云计算，那是因为云计算自身特点的优势符合他们的需求，接下来我们先了解一下云计算的特点。

1.2.1 按需自助服务

说到按需自助我们最先想到的就是超市，每个顾客在超市里都可以按照自己的需求挑选需要的商品，如果是同类商品，可以自己查看说明、价格、品牌来确定是按照性价比或者其它来决定购买哪一款。按需自助也是云计算的特点之一，我们在《钢铁侠》系列中经常看到类似的情节，首先，钢铁侠在更换战甲时完全是自助，基本不需要别人的帮助，他也不会让别人操纵。另外一个就是按需，这个在《钢铁侠》系列中体现最明显，战甲可以从手表里变形出来，也可以从手提箱或者从直升飞机的座位中变形出来，钢铁侠可以按照自己的需求，把它从任何地方召唤出来，甚至是自己的皮肤底下。

回到现实中，我们根据自己的需求下载不同 APP 或者在华为公有云网站上购买需要的服务，在整个下载或者购买过程中，基本不需要他人帮忙，我们可以自行完成下载或者购买。

图 1-4 推荐云服务

按需自助的前提是了解自己的需求，并知道哪款产品能够解决这个需求。在超市中，有很多的商品，在云计算提供商的官网上也一样有很多类型的产品，如上图，在购买以前，用户需要购买哪种产品，就像去超市以前，顾客需要知道洗衣液是用来洗衣服的，咖啡是用来喝的一样。

1.2.2 广泛的网络接入

关于云计算的第二个特点，我们还是从漫威的电影说起。在《复仇者联盟 2》中，大反派奥创是钢铁侠和绿巨人不小心创造出来的，然后它抢走了心灵宝石、安排绯红女巫激怒绿巨人等等，并多次从超级英雄们的围堵追缴中逃脱，直到幻视被创造出来，把它的网络封掉才被打败。电影中多次提到，奥创的逃脱依赖的是网络，它可以通过网络将自己转移到不同的载体中，只要有网络，奥创就永远在线，永远不会被杀死，所以它一觉醒就破坏掉能把自己困在防火墙以内的贾维斯。

云计算，通俗的讲，就是互联网加计算，所以网络接入是云计算自带的属性。

在当今社会，互联网几乎可以覆盖到全球每一个有人的角落，我们可以通过任意电子设备——PC、Pad、手机等连接到网络中，这也就意味着通过任何电子设备都可以使用云计算，这点要比奥创先进，奥创只能将自己传送到机器人上，如果那些机器人全部报废了，奥创也就无处可逃了，而云计算的载体则更加广泛，在办公室可以使用 PC，在机场车站可以使用手机或者 Pad，没有网线可以用 WiFi 代替，没有 WiFi，用流量也没有问题，总之，可以接入网络的地方，就有云计算。

如图 1-5 和图 1-6 显示的是使用 PC 和手机端登录华为云使用相同的 EI 服务。

图 1-5 电脑端华为云 EI 体验服务

图 1-6 手机端华为云 EI 体验服务

1.2.3 资源池化

资源池化是实现按需自助服务的前提之一，通过资源池化不但可以把同类商品放在一起，而且还能将商品的单位进行细化。在超市里，我们会看到生鲜区、果蔬区以及其它，这样可以方便

顾客快速地找到自己需要的商品，但这种形式不是资源池化，只能算是资源归类，那什么算是资源池化呢?

资源池化除了将同类的资源转换为资源池的形式外，还需要将所有的资源分解到最小单位。

方便面，算是很多人生活必需品之一，同时也有很多人反映一包吃不饱，两包吃不完，这是因为超市中方便面的最小购买单位是“包”，如果使用资源池化的方式，就需要打破“包”这个单位，将所有的面放在一个“池子”里，需要多少买多少。比如自助餐厅就是这样做的，将果汁按照不同的口味分开，客户需要多少就打多少。

资源池化还有一个作用就是可以屏蔽不同资源的差异性，如果餐厅中提供池化了的可乐，里面装的是百事可乐还是可口可乐或者两者都有，顾客是看不出来的。而在云计算中，可以被池化的资源包括计算、存储和网络等资源，计算资源包括 CPU 和内存，如果对 CPU 进行池化，用户端看到的 CPU 最小单位可以是一个核心，而不再体现 CPU 的厂商是 AMD 或者 Intel，具体如何实现，我们在《计算虚拟化》章节中介绍。

1.2.4 快速弹性伸缩

我们前面在讲按需自助服务时提到，为了应对热点事件的突发大流量，自助购买服务器进行扩容。而当热点事件降温后，访问流量趋于下降时，又可以将这些服务器释放进行减容，这种行为就属于典型的快速弹性伸缩。

快速弹性伸缩包括多种类型，除了人为手动扩容或减容，云计算还支持根据预设的策略进行自动扩容或减容，伸缩可以是增加或减少服务器数量，也可以是对单台服务器进行资源的增加或减少。

我们熟悉的具备这个特性最典型的例子就是儿时的偶像——孙悟空的兵器金箍棒，可大可小，第一次出场时，“乃是一根铁柱子，约有斗来粗，二丈有余长”，在孙悟空的默念下，一再地变小变细，“拿出外面，只有丈二长短，碗口粗细”，在后来孙悟空的炫耀中，“手中那棒，上抵

三十三天，下至十八层地狱”，而孙悟空拔根毫毛变出千万个分身的时候，金箍棒也会变成千万个。

在云计算中，快速弹性伸缩对用户来说，最大的好处是在保证业务或者应用稳定运行的前提下节省成本。企业在创立初期，可以购买少量的资源，随着企业规模的扩大，可以逐步增加资源方面的投资；另一方面，在特殊时期可以将所有资源集中供给重点业务使用，非特殊时期，将空闲资源转做它用；如果特殊时期资源不足，可以临时增加，度过特殊时期后，再将增加的资源释放。无论是哪种情景，对于用户来说都是很方便的。

1.2.5 可计量服务

首先，计量不是计费，尽管通过计量可以进行计费。

在云计算中，大部分服务都需要付费使用，但也有服务是免费的，比如，弹性伸缩可以作为一个服务为用户开通，大部分时间这个服务是免费的。

计量是利用技术和其它手段实现单位统一和量值准确可靠的测量，换句话说，云计算中的服务都是可测量的，有的是根据时间，有的是根据资源配额，还有的是根据流量。服务可测量可以准确地根据客户的业务进行自动控制和优化资源配置。

对于用户来说可以很清晰地看到自己购买服务的使用情况，还可以根据需求来购买相应数量的服务。我们还是用孙悟空的金箍棒来举例，在原著和影视作品中，我们可以看到金箍棒能根据孙悟空的需求变大变小，但是具体变到多大，缩到多小都是随心。如果金箍棒是云计算模式，面对白骨精，可以让它变成 3 米长，1 米粗，并且保持这个状态 30 分钟，面对小一点的妖怪可以小一些，2 米长，0.7 米粗，保持 15 分钟。如果不用的时候，缩小到 1 厘米长，0.1 厘米粗好放入到耳中。

如图 1-7 和图 1-8 体现的就是云计算中服务的计量情况。

比如弹性云服务器，可以是 1 个 vCPU，1GB 内存，40G 硬盘，使用时间为 1 个月。

图 1-7 ECS 计量方式

如果是云速建站服务，计量方式如下：

图 1-8 云速建站计量方式

1.3 云计算的定义

现阶段对云计算的定义有多种说法，对于到底什么是云计算，至少可以找到 100 种解释。广为接受的说法是美国国家标准与技术研究院（NTSI）定义：云计算是一种模型，它可以实现随时随地、便捷地、随需应变地从可配置计算资源共享池中获取所需的资源（例如，网络、服务器、存储、应用及服务），资源能够快速供应并释放，使管理资源的工作量和与服务提供商的交互减小到最低限度。

说明一下定义中的重点：

1、云计算不是技术，而是一种模型；

2、通过云计算，用户可以使用的资源包括网络、服务器、存储、应用及服务等，这些资源全部属于 IT 领域，云计算的目标就是让大众像获取水电一样获取到 IT 服务；

3、资源的使用可以随时随地，前面我们讲过云计算的特点，“随时随地”的前提是网络可达；

4、资源能够快速供应并释放对应了云计算快速弹性伸缩的特点，而与服务提供商的交互减小到最低限度对应了按需自助服务。

通俗的讲，云，是网络、互联网的一种比喻说法，即互联网与建立互联网所需要的底层基础设施的抽象体。“计算”指的是一台足够强大的计算机提供的计算服务（包括各种功能，资源，存储）。“云计算”可以理解为：通过互联网可以使用足够强大的计算机为用户提供的服务，这种服务的使用量可以使用统一的单位来描述。

注：“云”代表互联网是来自数通领域，在画拓扑图的时候，工程师们会使用云图形代表“拿来就可以使用，但是不必关心其内部技术细节”的网络，最后慢慢演变为拿云图形来特指互联网。

既然云计算是由互联网和计算组成，那么云计算的发展史就是由互联网发展史和计算模式发展史组成的，下一章我们将介绍云计算是怎么发展起来的，在这之前，先分享一个因互联网技术没有成熟而导致云计算尝试失败的案例。

拉里·埃里森，ORACLE 的创始人，IT 界的传奇人物，如果感兴趣可以在网上搜一下他的事迹。ORACLE 成立的时候，美国另外两家传奇的公司也成立了，一个是苹果，一个是微软，给大众的感觉，拉里·埃里森的风头总是稍逊比尔盖茨。最开始的时候，微软的主打产品是操作系统，ORACLE 的主打产品是数据库，但是在 1988 年微软也推出了数据库——SQL Server，这不是赤裸裸的抢地盘吗？所以拉里·埃里森推出了一款不需要安装操作系统的互联网电脑作为反击，互联网电脑没有本地硬盘，操作系统、用户数据和程序都放在远端数据中心的服务器中，价格上也很有优势。但是，有一点拉里·埃里森没有考虑到，当时推出这款产品的时间是 1995 年，当时互联网还没有普及，也没有足够大的带宽，网速很慢，几乎无法支持在线操作，用户体验很不好，所以在两年后草草收场。

互联网电脑可以算是云计算的雏形，然而这个概念太超前了，再加上 2000 年左右互联网泡沫的破裂，影响了人们对云端应用的信心，直到 2006 年亚马逊推出了 AWS（后面再详细介绍）。

1.4 云计算的起源和发展

前面在云计算的定义中讲过，云计算相当于互联网加计算模型，云计算的发展史就是互联网和计算模型的发展史，那么接下来，我们来简单介绍一下它们的发展史，首先是互联网的发展史。

1.4.1 互联网发展史

早期计算机都是单机运行，数据的计算和传输都是在本机完成，互联网的诞生将这些计算机连接起来，最终将整个世界都连接起来，下面是一些在互联网发展过程中非常有代表意义的里程碑事件。

1969 年，ARPANET 诞生，它被认为是互联网的前身。很多技术的诞生和发展都是因为战争，互联网也一样，据说 ARPANET 的诞生是因为美国担心其它国家的核武器会摧毁美国的核心军事基地，这样整个国家的军事防御能力就会被摧毁，所以美国国防部需要建立一个高容错的网络系统，最早加入 ARPANET 的节点只有四个，分别是位于美国中部的四个大学：加州大学洛杉矶分校（UCLA），斯坦福研究所（SRI），加州大学圣芭芭拉分校（UC Santa Barbara）和犹他大学（University of Utah）。ARPANET 的诞生，标注着互联网时代的开始。随后的几年，加入 ARPANET 的节点越来越多，同时非军事领域的用户也越来越多，1983 年，出于安全的考虑，ARPANET 将其中的 45 个节点分离出去，形成了一个专门的军事网络，叫 MILNET，剩余的节点用于民用。

1981 年，首个 TCP/IP 协议的完整规范建立，互联网沟通语言诞生。为什么需要 TCP/IP 协议？TCP/IP 协议其实是一个协议集合，它包括了 TCP 协议（Transport Control Protocol，传输控制协议），IP 协议（Internet Protocol，Internet 协议）及其它一些协议。最早在 ARPANET 上用的协议叫 NCP（Network Control Protocol，网络控制协议），但是随着 ARPANET 的壮大，NCP 无法满足大型网络的需求，而 TCP/IP 生来就是为大型甚至巨型网络服务的，所以很快，在 1983 年 1 月 1 日，ARPANET 就将 NCP 代替为 TCP/IP。

1983 年，ARPANET、PRNET 和 SATNET 三个原始网络采用 TCP/IP 通讯。最早的三大网络同时切换到 TCP/IP，这标志着互联网开始快速发展。

1984 年，DNS 诞生。TCP/IP 被采用以后，互联网的发展变得更加迅速，加入网络中的计算机越来越多，每一个计算机都采用 TCP/IP 标准的数字化的 IP 地址来标识彼此。计算机的数量越来越多，IP 地址就越来越多，而且还不好记，这就好比使用身份证号来代表我们身边所有的人，彼此沟通的时候先用一长串数字叫对方，可是我们根本记不住这么多这么长的数字。所以，仅使用 IP 地址来表示某台计算机对系统维护的工作人员来说不是什么好事，因为工作变得越来越困难，所以开发一个新的架构或者技术来解决这些问题的呼声也越来越高，于是，DNS 应运而生。DNS 可以将数字化的 IP 地址和更容易让人记住的域名互相转换，相当于使用简短易记的名字来代替身份证号码，这就大大降低了我们沟通的难度。最初的域名包括两部分：名称，比如 HUAWEI，和分类或目的，比如代表 commercial 的 “.com”。这样，维护人员只要输入 HUAWEI.com 就能找到对应的 IP 地址的计算机。发展到后来，通过 DNS，我们可以在全球范围内使用域名来访问对应的主页等。

1986 年，现代邮件路由系统 MERS 开发完成。

1989 年，第一个商用网络运营商 PSINet 成立。在 PSINet 创建之前，大多数网络都是由政府或者军方出资用来在军事、工业或者科学方面做研究，PSINet 的成立，代表着互联网进入了商业化运作的时代。

1990 年，首个网络搜索引擎 Archie 出现。在互联网发展早期，网络上的信息较少，所以查找起来也比较容易，随着互联网的发展，互联网上的信息量爆炸式增长，用户想从中找到自己需要的信息无异于大海捞针，所以需要一个搜索引擎或者搜索网站来索引和查找。Archie 就是最早的网络搜索引擎，当时网络中的文件传输很多，然而各个文件都散落在各个 FTP 服务器上，查询起来非常不方便，于是 Archie 被开发出来，开发者叫做 Alan Emtage——一个来自蒙特利尔大学的学生，使用 Archie 可以使用文件名称来查找文件。

1991 年，www 正式向公众开放。看到这里大家不要惊讶，我们现在每天都在用的 www 是在 1991 年才被开放的，也就是才不到 30 年。www，World Wide Web，万维网，现在简称 web，由欧洲粒子研究中心的科学家 Tim Berners-Lee 发明，它是互联网发展史上里程碑式的技术，它使超媒体可以在网络上传播和互联，超媒体可以是文档，也可以是语音或者视频，是一种全新的信息的表达方式。www 诞生后，所有伟大的、里程碑式的互联网公司随之诞生，并且真正改变普通人生活的各类网络应用才开始不断涌现。

1995 年，亚马逊、eBay 等电商成立。在互联网发展史上，出现了很多伟大的公司，比如雅虎、谷歌等。这里单独说一下亚马逊，因为它是第一个真正意义上实现云计算的互联网公司。亚马逊早期出售的商品是图书，为了处理商品信息和用户资料，亚马逊建立了庞大的数据中心。在美国，有类似淘宝“双十一”的“黑色星期五”，在这一天，亚马逊需要处理大量的信息，数据中心的所有设备会全部开启，但是过了这一天，大量的设备会处于闲置状态，为了不造成浪费，亚马逊就将多余的资源租出去，于是在 2006 年推出了首款云计算产品——EC2（Elastic Compute Cloud）。

注：亚马逊、谷歌、阿里巴巴、腾讯等很多我们耳熟能详的公司都属于互联网企业，另外像 IBM、思科、华为、联想等公司属于传统 IT 企业。

2000 年，互联网泡沫破裂。互联网让人们见识到了它的神奇，导致发展过快，催生了大量的泡沫，终于在 2000 年前后，互联网泡沫破裂，我们前面提到的 PSINet 就是在这个时期破产倒闭的。

再后来经历了泡沫破裂的互联网又迅速得到了发展，2004 年，Facebook 成立，当年也被称为社交网络元年。

1.4.2 计算发展史

1.4.2.1 并行计算

传统上，一般的软件设计都是串行式计算，具体如下：

1、一个“problem”被划分成一串离散的“Instructions”；

2、“Instructions”会在单个 CPU 上一个一个被执行；

3 、CPU 在同一时间只能处理一个“Instruction”。

图 1-9 串行计算示意图

一个 Problem 只采用串行计算的话，如果 Problem 太复杂，处理的时间会较长。如果解决更大规模的问题，尤其是当计算机的内存受到限制的时候，用单个 CPU 来解决是不切实际或者根本不可能的，例如，网络搜索引擎和网络数据库每秒要处理上百万次的计算，用串行计算基本不大可能。

在理论和实际上，想要轻易地制造更快的串行计算机存在着巨大的限制：

1、传输速度——线性计算机的执行速度直接取决于数据在硬件中传输的速度。光速的绝对限制是每纳秒 30cm，铜导线是每纳秒 9cm。不断提升的执行速度更加靠近极限。

2、微型化的极限——处理器技术使芯片集成了更多的晶体管。但是，即使使用分子或者原子级别的组件也会很快达到芯片集成晶体管的极限。

3、经济上的限制——让单个芯片变得更快需要增加昂贵的投入。用多个一般的芯片来取代单个高性能的芯片或许性能会更好而且更便宜。

既然难以提高单个 CPU 的计算性能，那么我们就让多个 CPU 同时参与到任务的执行中，也就是在最简单的情形下，使用并行计算来解决串行计算的限制：

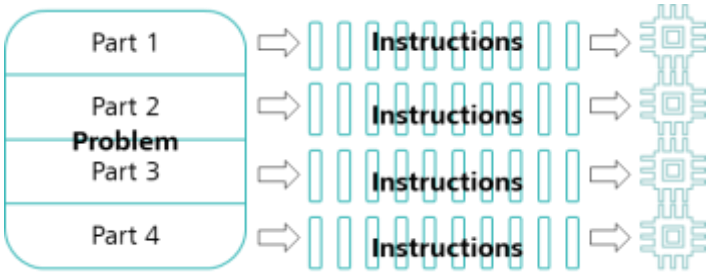


图 1-10 并行计算示意图

1、先把一个大的 Problem 分解成多个可被同时处理的 Part；

2、再把每个 Part 划分成一串离散的 Instructions；

3、每个 Part 把自己的 Instructions 交给各自的 CPU 进行处理，每个 CPU 同时处理不同 Part 的 Instruction；

4、再加入一个统一控制机制对整个过程进行控制。

在历史上，并行计算被认为是高端计算，用于为复杂的科学计算和基于真实世界的工程问题建模。今天，商务应用是推动计算机快速发展的更大的推动力。这些应用需要用复杂的方法处理大量数据。

使用并行计算的原因有以下几点：

1、节省时间和成本：理论上，使用更多的资源会使一个任务提前完成，而且会节约潜在的成本。况且可以使用便宜的、甚至市面将要淘汰的 CPU 来构建并行聚簇。

2、解决我们前面提到过的使用串行计算无法解决的问题。

并行计算的 CPU 可以来自同一计算机，也可以来自同一个网络中的不同计算机。

1.4.2.2 分布式计算

分布式计算属于研究分布式系统的计算机科学领域。分布式系统，是将自己的所有组件分散在属于不同网络的计算机上，这些计算机通过统一的消息机制来相互通讯和配合。分布在不同网络计算机上的组件互相协作，完成共同的目标。

分布式计算比起其它算法具有以下几个优点：

1、稀有资源可以共享。

2、通过分布式计算可以在多台计算机上平衡计算负载。

3、可以把程序放在最适合运行它的计算机上。

其中，共享稀有资源和平衡负载是计算机分布式计算的核心思想之一。

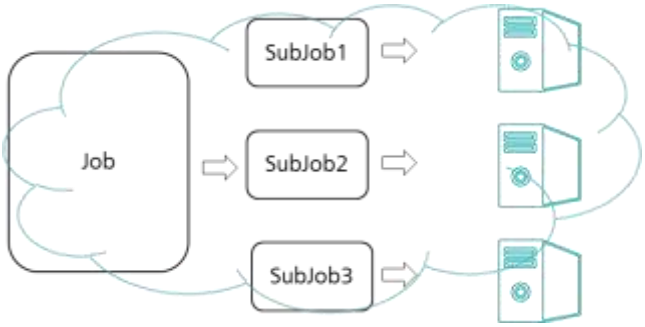


图 1-11 分布式计算示意图

并行计算与分布式计算都是通过运用并行的方式来获得更高性能，化大任务为小任务。简单来说，如果处理单元共享内存，就称为并行计算，反之就是分布式计算。也有人认为分布式计算是并行计算的一种特例。

其实分布式的任务包相互之间有独立性，上一个任务包的结果未返回或者处理结果错误，对下一个任务包的处理几乎没有什么影响。因此，分布式的实时性要求不高，而且允许存在计算错误（因为每个计算任务给好几个参与者计算，结果上传到服务器后要比较，然后对结果差异大的进行验证）。

而并行程序并行处理的任务包相互之间有很大的联系，并且并行计算的每一个任务块都是必要的，没有浪费的分割的，就是每个任务包都要处理，而且计算结果相互影响，这就要求每个计算结果要绝对正确，而且在时间上要尽量做到同步，而分布式的很多任务块可以不用处理，比如大量的无用数据块，所以说分布式计算的速度尽管很快，但是真正的“效率”是低之又低 的，可能一直在寻找，但是永远都找不到，也可能一开始就找到了；而并行处理不同，它的任务包个数相对有限，在一个有限的时间应该是可以完成的。

1.4.2.3 网格计算

网格计算是利用广泛的零散的计算资源完成一个共同任务，它也是分布式计算的一种。根据IBM 对“网格”的定义，它将本地网络或者互联网上零散的可用计算资源集合起来，使终端用户或者应用觉得他们在使用一台性能强悍的虚拟计算机。网格计算的愿景是创立一个虚拟动态的资源集合，使个人和组织机构能够安全协调的使用这些资源。网格计算通常使用集群的方式实现。

1.4.2.4 云计算

云计算（Cloud Computing），是一种新兴的共享基础架构的方法，可以将巨大的系统池连接在一起以提供各种 IT 服务。很多因素推动了对这类环境的需求，其中包括连接设备、实时数据流、SOA 的采用以及搜索、开放协作、社会网络和移动商务等这样的 Web2.0 应用的急剧增长。另外，数字元器件性能的提升也使 IT 环境的规模大幅度提高，从而进一步加强了对一个由统一的云进行管理的需求。云计算被它的吹捧者视为“革命性的计算模型”，因为它使得超级计算能力通过互联网自由流通成为了可能。企业与个人用户无需再投入昂贵的硬件购置成本，只需要通过互联网来购买租赁计算力，“把你的计算机当做接入口，一切都交给互联网吧”。

狭义云计算是指 IT 基础设施的交付和使用模式，指通过网络以按需、易扩展的方式获得所需的资源（硬件、平台、软件）。 提供资源的网络被称为“云”。“云”中的资源在使用者看来是可以无限扩展的，并且可以随时获取，按需使用，随时扩展，按使用付费。这种特性经常被称为像水电一样使用 IT 基础设施。

广义云计算是指服务的交付和使用模式，指通过网络以按需、易扩展的方式获得所需的服务。这种服务可以是 IT、软件和互联网相关的，也可以是其它任意的服务。

云计算一般会具有以下相同点：

（1） 超大规模。“云”具有相当的规模，Google 云计算已经拥有 100 多万台服务器，Amazon、IBM、微软、Yahoo 等的“云”均拥有几十万台服务器。企业私有云一般拥有数百上千台服务器。“云”能赋予用户前所未有的计算能力。

（2） 虚拟化。云计算支持用户在任意位置、使用各种终端获取应用服务。所请求的资源来自“云”，而不是固定的有形的实体。应用在“云”中某处运行，但实际上用户无需了解、也不用关心应用运行的具体位置。只需要一台笔记本或者一个手机，就可以通过网络服务来实现我们需要的一切，甚至包括超级计算这样的任务。

（3） 高可靠性。“云”使用了数据多副本容错、计算节点同构可互换等措施来保障服务的高可靠性，使用云计算比使用本地计算机可靠。

（4） 通用性。云计算不针对特定的应用，在“云”的支撑下可以构造出千变万化的应用，同一个“云”可以同时支撑不同的应用运行。

（5） 高可扩展性。“云”的规模可以动态伸缩，满足应用和用户规模增长的需要。

（6） 按需服务。“云”是一个庞大的资源池，你按需购买；云可以像自来水、电、煤气那样计费。

（7） 极其廉价。由于“云”的特殊容错措施可以采用极其廉价的节点来构成“云”，“云”的自动化集中式管理使大量企业无需负担日益高昂的数据中心管理成本，“云”的通用性使资源的利用率较之传统系统大幅提升，因此用户可以充分享受“云”的低成本优势，经常只要花费几百美元、几天时间就能完成以前需要数万美元、数月时间才能完成的任务。

1.4.3 云计算的发展

云计算理念从最初诞生至今，企业 IT 架构从传统非云架构，向目标云化架构的演进，总结来说，经历了如下三大里程碑发展阶段：

1、云计算 1.0：面向数据中心管理员的 IT 基础设施资源虚拟化阶段。该阶段的关键特征体现为通过计算虚拟化技术的引入，将企业 IT 应用与底层的基础设施彻底分离解耦，将多个企业 IT 应用实例及运行环境（客户机操作系统）复用在相同的物理服务器上，并通过虚拟化集群调度软件，将更多的 IT 应用复用在更少的服务器节点上，从而实现资源利用效率的提升。HCIA-Cloud Computing 重点关注这个阶段，将介绍该阶段的云计算如何实现及优势。

2、云计算 2.0：面向基础设施云租户和云用户的资源服务化与管理自动化阶段。该阶段的关键特征体现为通过管理平面的基础设施标准化服务与资源调度自动化软件的引入，以及数据平面的软件定义存储和软件定义网络技术，面向内部和外部的租户，将原本需要通过数据中心管理员人工干预的基础设施资源复杂低效的申请、释放与配置过程，转变为在必要的限定条件下（比如资源配额、权限审批等）的一键式全自动化资源发放服务过程。这个转变大幅提升了企业 IT 应用所需的基础设施资源的快速敏捷发放能力，缩短了企业 IT 应用上线所需的基础设施资源准

备周期，将企业基础设施的静态滚动规划转变为动态资源的弹性按需供给过程。这个转变同时为企业 IT 支撑其核心业务走向敏捷，更好地应对瞬息万变的企业业务竞争与发展环境奠定了基础。云计算 2.0 阶段面向云租户的基础设施资源服务供给，可以是虚拟机形式，可以是容器（轻量化虚拟机）形式，也可以是物理机形式。该阶段的企业 IT 云化演进，暂时还不涉及基础设施层之上的企业 IT 应用与中间件、数据库软件架构的变化。

3、云计算 3.0：面向企业 IT 应用开发者及管理维护者的企业应用架构的分布式微服务化和企业数据架构的互联网化重构及大数据智能化阶段。该阶段的关键特征体现为企业 IT 自身的应用架构逐步从（依托于传统商业数据库和中间件商业套件，为每个业务应用领域专门设计的、烟囱式的、高复杂度的、有状态的、规模庞大的）纵向扩展应用分层架构体系，走向（依托开源增强的、跨不同业务应用领域高度共享的）数据库、中间件平台服务层以及（功能更加轻量化解耦、数据与应用逻辑彻底分离的）分布式无状态化架构，从而使得企业 IT 在支撑企业业务敏捷化、智能化以及资源利用效率提升方面迈上一个新的高度和台阶，并为企业创新业务的快速迭代开发铺平了道路。

针对上述三大云计算发展演进里程碑阶段而言，云计算 1.0 普遍已经是过去式，且一部分行业、企业客户已完成初步规模的云计算 2.0 建设商用，正在考虑该阶段的进一步扩容，以及面向云计算 3.0 的演进；而另一部分客户则正在从云计算 1.0 走向云计算 2.0，甚至同步展开云计算 2.0 和 3.0 的演进评估与实施。

扩展知识，云计算 1.0 与 2.0/3.0 阶段之间的主要差异体现为如下七点：

注：以下内容来自《云计算架构技术与实践》。

差别 1：从 IT 非关键应用走向电信网络应用和企业关键应用。从云计算面向企业 IT 及电信网络的使用范围的视角来看，云计算发展初期，虚拟化技术主要局限于非关键应用，比如办公桌面云，开发测试云等。该阶段的应用往往对底层虚拟化带来的性能开销并不敏感。人们更加关注于资源池规模化集中之后资源利用效率的提升以及业务部署效率的提升。然而随着云计算的持续深入普及，企业 IT 云化的范围已从周边软件应用，逐步走向更加关键的企业应用，甚至企业的

核心生产 IT 系统。因此，如何确保云平台更高效、更可靠地支撑时延敏感的企业关键应用，就变得至关重要。对于企业 IT 基础设施的核心资产而言，除了实实在在的计算、存储、网络资源等有形物理资产之外，最有价值的莫过于企业数据这些无形资产。在云计算的计算虚拟化技术发展初期阶段，Guest OS 与 Host OS 之间的前后端 I/O 队列在 I/O 吞吐上的开销较大，而传统的结构化数据由于对 I/O 性能吞吐和时延要求很高，这两个原因导致很多事务关键型结构化数据在云化的初期阶段并未纳入虚拟化改造的范畴，从而使得相关结构化数据的基础设施仍处于虚拟化乃至云计算资源池的管理范围之外。然而随着虚拟化 XEN/KVM 引擎在 I/O 性能上的不断优化提升（如采用 SR-IOV 直通、多队列优化技术），使得处于企业核心应用的 ERP 等关系型关键数据库迁移到虚拟化平台上部署和运行已不是问题。与此同时，云计算在最近 2～3 年内，已从概念发源地的互联网 IT 领域，渗透到电信运营商网络领域。互联网商业和技术模式的成功，启发电信运营商们通过引入云计算实现对现有电信网络和网元的重构来打破传统意义上电信厂家所采用的电信软件与电信硬件紧绑定的销售模式，同样享受到云计算为 IT 领域带来的红利，诸如：硬件 TCO 的降低，绿色节能，业务创新和部署效率的提升，对多国多子网的电信功能的快速软件定制化以及更强的对外能力开放。

差别 2：从计算虚拟化走向存储虚拟化和网络虚拟化，从支撑云计算按需、弹性分配资源，与硬件解耦的虚拟化技术的角度来看，云计算早期阶段主要聚焦在计算虚拟化领域。事实上，众所周知的计算虚拟化技术早在 IBM 370 时代就已经在其大型机操作系统上诞生了。技术原理是通过在 OS 与裸机硬件之间插入虚拟化层，在裸机硬件指令系统之上仿真模拟出多个 370 大型机的“运行环境”，使得上层“误认为”自己运行在一个独占系统之上，而实际上是由计算虚拟化引擎在多个虚拟机之间进行 CPU 分时调度，同时对内存、I/O、网络等也进行访问屏蔽。只不过后来当 x86 平台演进成为 IT 领域硬件平台的主流之后，VMware ESX、XEN、KVM 等依托于单机 OS 的计算虚拟化技术才将 IBM 370 的虚拟化机制在 x86 服务器的硬件体系架构下实现和商品化，并且在单机/单服务器虚拟化的基础上引入了具备虚拟机动态迁移和 HA 调度能力的中小集群管理软件（如 vCenter/vSphere、XEN Center、FusionSphere 等），从而形成当前的计算虚拟化主体。随着数据和信息越来越成为企业 IT 中最为核心的资产，作为数据信息持久化载体

的存储已经逐步从服务器计算中剥离出来成为了一个庞大的独立产业，与必不可少的 CPU 计算能力一样，在数据中心发挥着至关重要的作用。当企业对存储的需求发生变化时该如何快速满足新的需求以及如何利用好已有的多厂家的存储，这些问题都需要存储虚拟化技术来解决。与此同时，现代企业数据中心的 IT 硬件的主体已经不再是封闭的、主从式架构的大小型机一统天下的时代。客户端与服务器之间南北方向的通信、服务器与服务器之间东西方向的协作通信以及从企业内部网络访问远程网络和公众网络的通信均已走入了基于对等、开放为主要特征的以太网互联和广域网互联时代。因此，网络也成为计算、存储之后，数据中心 IT 基础设施中不可或缺的“三要素”之一。就企业数据中心端到端基础设施解决方案而言，服务器计算虚拟化已经远远不能满足用户在企业数据中心对按需分配资源、弹性分配资源与硬件解耦的分配资源的能力需求，由此存储虚拟化和网络虚拟化技术应运而生。除了云管理和调度所完成的管理控制面的 API 与信息模型归一化处理之外，虚拟化的重要特征是通过在指令访问的数据面上，对所有原始的访问命令字进行截获，并实时执行“欺骗”式仿真动作，使得被访问的资源呈现出与其真正的物理资源不同的（软件无须关注硬件）、“按需获取”的颗粒度。对于普通 x86 服务器来说，CPU 和内存资源虚拟化后再将其（以虚拟机 CPU/内存规格）按需供给给资源消费者（上层业务用户）。计算能力的快速发展，以及软件通过负载均衡机制进行水平扩展的能力提升，计算虚拟化中仅存在资源池的“大分小”的问题。然而对于存储来说，由于最基本的硬盘（SATA/SAS）容量有限，而客户、租户对数据容量的需求越来越大，因此必须考虑对数据中心内跨越多个松耦合的分布式服务器单元内的存储资源（服务器内的存储资源、外置 SAN/NAS 在内的存储资源）进行“小聚大”的整合，组成存储资源池。这个存储资源池，可能是某一厂家提供的存储软硬件组成的同构资源池，也可以是被存储虚拟化层整合成为跨多厂家异构存储的统一资源池。各种存储资源池均能以统一的块存储、对象存储或者文件的数据面格式进行访问。对于数据中心网络来说，其实网络的需求并不是凭空而来的，而是来源于业务应用，与作为网络端节点的计算和存储资源有着无法切断的内在关联性。然而，传统的网络交换功能都是在物理交换机和路由器设备上完成的，网络功能对上层业务应用而言仅仅体现为一个一个被通信链路连接起来的孤立的“盒子”，无法动态感知来自上层业务的网络功能需求，完全需要人工配置的方式来实现对业务层网络组网与安全

隔离策略的需要。在多租户虚拟化的环境下，不同租户对于边缘的路由及网关设备的配置管理需求也存在极大的差异化，而物理路由器和防火墙自身的多实例能力也无法满足云环境下租户数量的要求，采用与租户数量等量的路由器与防火墙物理设备，成本上又无法被多数客户所接受。于是人们思考是否可能将网络自身的功能从专用封闭平台迁移到服务器通用 x86 平台上来。这样至少网络端节点的实例就可以由云操作系统来直接自动化地创建和销毁，并通过一次性建立起来的物理网络连接矩阵，进行任意两个网络端节点之间的虚拟通讯链路建立，以及必要的安全隔离保障，从而里程碑式地实现了业务驱动的网络自动化管理配置，大幅度降低数据中心网络管理的复杂度。从资源利用率的视角来看，任意两个虚拟网络节点之间的流量带宽，都需要通过物理网络来交换和承载，因此只要不超过物理网络的资源配额上限（缺省建议物理网络按照无阻塞的 CLOS 模式来设计实施），只要虚拟节点被释放，其所对应的网络带宽占用也将被同步释放，因此也就相当于实现对物理网络资源的最大限度的“网络资源动态共享”。换句话说，网络虚拟化让多个盒子式的网络实体第一次以一个统一整合的“网络资源池”的形态，出现在业务应用层面前，同时与计算和存储资源之间，也有了统一协同机制。

差别 3：资源池从小规模的资源虚拟化整合走向更大规模的资源池构建，应用范围从企业内部走向多租户的基础设施服务乃至端到端 IT 服务。从云计算提供像用水用电一样方便的服务能力的技术实现角度来看，云计算发展早期，虚拟化技术（如 VMware ESX、微软 Hyper-V、基于 Linux 的 XEN、KVM）被普遍采用，被用来实现以服务器为中心的虚拟化资源整合，在这个阶段，企业数据中心的服务器只是部分孤岛式的虚拟化以及资源池整合，还没有明确的多租户以及服务自动化的理念，服务器资源池整合的服务对象是数据中心的基础设施硬件以及应用软件的管理人员。在实施虚拟化之前，物理的服务器及存储、网络硬件是数据中心管理人员的管理对象，在实施虚拟化之后，管理对象从物理机转变为虚拟机及其对应的存储卷、软件虚拟交换机，甚至软件防火墙功能。目标是实现多应用实例和操作系统软件在硬件上最大限度共享服务器硬件，通过多应用负载的削峰填谷达到资源利用率提升的目的，同时为应用软件进一步提供额外的 HA/FT（High Availability/Fault Tolerance，高可用性/容错）可靠性保护，以及通过轻载合并、重载分离的动态调度，对空载服务器进行下电控制，实现 PUE 功耗效率的优化提升。然

而，这些虚拟化资源池的构建，仅仅是从数据中心管理员视角实现了资源利用率和能效比的提升，与真正的面向多租户的自动化云服务模式仍然相差甚远。因为在云计算进一步走向普及深入的新阶段，通过虚拟化整合之后的资源池的服务对象，不能再仅仅局限于数据中心管理员本身，而是需要扩展到每个云租户。因此云平台必须在基础设施资源运维监控管理 Portal 的基础上，进一步面向每个内部或者外部的云租户提供按需定制基础设施资源，订购与日常维护管理的 Portal 或者 API 界面，并将虚拟化或者物理的基础设施资源的增、删、改、查等权限按照分权分域的原则赋予每个云租户，每个云租户仅被授权访问其自己申请创建的计算、存储以及与相应资源附着绑定的 OS 和应用软件资源，最终使得这些云租户可以在无须购买任何硬件 IT 设备的前提下，实现按需快速资源获取，以及高度自动化部署的 IT 业务敏捷能力的支撑，从而将云资源池的规模经济效益，以及弹性按需的快速资源服务的价值充分发掘出来。

差别 4：数据规模从小规模走向海量，数据形态从传统结构化走向非结构化和半结构化。从云计算系统需要提供的处理能力角度看，随着智能终端的普及、社区网络的火热、物联网的逐步兴起，IT 网络中的数据形态已经由传统的结构化、小规模数据，迅速发展成为有大量文本、大量图片、大量视频的非结构化和半结构化数据，数据量也是呈几何指数的方式增长。对非结构化、半结构化大数据的处理而产生的数据计算和存储量的规模需求，已远远超出传统的 Scale-Up 硬件系统可以处理的，因此要求必须充分利用云计算提供的 Scale-Out 架构特征，按需获得大规模资源池来应对大数据的高效高容量分析处理的需求。企业内日常事务交易过程中积累的大数据或者从关联客户社交网络以及网站服务中抓取的大数据，其加工处理往往并不需要实时处理，也不需要系统处于持续化的工作态，因此共享的海量存储平台，以及批量并行计算资源的动态申请与释放能力，将成为未来企业以最高效的方式支撑大数据资源需求的解决方案选择。

差别 5：企业和消费者应用的人机交互计算模式，也逐步从本地固定计算走向云端计算、移动智能终端及浸入式体验瘦终端接入的模式随着企业和消费者应用云化演进的不断深入，用户近端计算、存储资源不断从近端计算剥离，并不断向远端的数据中心迁移和集中化部署，从而带来了企业用户如何通过企业内部局域网及外部固定、移动宽带广域网等多种不同途径，借助固

定、移动，乃至浸入式体验等多种不同瘦终端或智能终端形态接入云端企业应用的问题。面对局域网及广域网连接在通信包转发与传输时延不稳定、丢包以及端到端 QoS 质量保障机制缺失等实际挑战，如何确保远程云接入的性能体验达到与本地计算相同或近似的水平，成为企业云计算 IT 基础设施平台面临的又一大挑战。为应对云接入管道上不同业务类型对业务体验的不同诉求，业界通用的远程桌面接入协议在满足本地计算体验方面已越来越无法满足当前人机交互模式发展所带来的挑战，需要重点聚焦解决面向 IP 多媒体音视频的端到端 QoS/QoE 优化，并针对不同业务类别加以动态识别并区别处理，使其满足如下场景需求：

- 普通办公业务响应时延小于 100ms，带宽占用小于 150Kbps：通过在服务器端截获 GDI/DX/OpenGL 绘图指令，结合对网络流量的实时监控和分析，从而选择最佳传输方式和压缩算法，将服务端绘图指令重定向到瘦客户端或软终端重放，从而实现时延与带宽占用的最小化。
- 针对虚拟桌面环境下 VoIP 质量普遍不佳的情况，缺省的桌面协议 TCP 连接不适合作为 VoIP 承载协议的特点：采用 RTP/UDP 代替 TCP，并选择 G.729/AMR 等成熟的 VoIP Codec；瘦客户端可以在支持 VoIP/UC 客户端的情况下，尽量引入 VoIP 虚拟机旁路方案，从而减少不必要的额外编解码处理带来的时延及话音质量上的开销。上述优化措施使得虚拟桌面环境下的话音业务 MOS 平均评估值从 3.3 提升到 4.0。
- 针对远程云接入的高清（1080p/720p）视频播放场景：在云端桌面的多虚拟机并发且支持媒体流重定向的场景下，针对普通瘦终端高清视频解码处理能力不足的问题，桌面接入协议客户端软件应具备通过专用 API 调用具备瘦终端芯片多媒体硬解码处理能力；部分应用如 Flash 以及直接读写显卡硬件的视频软件，必须依赖 GPU 或硬件 DSP 的并发编解码能力，基于通用 CPU 的软件编解码将导致画面停滞、体验无法接受，此时就需要引入硬件 GPU 虚拟化或 DSP 加速卡来有效提升云端高清视频应用的访问体验，达到与本地视频播放相同的清晰与流畅度。桌面协议还能够智能识别并区分画面变化热度，仅对变化度高且绘图指令重定向无法覆盖部分才启动带宽消耗较高的显存数据压缩重定向。

- 针对工程机械制图、硬件 PCB 制图、3D 游戏，以及近期兴起 VR 仿真等云端图形计算密集型类应用：同样需要大量的虚拟化 GPU 资源进行硬件辅助的渲染与压缩加速处理，同时对接入带宽（单路几十到上百 M 带宽，并发数达到 10G/100G）提出了更高的要求，在云接入节点与集中式数据中心站点间的带宽有限的前提下，就需要考虑进一步将大集中式的数据中心改造为逻辑集中、物理分散的分布式数据中心，从而将 VDI/VR 等人机交互式重负载直接部署在靠近用户接入的 Service PoP 点的位置上。

另一方面，正当全球消费者 IT 步入方兴未艾的 Post-PC 时代大门之时，iOS 及 Android 移动智能终端同样正在悄悄取代企业用户办公位上的 PC 甚至便携电脑，企业用户希望通过智能终端不仅可以方便地访问传统 Windows 桌面应用，同样期待可以从统一的“桌面工作空间”访问公司内部的 Web SaaS 应用、第三方的外部 SaaS 应用，以及其他 Linux 桌面系统里的应用，而且希望一套企业的云端应用可以不必针对每类智能终端 OS 平台开发多套程序，就能够提供覆盖所有智能终端形态的统一业务体验，针对此 BYOD 云接入的需求，企业云计算需在 Windows 桌面应用云接入的自研桌面协议基础上，进一步引入基于 HTML5 协议、支持跨多种桌面 OS 系统、支持统一认证及应用聚合、支持应用零安装升级维护，及异构智能终端多屏接入统一体验的云接入解决方案——Web Desktop。

差别 6：云资源服务从单一虚拟化，走向异构兼容虚拟化、轻量级容器化以及裸金属物理机服务器在传统企业 IT 架构向目标架构演进的过程中，为了实现应用的快速批量可复制，以闭源 VMware、Hyper-V 及开源 XEN、KVM 为代表的虚拟化是最早成熟和广泛采纳的技术，使得应用安装与配置过程可基于最佳实践以虚拟机模板和镜像的形式固化下来，从而在后续的部署过程中大大简化可重复的复杂 IT 应用的安装发放与配置过程，使得软件部署周期缩短到以小时乃至以分钟计算的程度。然而，随着企业 IT 应用越来越多地从小规模、单体式的有状态应用走向大规模、分布式、数据与逻辑分离的无状态应用，人们开始意识到虚拟机虽然可以较好地解决大规模 IT 数据中心内多实例应用的服务器主机资源共享的问题，但对于租户内部多个应用，特别是成百上千，甚至数以万计的并发应用实例而言，均需重复创建成百上千的操作系统实例，资源消

耗大，同时虚拟机应用实例的创建、启动，以及生命周期升级效率也难以满足在线 Web 服务类、大数据分析计算类应用这种突发性业务对快速资源获取的需求。以 Google、Facebook、Amazon 等为代表的互联网企业，开始广泛引入 Linux 容器技术（namespace、cgroup 等机制），基于共享 Linux 内核，对应用实例的运行环境以容器为单位进行隔离部署，并将其配置信息与运行环境一同打包封装，并通过容器集群调度技术（如 Kubernetes、MESOS、Swarm 等）实现高并发、分布式的多容器实例的快速秒级发放及大规模容器动态编排和管理，从而将大规模软件部署与生命周期管理，以及软件 DevOps 敏捷快速迭代开发与上线效率提升到了一个新的高度。尽管从长远趋势上来看，容器技术终将以其更为轻量化、敏捷化的优势取代虚拟化技术，但在短期内仍很难彻底解决跨租户的安全隔离和多容器共享主机超分配情况下的资源抢占保护问题，因此，容器仍将在可见的未来继续依赖跨虚拟机和物理机的隔离机制来实现不同租户之间的运行环境隔离与服务质量保障。与此同时，对于多数企业用户来说，部分企业应用和中间件由于特殊的厂家支持策略限制，以及对企业级高性能保障与兼容性的诉求，特别是商用数据库类业务负载，如 Oracle RAC 集群数据和 HANA 内存计算数据库，并不适合运行在虚拟化上，但客户依然希望针对这部分应用负载可以在物理机环境下获得与虚拟化、容器化环境下相似的基础设施资源池化按需供给和配置自动化能力。这就要求云平台和云管理软件不仅仅要实现物理机资源自身的自动化操作系统与应用安装自动化，也需要进一步在保障多租户隔离安全的情况下实现与存储和网络资源池协同的管理与配置自动化能力。

差别 7：云平台和云管理软件从闭源、封闭走向开源、开放，从云计算平台的接口兼容能力角度看，云计算早期阶段，闭源 VMware vSphere/vCenter、微软 SystemCenter/Hyper-V 云平台软件由于其虚拟化成熟度遥遥领先于开源云平台软件的成熟度，因此导致闭源的私有云平台成为业界主流的选择。然而，随着 XEN/KVM 虚拟化开源，以及 OpenStack、CloudStack、Eucalyptus 等云操作系统 OS 开源软件系统的崛起和快速进步，开源力量迅速发展壮大起来，迎头赶上并逐步成长为可以左右行业发展格局的重要决定性力量。仅以 OpenStack 为例，目前 IBM、HP、SUSE、Redhat、Ubuntu 等领先的软硬件公司都已成为 OpenStack 的白金会员，从 2010 年诞生第一个版本开始，平均每半年发布一个新版本，所有会员均积极投身到开源贡献

中来，繁荣的社区发展驱动其功能不断完善，并稳步、快速地迭代演进。2014 年上半年，OpenStack 的成熟度已与 vCloud/ vSphere 5.0 版本的水平相当，满足基本规模商用和部署要求。从目前的发展态势来看，OpenStack 开源大有成为云计算领域的 Linux 开源之势。回想 2001 年前后，当 Linux OS 仍相当弱小、UNIX 操作系统大行其道、占据企业 IT 系统主要生产平台的阶段，多数人不会想象到仅 10 年的时间，开源 Linux 已取代闭源 UNIX，成为主导企业 IT 服务端的缺省操作系统的选择，小型机甚至大型机硬件也正在进行向通用 x86 服务器的演进。

1.5 云计算的模式

云计算目前尚未有全球统一的标准分类，但是目前业内基本达成了共识，一般会从运营模式和服务模式来分类，下面我们逐一进行介绍。

1.5.1 按照运营模式分类

- 公有云

公有云是最先出现的云计算，也是最被大众熟知的云计算。我们前面提到的百度网盘、华为手机云备份、有道云笔记和网易云音乐都属于公有云。目前，公有云可以提供给用户众多的服务，用户可以通过互联网像使用水电一样使用 IT 服务。

公有云通常是由云服务提供商搭建的。从最终用户的角度来说，只需要购买云计算资源或者服务，而云计算所用到的硬件及相应的管理工作都由第三方服务商负责。公有云的资源向公众开放，使用公有云需依赖互联网。

- 私有云

私有云通常部署在企业或单位内部，运行在私有云上的数据全部保存在企业自有的数据中心内，如果需要访问这些数据，就需要经过部署在数据中心入口的防火墙，这样可以在最大程度上保护数据。私有云可以基于企业已有的架构进行部署，也可以使用绝大部分已经花了大价钱购买的硬件设备，可以保护客户的现有投资。所有的事情都有两面性，如果企业采用了私有云，可以保证数据的安全，可以利旧设备，但是，随着时间的推移，设备会越来越旧，更换这些设备需要一笔不小的费用。另一方面，为了保证数据的安全，用户之间可共享的东西非常少，就算有，也绝对不会和其它企业或者单位共享。

近些年，关于私有云还有一种说法，在公有云上购买专属云的服务，这种方式可以将企业的关键业务放到公有云上，保证用户在云上拥有专属的计算和存储资源，并使用高度可靠的隔离网络，满足租户关键应用系统的高可靠、高性能和高安全等要求。

- 混合云

混合云是一种比较灵活的云计算模式，它可能包含了公有云、私有云或者后面要讲的行业云中的两种或两种以上的云，用户的业务可以根据需求在这几种云上切换。由于安全和控制原因，并非所有的企业信息都能放置在公有云上，这样大部分已经应用云计算的企业将会使用混合云模式。很多企业将选择同时使用公有云和私有云，有一些也会同时建立公共云。因为公有云只会向用户使用的资源收费，所以集中云将会变成处理需求高峰的一个非常便宜的方式。比如对一些零售商来说，他们的操作需求会随着假日的到来而剧增，或者是有些业务会有季节性的上扬。同时混合云也为其它目的的弹性需求提供了一个很好的基础，比如，灾难恢复。这意味着私有云把公有云作为灾难转移的平台，并在需要的时候去使用它。这是一个极具成本效应的理念。另一个好的理念是，使用公有云作为一个选择性的平台，同时选择其它的公有云作为灾难转移平台。

混合云允许用户利用公共云和私有云的优势。还为应用程序在多云环境中的移动提供了极大的灵活性。此外，混合云模式具有成本效益，因为企业可以根据需要决定使用成本更昂贵的云计算资源。

由于设置更加复杂而难以维护和保护。此外，由于混合云是不同的云平台、数据和应用程序的组合，因此整合可能是一项挑战。在开发混合云时，基础设施之间也会出现主要的兼容性问题。

- 行业云

行业云就是由行业内或某个区域内起主导作用或者掌握关键资源的组织建立和维护，以公开或者半公开的方式，向行业内部或相关组织和公众提供有偿或无偿服务的云平台。

行业云不是一个新的概念，它是一个特定的创建云的方式，和公有云和私有云不一样的地方是行业云在创建过程中就会包含一些行业特性，比如，云计算具备了医药行业特性后，可将患者的病例及急诊记录放到云端，每个医院的医生都可以从云端获取到相关的有用信息。

行业云是一个很大的机遇，但是也非常具备挑战性，拿医药行业云来说，患者的所有信息都会被同一行业内的相关人员看到，如何保障个人信息就需要更高的云安全技术或手段。

1.5.2 按照服务模式分类

在云计算中，一般我们部署的所有应用都遵循统一的分层结构，应用程序是最终呈现给用户，用户通过应用程序的界面保存或创建出自己的数据，为了保证应用程序的正常运行，需要依赖最底层的硬件资源、运行在硬件资源上的操作系统，以及运行在操作系统之上的中间件和应用程序的运行环境。我们把应用程序在内的所有部分称为软件层，将最底层的硬件资源，包括网络资源、存储资源和计算资源，以及虚拟化层称为基础设施层，运行在操作系统之上、应用程序之下的所有中间部分称为平台层。

如果基础设施层由云服务商提供，其它由用户自营，这种模式称为 IaaS；如果基础设施层和平台层由云服务商提供，其它由用户自营，这种模式称为 PaaS；如果全部由云服务商提供，这种模式称为 SaaS。

使用运行在 PC 上的游戏举例，图 1-12 是从游民星空中截取出来的，是关于大型单机游戏《只狼：影逝二度》的配置需求，具体如下：

配置需求

最低配置		推荐配置	
系统	Windows 7 / 8 / 10(游戏仅支持64位)	系统	Windows 7 / 8 /10(游戏仅支持64位)
CPU	Intel Core i3-2100 / AMD FX-6300	CPU	Intel Core i5-2500K / AMD Ryzen 5 1400
内存	4 GB	内存	8 GB
硬盘	25 GB	硬盘	25 GB
显卡	NVIDIA GeForce GTX 760 / AMD Radeon HD 7950	显卡	NVIDIA GeForce GTX 970 / AMD Radeon RX 570
DX	DirectX 11.0	DX	DirectX 11.0

图 1-12 《只狼：影逝二度》配置需求

在此图片中，我们可以看到这款游戏对硬件的要求，如果这台 PC 由我们自己购买，然后自己安装操作系统及游戏软件，这种模式是传统的模式，属于非云计算类；假如我们拿着这个配置清单到公有云上购买了一台这样的云服务器，使用镜像完成了操作系统的安装，然后自己进行软件包的下载和安装，我们使用的 IaaS 的模式；在安装此类大型游戏时，经常会出现以下告错：

图 1-13 NET Framework 初始化报错信息

这是因为该游戏软件正常运行需要.NET Framework 的支持，属于该软件的运行环境。如果我们购买的不仅仅是硬件设备，而是已经安装和运行了.NET Framework 的环境，这种模式属于PaaS。

如果使用的是最后一种——SaaS，我们直接购买的是已经安装并且激活了的游戏环境，只需要输入自己的用户名密码即可开始我们的游戏之旅。

2 计算虚拟化简介

云计算 1.0 时代使用的主要技术是虚拟化，虚拟化可以将传统 IT 中的应用程序及操作系统与硬件解耦，这个功能很符合云计算的要求，并且虚拟化和云计算一样，都可以通过这个功能构建可用的、稳定的运行环境给应用程序，因此导致很多人很容易就将虚拟化和云计算弄混。

其实虚拟化只提供了 IaaS 模式的服务，而云计算还提供了其他模式的服务，但是其它模式的服务几乎都是在 IaaS 的基础上发展起来的，所以，虚拟化作为云计算的入门技术，需要被每一个云计算的从业者掌握。

虚拟化最大的功能就是将操作系统和应用程序都运行在虚拟机上，所以本章主要介绍虚拟机是如何被创建出来的，以及实现虚拟化的机制有哪些。

2.1 虚拟化简介

2.1.1 虚拟化是什么?

虚拟化和云计算不一样，虚拟化是一种技术，使用虚拟化，我们可以在一台物理服务器上模拟出多个独立的服务器来。通常，应用程序需要安装在操作系统上，而一台物理服务器只能同时运行一个操作系统，使用了虚拟化以后，每个模拟出来的服务器都可以有自己独立的操作系统，这样就相当于在一台服务器上同时运行了多个操作系统。

虚拟化的本质就是将原先的物理设备进行逻辑化，转化成一个文件夹或文件，实现软硬件的解耦。

从前，我们使用的物理设备是看得见摸得着的，对着设备清单或者物理配置清单，我们都可以找到对应的实物，例如在图 2-1 中，我们可以清晰地看到 CPU、内存、硬盘和网卡等设备。

图 2-1 物理服务器组件

使用虚拟化后，物理服务器转变成一个文件夹或文件，这里面一般会包含两部分，一部分用来记录虚拟机的配置信息，另一部分用来保存用户数据的磁盘文件。下图是从一个 KVM 虚拟机的配置文件中摘取的一部分，可以看到该虚拟机的名称、CPU 和内存配置、硬盘配置及网络配置等信息。

```
<name>instance-00000001</name>
  <uuid>0cc37c1c-3de2-416b-b291-7e3d612b1b39</uuid>
……
<memory unit='KiB'>8290304</memory>
<CPU mode='host-passthrough' check='none'>
   <topology sockets='4' cores='1' threads='1'/>
  </CPU>
……
<disk type='file' device='cdrom'>
     <driver name='qemu' type='raw' cache='none'/>
     <source file='/opt/HUAWEI/image/instances/0cc37c1c-3de2-416b-b291-
7e3d612b1b39/disk'/>
     <target dev='hda' bus='ide'/>
     <readonly/>
     <boot order='2'/>
     <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
……
<interface type='bridge'>
```

```
      <mac address='fa:16:3e:b2:89:fa'/>
      <source bridge='plybac46573-e7'/>
      <virtualport type='openvswitch'>
        <parameters interfaceid='bac46573-e780-471b-b0c8-1193cabbf2bf'/>
      </virtualport>
      <target dev='tapbac46573-e7'/>
      <model type='virtio'/>
      <driver name='vhost'  vringbuf='4096'/>
      <boot order='3'/>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x03'
function='0x0'/>
    </interface>
```

使用虚拟化前，每台物理机上只能同时运行一个操作系统，如果在服务器上运行多个主应用程序，那么不同应用之间可能会产生冲突和性能问题。实际上，最佳做法是每个服务器仅运行一个应用程序以避免这些问题，但是这么做的结果是系统资源长时间利用率较低。

使用虚拟化后，每台物理机上可以同时运行多个虚拟机，每个虚拟机上又可以运行一个操作系统，硬件资源利用率得到了有效提高，减少了硬件资源的浪费。并且由于虚拟化技术实现了软硬件的解耦合，虚拟化可以不受当前服务器的限制，在集群范围内实现业务的在线动态迁移，并且在迁移过程中可以做到业务无中断、用户无感知。虚拟机的动态迁移为高可用性 HA、动态资源调度 DRS 和分布式电源管理 DPM 等高级特性提供了可能，也为企业数据中心实现了业务的可移动性、降低运行成本、减少管理费用、整合服务器、容错容灾等。这些特性我们将在第五章《虚拟化特性介绍》中详细阐述。

2.1.2 计算虚拟化发展史

将物理服务器虚拟成多个虚拟机在 IT 领域不是一个新技术，其实早在 1964 年，“蓝色巨人”IBM 就开始尝试在大型机上实现虚拟化，甚至在 1961 年，IBM 的 709 机就已经实现了分时系统，将 CPU 占用切分为多个极短（1/100sec）的时间片，每一个时间片都执行着不同的任务。通过对这些时间片的轮询，就可以将一个 CPU 虚拟化（伪装）成多个虚拟 CPU，并且让每一颗虚拟 CPU 看起来都在同时运行，这就是虚拟机的雏形。后来的 system360 机也支持分时系统。

在 1972 年的时候，IBM 正式将 system370 机的分时系统命名为虚拟机。

1990 年，IBM 推出的 system390 机支持逻辑分区，将一个 CPU 分为若干份（最多 10 份），而且每份 CPU 都是独立的，也就是说，一个物理 CPU 可以逻辑地分为 10 个 CPU。

IBM 将分时系统开源后，开始实现基于 x86 架构的虚拟化。1999 年，VMware 推出了最早的能在 x86 架构上运行的虚拟化产品。

在 VMware 开发虚拟化产品的同时，20 世纪 90 年代，伦敦剑桥大学的 Lan Pratt 和 Keir Fraser 在一个叫做 Xenoserver 的研究项目中，开发了 Xen 虚拟机。作为 Xenoserver 的核心，Xen 虚拟机负责管理和分配系统资源，并提供必要的统计功能。在那个年代，x86 的处理器还不具备对虚拟化技术的硬件支持，所以 Xen 从一开始是作为一个准虚拟化的解决方案出现的。因此，为了支持多个虚拟机，其内核必须针对 Xen 做特殊的修改才可以运行。为了吸引更多的开发人员参与，2002 年 Xen 正式被开源，并先后推出了 1.0 和 2.0 版本。在这之后，Xen 作为虚拟化解决方案，开始被诸如 Redhat、Novell 和 Sun 的 Linux 发行版集成。2004 年，Intel 的工程师开始为 Xen 添加硬件虚拟化的支持，从而为即将上市的新款处理器做必需的软件准备。在他们的努力下，2005 年发布的 Xen 3.0，开始正式支持 Intel 的 VT 技术和 IA64 架构，因此 Xen 虚拟机就可以运行完全没有修改的操作系统了。

除了 Xen，另外一个就是大名鼎鼎的 KVM。KVM 最初是被以色列的一个创业公司 Qumranet 当做 VDI（Virtual Desktop Infrastructure）产品的虚拟机而开发的。为了简化开发，KVM 的开发人员并没有选择从底层开始重新写一个 Hypervisor（下一节有对 Hypervisor 的介绍），而是选择基于 Linux kernel，通过加载新的模块使 Linux Kernel 本身变成一个 Hypervisor。2006 年 10 月，在完成基本功能、动态迁移以及主要的功能和性能的优化后，Qumranet 正式对外宣布 KVM 的诞生。同年 10 月，KVM 模块的源代码被正式纳入 Linux Kernel，成为内核源代码的一部分。2008 年 9 月 4 日，与内核社区有着很深渊源的著名 Linux 发行版提供商——Redhat 公司出人意料地出资 1 亿 700 百万美金，收购了 Qumranet，成为了 KVM 开源项目的新东家。得益于此次收购，Redhat 公司有了自己的虚拟机解决方案，于是开始在自己的产品中用 KVM 替换 Xen。2010 年 11 月，Redhat 公司推出了新的企业版

Linux——RHEL 6，该发行版集成了最新的 KVM 虚拟机，替换了在 RHEL 5.x 系列中集成的 Xen。

在 2006 年至 2010 年期间，各大传统 IT 厂商在虚拟化方面都推出了自己的产品。2007 年，HP 推出了 Integrity 虚拟机，微软在 Windows Server 2008 R2 中加入了 Hyper-v。

当 x86 架构虚拟化发展地如火如荼时，一种轻量级的虚拟化也在慢慢发展着，那就是容器。最早的容器产品可以追溯到 1979 年的 UNIX chroot，后来经过多年的发展，直到 2008 年 Linux 推出了 LXC，也就是 Linux Containers，它是第一套完整的 Linux 容器管理实现方案。其功能通过 cgroups 以及 Linux namespaces 实现。LXC 通过 liblxc 库进行交付，并提供可与 Python3、Python2、Lua、Go、Ruby 以及 Haskell 等语言相对接的 API。相较于其它容器技术，LXC 能够在无需任何额外补丁的前提下运行在原版 Linux 内核上。目前 LXC 项目由 Canonical 有限公司负责赞助及托管。2013 年，推出了 Docker 容器项目。Docker 在起步阶段同样使用 LXC，之后将其替换为自己的 libcontainer 库。与其它容器平台不同，Docker 引入了一整套与容器管理相关的生态系统。其中包括一套高效的分层式容器镜像模型、一套全局及本地容器注册表、一个精简化 REST API 以及一套命令行界面等等。在后期发展阶段，Docker 公司还构建出一套名为 Docker Swarm 的容器集群管理解决方案。2014 年，Rocket 推出，它最初是由 CoreOS 开发的，专门用于解决 Docker 中存在的部分缺陷。CoreOS 方面也提到，他们当初开发的目标是在满足安全性与生产要求能力上超越 Docker。更重要的是，其基于 App Container 规范，并使其成为一项更为开放的标准。

2.1.3 计算虚拟化的分类

在介绍虚拟化分类前，我们先了解一下在虚拟化中常用到的几个名词。

首先，业界对于运行虚拟机的物理主机称为宿主机，即 Host Machine，而宿主机安装运行的操作系统称为宿主机操作系统，即 Host OS。运行在宿主机上的虚拟机称为客户机，即 Guest Machine。虚拟机安装运行的操作系统称为客户机操作系统，即 Guest OS。位于 Host

OS 和 Guest OS 之间的是所有虚拟化技术的核心——Hypervisor，也可以称为 VMM（Virtual Machine Monitor)。

在物理架构中，主机只有两个层次，即硬件层 Host Machine 和操作系统层 Host OS，应用安装在 Host OS 上；在虚拟化架构中，主机分为三个层次，硬件 Host Machine，其上为虚拟机监视器 Hypervisor，再上为虚拟机 Guest Machine 及其操作系统 Guest OS，应用安装在 Guest OS 上。在一个 Host Machine 上可以创建并运行多个 Guest Machine。

根据 Hypervisor 的不同类型，我们将虚拟化分为 I 型和 II 型两种，也有其它的说法，比如把容器算成 III 型，由于容器不在本课程的讨论范围内，所以此处以 I 型和 II 型为准。

I 型虚拟化，也称裸金属虚拟化，Hypervisor 直接调用硬件资源，不需要底层 Host OS，或者说在 I 型虚拟化中，可以将 Hypervisor 看作是一个定制的 Host OS，除了起到 VMM 的作用外，一般不能在其上安装其它应用。Hypervisor 主要实现两个基本功能：首先是识别、捕获和响应虚拟机发出的 CPU 特权指令或保护指令（特权指令和保护指令在 CPU 虚拟化中介绍)；其次，它负责处理虚拟机队列和调度，并将物理硬件的处理结果返回给相应的虚拟机。也就是说，Hypervisor 负责管理所有的资源及虚拟环境。VMM 可以看作是一个为虚拟化而生的完整操作系统，管理所有资源（CPU、内存和 I/O 设备)。VMM 承担管理资源的重任，并向上提供虚拟机 VM 用于运行 Guest OS 的环境，因此 VMM 还负责虚拟环境的创建和管理。采用该结构的虚拟化产品主要有：VMWare ESX Server、Citrix XenServer 和 FusionCompute 等。

注：通俗的讲，I 型虚拟化中 Hypervisor 啥也不干，就专门负责将物理的硬件资源转换为虚拟资源供 Guest OS 使用，Guest OS 就像直接运行在物理硬件上一样，所以称为裸金属。

I 型虚拟化有以下特点：

- 优点：虚拟机不依赖于操作系统，支持多种操作系统和多种应用。
- 缺点：虚拟化层内核开发难度大。

II 型虚拟化，也称宿主型虚拟化，此模型的物理资源由 Host OS（例如 Windows，Linux 等）管理，实际的虚拟化功能由 VMM 提供，而 VMM 作为底层操作系统（Windows 或 Linux

等）上的一个普通应用程序，通过其创建相应的虚拟机，共享底层服务器资源。VMM 通过调用 Host OS 的服务来获得资源，实现 CPU、内存和 I/O 设备的虚拟化。VMM 创建出虚拟机 VM 后，通常将 VM 作为 Host OS 的一个进程参与调度。采用该结构的虚拟化产品主要有：VMware Workstation、Virtual PC 等。

II 型虚拟化有以下特点：

- 优点：简单、易于实现。
- 缺点：安装和运行应用程序依赖于主机操作系统对设备的支持。管理开销较大，性能损耗大。

注：和 I 型虚拟化不一样，II 型虚拟化中 Hypervisor 只是 Host OS 的一个应用程序，所有的硬件资源还是归 Host OS 管理。

无论是 I 型还是 II 型虚拟化，都具有分区、隔离、封装和独立的特点。

（1）分区：分区意味着虚拟化层为多个虚拟机划分服务器资源的能力；每个虚拟机可以同时运行一个单独的操作系统（相同或不同的操作系统），使您能够在一台服务器上运行多个应用程序；每个操作系统只能看到虚拟化层为其提供的“虚拟硬件”（虚拟网卡、CPU、内存等），使它误以为运行在自己的专用服务器上。分区解决了以下两个方面的问题：

- 每个分区划分资源配额，防止虚拟化超配额使用资源
- 每个虚拟机单独安装操作系统，彼此互不影响

（2）隔离：通过分区所建立的多个虚拟机之间采用逻辑隔离措施，防止相互影响。隔离解决了以下两个方面的问题：

- 一个虚拟机的崩溃或故障（例如，操作系统故障、应用程序崩溃、驱动程序故障，等等）不会影响同一服务器上的其它虚拟机
- 一个虚拟机中的病毒、蠕虫等，与其它虚拟机相隔离，就像每个虚拟机都位于单独的物理机器上一样

通过隔离，我们可以进行资源控制以提供性能隔离，即为每个虚拟机指定最小和最大的资源使用量，以确保某个虚拟机不会占用所有资源而使同一系统中的其它虚拟机无资源可用；也可以在单一机器上同时运行多个负载/应用程序/操作系统，而不会出现我们刚才讨论的传统 x86 架构服务器的局限性时提到的那些问题（应用程序冲突、DLL 冲突等）

（3）封装：封装意味着将整个虚拟机（硬件配置、BIOS 配置、内存状态、磁盘状态、CPU 状态）储存在独立于物理硬件的一小组文件中。只需复制几个文件就可以随时随地根据需要复制、保存和移动虚拟机。不少人都用过 VMWare Workstation 这个虚拟化产品，我们创建的虚拟机是可以通过复制一小组虚拟机文件到其它安装有 VMWare Workstation 的电脑上重新打开运行的。对虚拟机的迁移而言，最重要的就是封装特性，封装也是虚拟化所有本质特性中最为重要的特性。这是因为虚拟机成为独立于硬件的文件，那么虚拟机就可以具备迁移和热插拔等特性，这些也都和虚拟机封装特性息息相关。

（4）相对硬件独立：虚拟机封装为独立文件后，虚拟机迁移只需要把虚拟机设备文件和配置文件或磁盘文件复制到另一台主机上运行即可，而不用关心底层的硬件类型是否兼容，这就是相对硬件的独立性。因为底层的硬件设备被其上运行的虚拟机监视器 VMM（Virtual Machine Monitor）屏蔽，运行在 VMM 之上的虚拟机只要关心目的主机是否也存在相同的 VMM，而不用关心底层的硬件规格配置等信息。这就好比我们在 Windows 7 系统上用 office 2007 编辑 word 文件，然后把这个 word 文件复制到另一台 Windows 10 系统的电脑上，这时我们只要关心是否有可以打开文档的 office 2007，而不必关心底层硬件 CPU 是什么型号，内存有多大规格。

2.2 计算虚拟化

计算虚拟化根据虚拟机组成的设备类型包含 CPU 虚拟化、内存虚拟化和 IO 虚拟化。

2.2.1 CPU 虚拟化

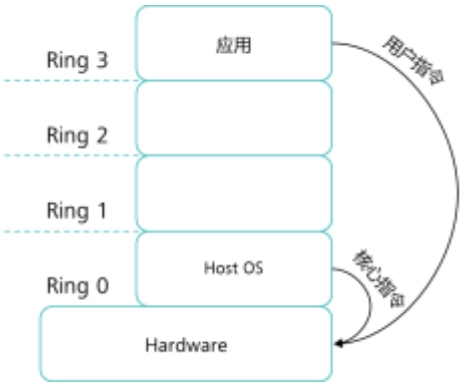


图 2-2 CPU 分级保护域

在讲 CPU 虚拟化前，我们先简单介绍一下 CPU 的分级保护域，在这种保护模式中，CPU 被分成 4 个环——Ring0、Ring1、Ring2 和 Ring3，其中 Ring0 的权限最高，Ring1 次之，Ring2 再次之，Ring3 最低。Ring0 的权限可以直接操作硬件，一般只有操作系统和驱动会允许拥有此权限。Ring3 的权限最低，所有程序都可以拥有此权限。为了保护计算机，一些危险的指令只能由操作系统执行，以防止一些恶意软件随意地调用硬件资源，比如某个程序需要开启摄像头就必须向 Ring0 的驱动程序请求开启，否则会被拒绝此类操作。

主机操作系统所发出的指令一般分为两种类型：特权指令和普通指令。

- 特权指令：是指用于操作和管理关键系统资源的指令，这些指令只在最高特权级上才能够运行，即必须在 Ring 0 级别上才能运行的指令。
- 普通指令：与特权指令相对的是普通指令，这些指令在 CPU 普通权限级别上就能够运行，即在 Ring 3 级别上就可以运行的指令。

在虚拟化环境中，还有一种特殊指令被称为敏感指令。敏感指令是指修改虚拟机的运行模式或宿主机状态的指令，也就是说将 Guest OS 中原本需要在 Ring 0 模式下才能运行的特权指令剥夺特权后，交给 VMM 来执行的指令。

虚拟化技术首先出现在 IBM 大型机上，大型机是如何解决 CPU 共享问题的呢？首先我们来了解一下大型机的 CPU 虚拟化方式。大型机 CPU 虚拟化采用的方法是“特权解除（Privilege deprivileging)”和“陷入模拟（Trap-and-Emulation)”，该方法也被称为经典虚拟化方式。它的基本原理是，将 Guest OS 运行在非特权级（即“特权解除”），而将 VMM 运行在最高特权级（即完全控制系统资源)。

这时候就出现了这样一个问题：如果虚拟机 Guest OS 发出特权操作指令怎么执行呢？因为所有虚拟机的系统都被解除了特权，于是“陷入模拟”就发挥作用了，它解除 Guest OS 的特权后，Guest OS 的大部分指令仍可以在硬件上直接运行，只有当执行到特权指令时，才会陷入到 VMM 模拟执行（陷入 - 模拟)。由 VMM 代替虚拟机向真正的硬件 CPU 发出特权操作指令。

CPU 经典虚拟化方式结合原始操作系统具有的定时器中断机制，完美解决了 CPU 虚拟化的问题。以下图 2-3 为例，虚拟机 1 发送特权指令 1 到虚拟机监视器 VMM，此时触发中断，虚拟机监视器 VMM 会将虚拟机 1 发送的特权指令 1 陷入到虚拟机监视器 VMM 中进行模拟，再转换成 CPU 的特权指令 1’，虚拟机监视器 VMM 根据调度机制调度到硬件 CPU 上执行，并返回结果给虚拟机 1。以下图 2-4 为例，当虚拟机 1 和虚拟机 2 同时发送特权指令到虚拟机监视器 VMM 时，指令都被陷入模拟，虚拟机监视器 VMM 调度机制进行统一调度。首先执行指令 1’，然后再执行指令 2’。所以采用定时器中断机制以及特权解除—陷入模拟的方法可以成功实现 CPU 的虚拟化功能。

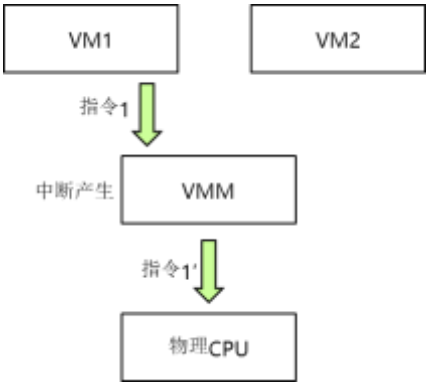


图 2-3 所有指令统一调度

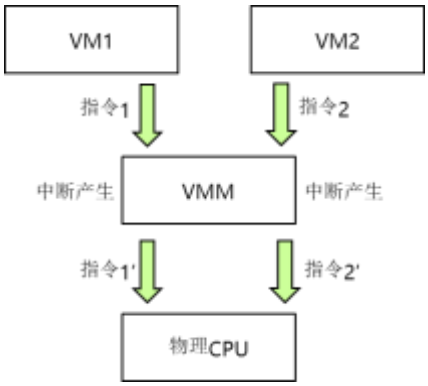


图 2-4 特殊指令示意图

那么为什么需要中断机制呢？CPU 在程序运行时系统外部、系统内部或当前运行的程序本身若出现紧急事件，CPU 立即中止当前运行的程序，自动转入相应的处理程序（中断服务程序），待处理完后，再返回原来的程序运行，这整个过程称为程序中断。例如你在看视频时，QQ 突然有信息弹出，在这个过程中就会触发中断机制。CPU 会暂停视频播放进程，而去执行 QQ 进程。CPU 在处理完 QQ 进程后，继续执行视频播放进程。当然，这个中断时间非常短暂，对用户是无感知的。

随着 x86 主机性能越来越强，如何将虚拟化技术应用到 x86 架构成为实现 x86 服务器虚拟化的主要问题。这时，人们自然而然想到曾经在大型机上使用的 CPU 虚拟化技术。那么大型机使用的 CPU 经典虚拟化方式能否移植到 x86 服务器上呢？答案是否定的，这是为什么呢？要回答这个问题，我们就需要先了解 x86 架构的 CPU 和大型机 CPU 的不同之处。

大型机（包括后来发展的小型机）是 PowerPC 架构，即精简指令集 RISC 计算机架构。RISC 架构的 CPU 指令集中，虚拟机特有的敏感指令是完全包括在特权指令中的，如下图 2-5 所示。在虚拟机操作系统解除特权后，特权指令和敏感指令都可以被正常陷入-模拟并执行，因为特权指令包含敏感指令，所以 RISC 架构的 CPU 采用特权解除和陷入模拟是没有问题的。但是 x86 架构的 CPU 指令集是不同于 RISC 架构的 CISC 架构，如下图 2-6 所示。

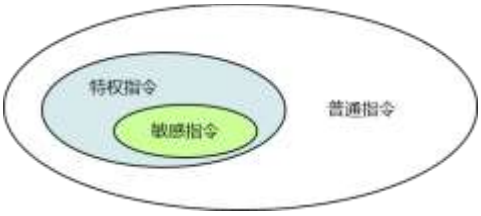


图 2-5 RISC 架构指令集

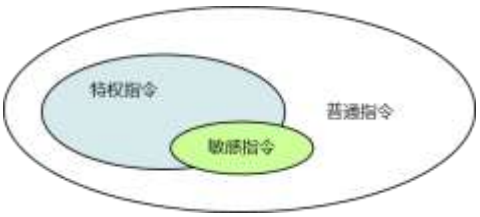


图 2-6 CISC 架构 CPU 指令集

在上图中可以看到 CISC 架构的 CPU 指令集的特权指令和敏感指令并不完全重合，具体来说，基于 x86 的 CISC 指令集有 19 条敏感指令不属于特权指令的范畴，这部分敏感指令运行在 CPU 的 Ring 1 用户态上。这会带来什么问题呢？显然，当虚拟机发出这 19 条敏感指令时，由于指令不属于特权指令，因而这些敏感指令不能陷入-模拟被虚拟机监视器（VMM）捕获，因此 x86 无法使用“解除特权”“陷入-模拟”的经典虚拟化方式实现 X86 架构的虚拟化。这种问题被

称为虚拟化漏洞问题。既然基于大型机的 CPU 虚拟化方案无法直接移植到 x86 平台上，那么 x86 应该采用什么方式实现 CPU 虚拟化呢?

聪明的 IT 架构师们想出了解决这个问题的三种方法。它们分别是：全虚拟化、半虚拟化以及硬件厂商提出的硬件辅助虚拟化。

- 全虚拟化解决方案

经典虚拟化方案不适合 x86 架构的 CPU，其根本原因在于那 19 条超出特权指令的敏感指令。如果识别出这些敏感指令，并使其可以被 VMM 陷入-模拟，那么 CPU 虚拟化就解决了。但是该如何识别出这 19 条指令呢?

有一种类似于“宁可错杀三千，绝不放过一个”的思路，也就是说将所有虚拟机发出的操作系统请求转发到虚拟机监视器（VMM），虚拟机监视器对请求进行二进制翻译（Binary Translation），如果发现是特权指令或敏感指令，则陷入到 VMM 模拟执行，然后调度到 CPU 特权级别上执行；如果只是应用程序指令则直接在 CPU 非特权级别上执行。这种方法由于需要过滤所有虚拟机发出的请求指令，因而被称为全虚拟化方式。全虚拟化的实现方式如下图 2-7。

全虚拟化方案最早是由 Vmware 提出并实现的，运行时虚拟机监视器 VMM 对虚拟机操作系统 Guest OS 二进制代码进行翻译，不修改虚拟机操作系统，虚拟机的可移植性和兼容性较强，但二进制翻译会带来虚拟机监视器（VMM）性能的开销。全虚拟化方式的优点是：不修改虚拟机操作系统，虚拟机的可移植性和兼容性较强，支持广泛的操作系统；但缺点是：运行时修改 Guest OS 二进制代码，性能损耗较大，并且引入了新的复杂性，导致虚拟机监视器（VMM）开发难度较大。正是因为全虚拟化具有上述缺点，因此 Xen 提出了半虚拟化解决方案。

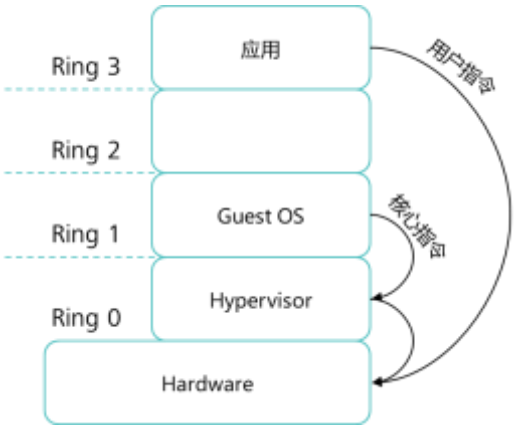


图 2-7 全虚拟化示意图

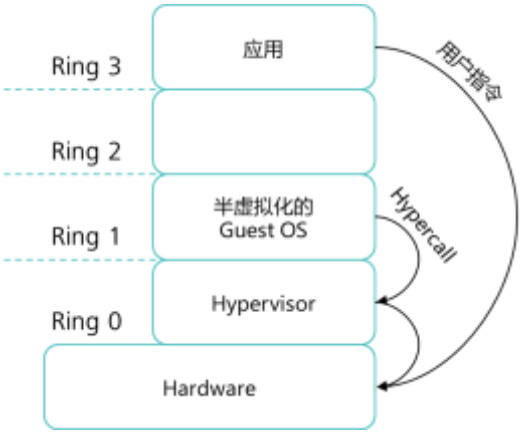


图 2-8 半虚拟化示意图

- 半虚拟化解决方案

虚拟化漏洞的问题来源于 19 条敏感指令，如果我们可以修改虚拟机操作系统 Guest OS 规避虚拟化漏洞，那么问题就容易解决了。

半虚拟化如上图 2-8 所示，修改虚拟机操作系统 Guest OS，让虚拟机操作系统能够意识到自己是被虚拟化的，虚拟机操作系统会“超级调用”（Hypercall）Hypervisor 层来替换虚拟化中的敏感指令，从而实现虚拟化，而其它应用程序等非敏感或非特权请求则直接在 CPU 非特权级别上执行。半虚拟化所具有的优点是：半虚拟化中的 Guest OS 可以同时支持多个不同的操作系统，并提供与原始系统相近的性能。但缺点是：半虚拟化中的 Host OS 只支持修改开源的操作系统，如 Linux，而对于未开源的如 Windows 系统，则无法实现半虚拟化。此外，被修改过的虚拟机操作系统 Guest OS 可移植性较差。

- 硬件辅助虚拟化解决方案

虚拟化漏洞问题的解决，无论是全虚拟还是半虚拟，都默认一个前提，即物理硬件是不具备虚拟化识别功能的，因此必须识别出这 19 条敏感指令，并通过虚拟机监视器 VMM 进行陷入-模拟。如果物理 CPU 直接支持虚拟化功能，并且可以识别出敏感指令，那么 CPU 虚拟化方式就将发生革命性的改变。

幸运的是，目前主流的 x86 主机的 CPU 都支持硬件虚拟化技术，即 Intel 推出了 VT-x（Intel Virtualization Technology）的 CPU，AMD 也推出了 AMD-V 的 CPU。VT-x 和 AMD-

V，这两种技术都为 CPU 增加了新的执行模式——root 模式，可以让虚拟机监视器 VMM 运行在 root 模式下，而 root 模式位于 CPU 指令级别 Ring 0 下。特权和敏感指令自动在 Hypervisor 上执行，所以无需全虚拟化或半虚拟化技术。这种通过硬件辅助虚拟化解决虚拟化漏洞，简化 VMM 的工作，不需要半虚拟化和二进制翻译的方法，被称为 CPU 的硬件辅助虚拟化技术。硬件辅助虚拟化技术如下图所示。

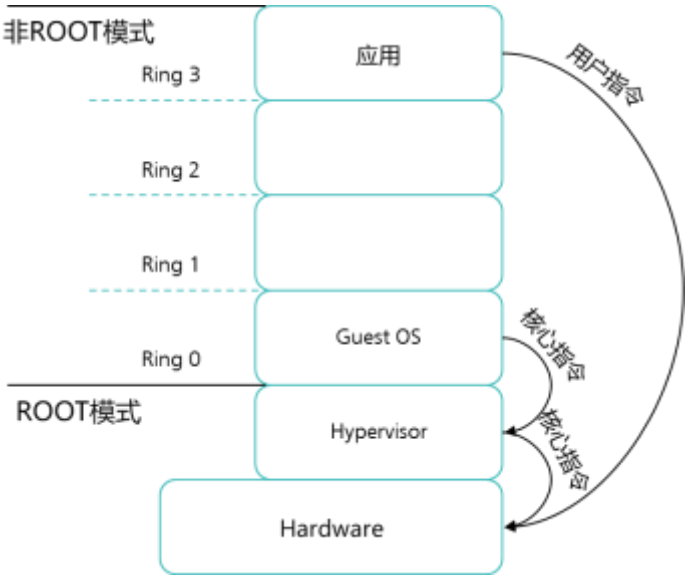


图 2-9 硬件辅助虚拟化

2.2.2 内存虚拟化

CPU 实现了虚拟化之后，与 CPU 关系极为密切的硬件——内存也将发生巨大的变化。为什么 CPU 虚拟化后会导致内存虚拟化的出现呢?

随着 CPU 虚拟化的出现和应用，运行在 VMM 层之上的虚拟机取代了物理主机，成为承载业务和应用的载体，并且在一台物理主机上有多台虚拟机同时运行着。那么问题来了，物理主机通常只有一根或几根内存条，面对多个虚拟机对内存的需求，该如何分配内存资源呢？显然，解决问题的办法还是虚拟化技术，因此内存虚拟化技术就出现了。内存虚拟化遇到的一个问题是：如何分配内存地址空间？因为通常情况下，物理主机在使用内存地址空间时，要求满足如下两点要求：

- 内存地址都是从物理地址 0 开始的

- 内存地址空间都是连续分配的

但很显然，引入虚拟化后就出现了这样的问题：首先是要求内存地址空间都从物理地址 0 开始，因为物理地址为 0 的内存地址空间只有一个，无法同时满足所有虚拟机使用的内存都从 0 开始的要求；其次是内存地址连续分配问题。即使可以为虚拟机分配连续的物理地址，但是内存使用效率不高，缺乏灵活性。

解决内存共享问题的思路就在于引入内存虚拟化技术。内存虚拟化就是把物理机的真实物理内存统一管理，包装成多份虚拟的内存给若干虚拟机使用。内存虚拟化技术的核心在于引入一层新的地址空间——客户机物理地址空间，客户机（Guest）以为自己运行在真实的物理地址空间中，实际上它是通过 VMM 访问真实的物理地址的，在 VMM 中保存的是客户机地址空间和物理机地址空间之间的映射表，如下图 2-10 所示。

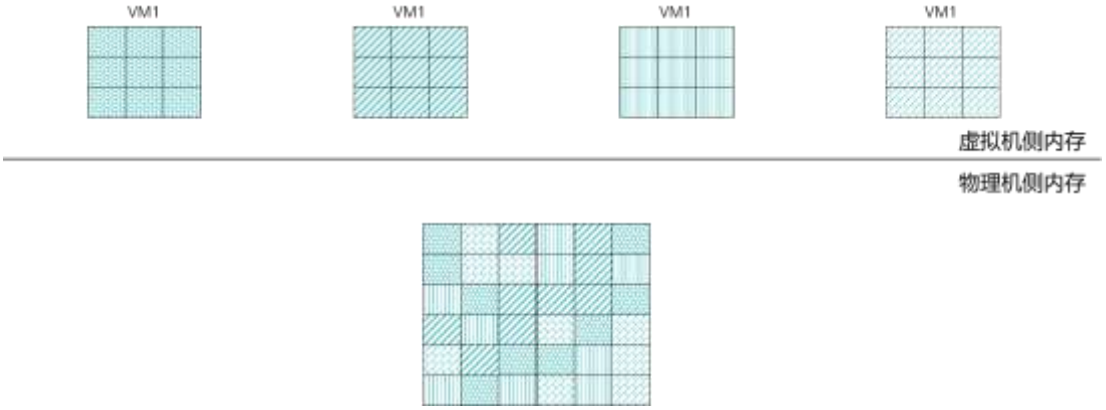


图 2-10 内存虚拟化示意图

内存虚拟化的内存地址转换涉及到三种内存地址，即虚拟机内存地址（Virtual Memory Address，即 VA）、物理内存地址（Physical Memory Address，即 PA）和机器内存地址（Machine Memory Address，即 MA）。为了能够在物理主机上运行多个虚拟机，需要实现 VA（虚拟内存）→PA（物理内存）→MA（机器内存）之间的地址转换。虚拟机 Guest OS 控制虚拟地址到客户内存物理地址的映射（VA→PA），但是虚拟机 Guest OS 不能直接访问实际机器内存，因此 Hypervisor 需要负责映射客户物理内存到实际机器内存（PA→MA）。

注：这个地方可能有人要问 MA 和 PA 的区别，比如一台服务器一共有 16 根 16G 的内存条，那么它的 PA 为 256G，而 MA 为 16 根分布在不同内存槽位的内存条。

2.2.3 I/O 虚拟化

由于计算虚拟化的出现，物理服务器上会创建出许许多多的虚拟机，并且每台虚拟机都需要访问物理主机的 I/O 设备。但 I/O 设备的数量毕竟是有限的，为了满足多个虚拟机共同使用 I/O 设备的需求，就需要虚拟机监视器 VMM 的参与。VMM 用于截获虚拟机对 I/O 设备的访问请求，再通过软件去模拟真实的 I/O 设备，进而响应 I/O 请求。从而让多个虚拟机访问有限的 I/O 资源。实现 I/O 虚拟化的方式主要有三种：全虚拟化、半虚拟化和硬件辅助虚拟化，其中硬件辅助虚拟化技术是目前 I/O 虚拟化的主流技术。

（1）全虚拟化：全虚拟化的实现原理是，通过 VMM 为虚拟机模拟出一个与真实设备类似的虚拟 I/O 设备，当虚拟机对 I/O 设备发起 I/O 请求时，VMM 截获虚拟机下发的 I/O 访问请求，再由 VMM 将真实的访问请求发送到物理设备进行处理。这种虚拟化方式的优点是：虚拟机无论使用任何类型的操作系统，操作系统都不需要为 I/O 虚拟化做任何修改，就可以让多个虚拟机直接使用物理服务器的 I/O 设备。但这种方式的缺陷是：VMM 需要实时截获每个虚拟机下发的 I/O 请求，截获请求后模拟到真实的 I/O 设备中。实时监控和模拟的操作都是通过 CPU 运行软件程序来实现的，因此会给服务器带来较严重的性能损耗。

（2）半虚拟化：半虚拟化方式与全虚拟化方式的明显区别是：它需要建立一个特权级别的虚拟机，即特权虚拟机。半虚拟化方式要求各个虚拟机运行前端驱动程序，当需要访问 I/O 设备时，虚拟机通过前端驱动程序把 I/O 请求发送给特权虚拟机，由特权虚拟机的后端驱动收集每个虚拟机所发出的 I/O 请求，再由后端驱动对多个 I/O 请求进行分时分通道处理。特权虚拟机运行真实的物理 I/O 设备驱动，将 I/O 请求发送给物理 I/O 设备，I/O 设备处理完成后再将结果返回给虚拟机。半虚拟化方式的优点是：主动让虚拟机把 I/O 请求发送给特权虚拟机，再由特权虚拟机访问真实的 I/O 设备，这就减少了 VMM 的性能损耗。但这种方式也有一个缺陷，即需要修改虚拟机操作系统，改变操作系统对自身 I/O 请求的处理方式，将 I/O 请求全部发给特权虚拟机处

理。这就要求虚拟机操作系统必须是可以被修改的（通常是 Linux 操作系统）。如下图 2-11 所示。

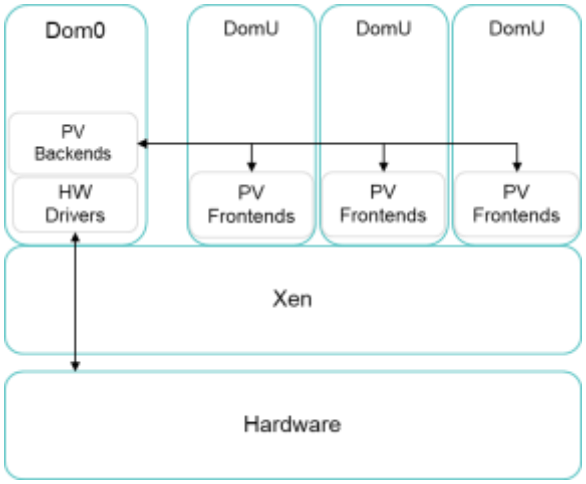


图 2-11 Xen 架构示意图

在上图 2-11 中，Domain 0 就是特权虚拟机，Domain U 是用户虚拟机。所有用户虚拟机的设备信息保存在特权虚拟机 Domain0 的 XenSToRe 中，用户虚拟机中的 XenBus（为 Xen 开发的半虚拟化驱动）通过与 Domain0 的 XenSToRe 通信，获取设备信息，加载设备对应的前端驱动程序。当用户虚拟机有 I/O 请求时，前端设备驱动将数据通过接口全部转发到后端驱动。后端驱动则对 I/O 请求的数据进行分时分通道处理。最终通过 Domain 0 的物理 I/O 设备驱动，将 I/O 请求发送给物理 I/O 设备。

我们来比较下全虚拟化和半虚拟化两种方式，全虚拟化的 VMM 相当于是一个调查员，他需要收集和归纳每一位客户的意见请求；而半虚拟化的 VMM 则相当于一个意见收纳箱（即特权虚拟机），各个用户自行将意见请求放入收纳箱，再由 VMM 统一处理这些意见请求。由于半虚拟化显著减少了 VMM 的性能损耗，因而能获得更好的 I/O 性能。但我们也注意到，全虚拟化和半虚拟化这两种方式都有一个共同的特点，即 I/O 访问处理都需要由 VMM 介入，这势必会造成虚拟机在访问 I/O 设备时的性能损耗。

（3）硬件辅助虚拟化：硬件辅助虚拟化不同于前面两种方式，它是将 I/O 设备驱动直接安装在虚拟机操作系统中，不需要对操作系统做任何修改即可使用。这种方式与我们熟悉的传统 PC

主机操作系统直接访问硬件是相同的。这就使得虚拟机访问 I/O 硬件所需的时间与传统 PC 机的访问时间相同。按照前面的例子，硬件辅助虚拟化就相当于一个智能的信息收集、处理平台，用户的意见请求可以直接向该平台提交，并自助完成业务处理，无需人工干预。因此硬件辅助虚拟化在 I/O 性能上远远超过全虚拟化和半虚拟化方式。但硬件辅助虚拟化需要特殊的硬件支持。

2.2.4 主流的计算虚拟化

通过 CPU 虚拟化、内存虚拟化和 I/O 虚拟化，进行物理资源的复用，一台物理主机可以同时运行多台虚拟服务器，每个虚拟服务器又能承载不同的业务，从而提高硬件的利用率。另外，由于虚拟服务器被逻辑划分成一个文件或文件夹，打破了软硬件的强耦合，通过自动移动这些文件或文件夹，就可以提高运行在虚拟机上的业务的可靠性。在云计算中，我们主要使用虚拟化来实现 IaaS 的云服务。

而云计算不仅仅是 IaaS，还包括 PaaS 和 SaaS，有一部分 PaaS 和 SaaS 是基于虚拟化来实现的，还有一部分是基于物理硬件+分布式计算来实现的。

2019 年春节，中国第一部硬科幻电影《流浪地球》上映，中国式情节深深地感染了每一个中国人，逼真的画面也震撼了所有人，电影画面的渲染使用了华为公有云的渲染解决方案，在这个解决方案中，有多个产品，如果是标清渲染的话，可以使用 C3 型号的弹性云服务器+其它的云服务，其中 C3 型号的弹性云服务器底层使用的是虚拟化技术。如果是全景渲染的话，可以使用裸金属服务+其它的云服务，这里的裸金属服务底层使用的就不是虚拟化技术，而是真正的物理服务器。

云计算是为用户提供随时随地可获取的 IT 服务，是一种商业模式或者服务模式，而虚拟化是一种技术手段，是云计算实现的重要手段之一。

主流的虚拟化技术有很多，一般分为开源和闭源两类。开源的虚拟化包括 KVM 和 Xen，闭源的虚拟化包括微软的 Hyper-v、VMware 的 vSphere、华为的 FusionSphere 等。

开源的技术是免费的，可以随时拿来用，它们的源代码是公开的，用户可以根据自己的需求定制一些特殊的功能。开源技术对使用者的技术要求很高，一旦系统出现问题，需要依靠自己的技术和经验来完成系统的修复。闭源的技术，用户是看不到源代码的，也不能进行个性化的定制，闭源的虚拟化产品一般都是收费的，为用户提供开箱即用的服务，在使用过程中，如果系统出现了问题，厂商会提供全程支持。

对于用户来说，开源和闭源没有好坏之分，只有哪种更适合。

在开源的虚拟化技术中，Xen 和 KVM 平分秋色，KVM 是全虚拟化，而 Xen 同时支持半虚拟化和全虚拟化。KVM 是 Linux 内核中的一个模块，用来实现 CPU 和内存的虚拟化，是 Linux 的一个进程，而其它的 I/O 设备（网卡、磁盘等）需要 QEMU 来实现。Xen 与 KVM 不同，它直接运行在硬件上，然后在其之上运行虚拟机，Xen 中的虚拟机分为两类：Domain0 和 DomainU，Domain0 是一个特权虚拟机，具有直接访问硬件和管理其它普通虚拟机 DomainU 的权限，在其它虚拟机启动前，Domain0 需要先启动。DomainU 是普通虚拟机，不能直接访问硬件资源，所有的操作都需要通过前后端驱动的方式转给 Domain0，再由 Domain0 完成具体的操作后将结果返回给 DomainU。

2.3 KVM 简介

华为的虚拟化产品在 R6.3 版本前是基于 Xen 开发的，从 R6.3 版本开始是基于 KVM 开发的。因此，接下来我们将简单介绍一下 KVM。

KVM，全称是 Kernel-based Virtual Machine（基于内核的虚拟机），是一种典型的 II 型全虚拟化，它之所以叫做基于内核的虚拟机，是因为 KVM 本身是一个 Linux 内核模块，当安装有 Linux 系统的物理机安装了这个模块后，就变成了 Hypervisor，而且还不会影响原先在该 Linux 上运行的其它应用程序，而且每个虚拟机都是进程，可以直接用 kill 命令杀掉。

一个普通的 Linux 安装了 KVM 模块后，会增加三种运行模式：

- Guest Mode：此模式主要是指虚拟机，包括虚拟机的 CPU、内存、磁盘等虚拟设备，该模式被置于一种受限的 CPU 模式下运行；
- User Mode：用户空间，此模式下运行的主要是 QEMU，它用来为虚拟机模拟执行 I/O 类的操作请求；
- Kernel Mode：内核空间，此模式下可以真正操作硬件，当 Guest OS 执行 I/O 类操作或特权指令操作时，需要向用户模式提交请求，然后由用户模式再次发起硬件操作请求给内核模式，从而真正操作硬件。

KVM 体系一般包括三部分：KVM 内核模块、QEMU 和管理工具，其中 KVM 内核模块和 QEMU 是 KVM 的核心组件，具体如下图 2-12 所示。

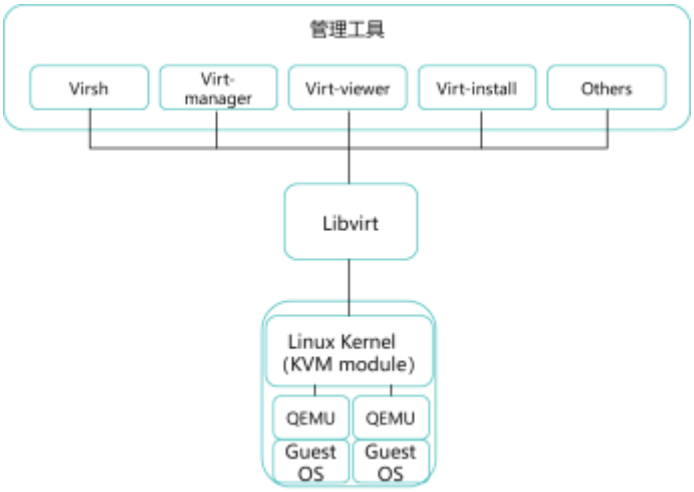


图 2-12 KVM 架构图

除了 KVM，其它的虚拟化产品几乎都是类似的架构。

KVM 内核模块是 KVM 虚拟机的核心部分，其主要功能是初始化 CPU 硬件，打开虚拟化模式，然后将 Geust Machine 运行在虚拟机模式下，并为虚拟客户机的运行提供一定的支持。

KVM 模块中，实现虚拟化功能的是 kvm.ko，还包括一个和处理器强相关的模块如 kvm-intel.ko 或 kvm-amd.ko，KVM 本身不能实现任何模拟功能，它仅仅是提供了一个/dev/kvm 接口，这个接口可被宿主机用来主要负责 vCPU 的创建、虚拟内存的地址空间分配、vCPU 寄存器的读写以及 vCPU 的运行。所以 kvm.ko 只提供了 CPU 和内存的虚拟化，但是一个虚拟机除

了 CPU 和内存外，还需要网卡、硬盘等其它的 I/O 设备，这时候就需要另外一个组件——QEMU 了，**KVM 核心模块和 QEMU 一起才能构成一个完整的虚拟化技术。**

其实 QEMU 原本不是 KVM 的一部分，它是一个通用的开源的使用纯软件来实现的虚拟化模拟器，Guest OS 以为自己在与硬件进行交互，其实真正交互的是 QEMU，然后再通过 QEMU 与硬件交互，这就意味着所有与硬件的交互都需要经过 QEMU，所以使用 QEMU 进行模拟的性能比较低。QEMU 本身可以模拟 CPU 和内存，在 KVM 中，只使用 QEMU 来模拟 IO 设备，KVM 的开发者将其改造成了 QEMU-KVM。

在 QEMU-KVM 中，KVM 运行在内核空间，QEMU 运行在用户空间，Guest OS 下发指令时，与 CPU 和内存相关的指令会通过 QEMU-KVM 中的/ioctl 调用/dev/kvm，从而将这部分指令交给内核模块来完成，从 QEMU 的角度来看，这样做也可以加速虚拟化。其它的 I/O 操作则由 QEMU-KVM 中的 QEMU 来实现，KVM 加上 QEMU 后才是完整意义上的虚拟化。

除了进行各种设备的虚拟化外，QEMU-KVM 还提供了原生工具来对虚拟机的创建、修改和删除等进行管理，Libvirt 是目前使用最为广泛的管理 KVM 虚拟机的工具和 API。

Libvirt 也是一个开源项目，它是一个非常强大的管理工具，被管理的虚拟化平台可以是 KVM，也可以是 Xen、VMware 以及 Hyper-V 等等。Libvirt 是一套由 C 语言开发的 API，其它的语言，比如 Java、Python、Perl 等，可以通过调用 Libvirt 的 API 去管理各个虚拟化平台。Libvirt 可以被很多应用使用，除了本身的 virsh 命令集外，Virt-manager、Virt-viewer、Virt-install 都可以通过 Libvirt 管理 KVM 虚拟机。

在云计算中，Hypervisor 种类众多，每个 Hypervisor 都有自己独特的管理工具，参数繁杂，难于使用，同时由于 Hypervisor 不统一，也没有统一的编程接口来对它们进行管理，对云计算的环境影响很严重。Libvirt 作为管理工具和 Hypervisor 的中间层，向下可以对接各种 Hypervisor，比如 KVM、Xen 等，向上可以提供各种语言的 API，而且它对上层的用户来说是完全透明的。

前面我们说过，QEMU 是一个实现 I/O 虚拟化的软件模拟工具，性能较差。例如，如果使用 QEMU 模拟一个 Windows 虚拟机网卡，我们在系统上看到该网卡的速率仅为 100M，当某些应用对网卡速率有要求时，就不能再使用 QEMU 了，我们需要引入一个新的技术——Virtio，用了 Virtio，模拟同样一个 Windows 虚拟机网卡，该网卡的速率可以提升到 10G。

我们先了解一下，没有 Virtio 时，虚拟机磁盘操作是如何实现的：

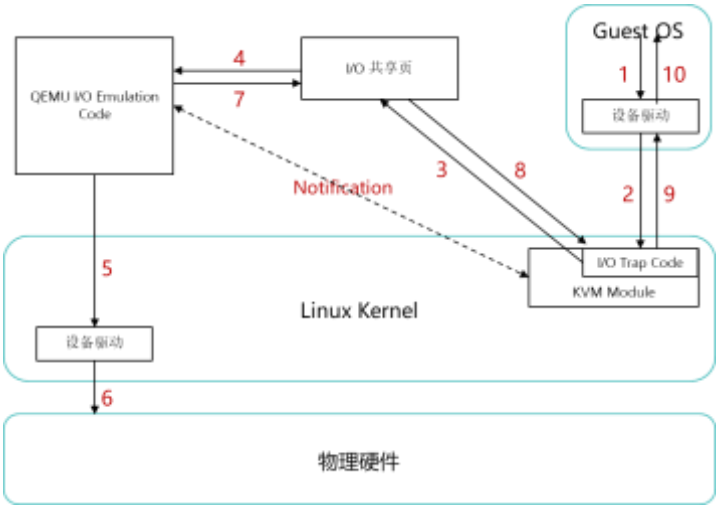


图 2-13 默认 I/O 操作流程

1 虚拟机中的磁盘设备发起一次 I/O 操作请求；

2 KVM 模块中的 I/O Trap Code（I/O 捕获程序）将这个 I/O 操作请求捕获到，进行相应的处理，然后将处理后的请求放到 I/O 共享页中；

3 KVM 模块会通知 QEMU，告诉它有新的 I/O 操作请求放到了共享页中；

4 QEMU 收到通知后，到共享页中获取该 I/O 操作请求的具体信息；

5 QEMU 对该请求进行模拟，同时根据 I/O 操作请求的信息调用运行在内核态的设备驱动，进行真正的 I/O 操作；

6 通过设备驱动对物理硬件执行真正的 I/O 操作；

7 QEMU 将执行后的结果返回到共享页中，同时通知 KVM 模块已完成此次 I/O 操作；

8 I/O 捕获程序从共享页中读取返回的结果；

9 I/O 捕获程序将操作结果返回给虚拟机；

10 虚拟机将结果返回给发起操作的应用程序。

注：注意第 2、3、7 步， KVM 其实除了捕获和通知，并没有对 I/O 操作做任何的修改，既然什么都没有修改，那么我们能不能把这一步去掉呢，所以就开发出了新的 virtio 技术。

使用 Virtio 时，具体的操作流程如下：

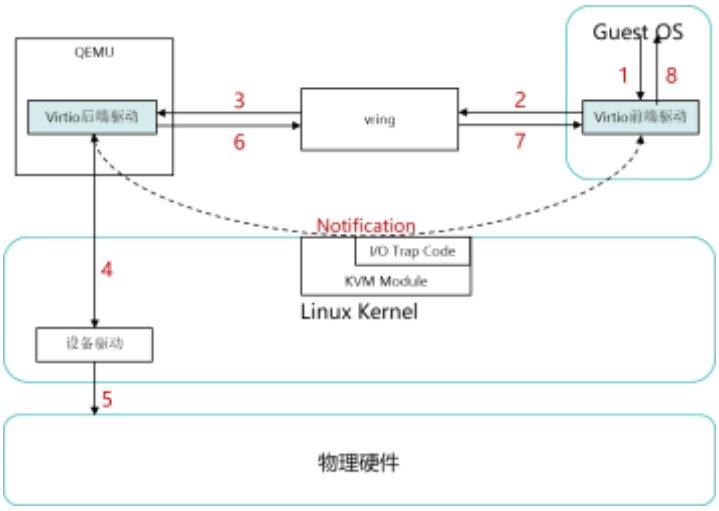


图 2-14 Virtio 下 I/O 操作流程

1 同样是由虚拟机发起 I/O 操作请求；

2 与使用默认模型不一样，这个 I/O 操作请求不会经过 I/O 捕获程序，而是直接以前后端的形式放到环形缓冲区，同时 KVM 模块通知后端驱动；

3 QEMU 到环形缓冲区获取到操作请求的具体信息；

4 后端驱动直接调用真实的物理设备驱动进行具体的 I/O 操作；

5 由真实的设备驱动完成 I/O 操作；

6 QEMU 将处理结果返回到环形缓冲区，并由 KVM 模块通知前端驱动；

7 前端驱动从环形缓冲区获取到此次 I/O 操作的结果；

8 前端驱动将结果返回给发起操作的应用程序。

通过对以上具体流程的介绍，我们可以得出如下使用 Virtio 的优点：

- 节省 QEMU 模拟时所需的硬件资源；
- 缩短 I/O 请求的路径，提高虚拟化设备的性能；

但是 Virtio 也存在一些缺点，比如有些比较老的或不常用的设备，无法使用 Virtio，只能使用 QEMU 方式进行模拟。

2.4 FusionCompute 简介

华为 FusionSphere 虚拟化套件是业界领先的虚拟化解决方案，能够帮助客户带来如下价值：

- 帮助客户提升数据中心基础设施的资源利用率。
- 帮助客户成倍缩短业务上线周期。
- 帮助客户成倍降低数据中心能耗。
- 利用虚拟化基础设施的高可用和强恢复能力，实现业务快速自动化故障恢复，降低数据中心成本，增加系统应用的正常运行时间。

FusionSphere 虚拟化套件通过在服务器上部署虚拟化软件，使一台物理服务器可以承担多台服务器的工作。通过整合现有的工作负载并利用剩余的服务器，以部署新的应用程序和解决方案，实现较高的整合率。

FusionCompute 是 FusionSphere 虚拟化套件中的必选组件，是云操作系统软件，主要负责硬件资源的虚拟化，以及对虚拟资源、业务资源、用户资源的集中管理。它采用虚拟计算、虚拟存储、虚拟网络等技术，完成计算资源、存储资源、网络资源的虚拟化。同时通过统一的接口，对这些虚拟资源进行集中调度和管理，降低业务的运行成本，保证系统的安全性和可靠性，协助运营商和企业构筑安全、绿色、节能的云数据中心。

FusionCompute 由两部分组成：CNA（Computing Node Agent，计算节点代理）和 VRM（Virtual Resource Manager，虚拟资源管理器）。除了 CNA 和 VRM，在其它资料上可能还会看到 UVP（Unified Virtualization Platform），UVP 是华为研发的统一虚拟化平台，它与 KVM 和 Xen 一样，也是一款 Hypervisor。FusionCompute 的 Hypervisor 使用裸金属架构，直接在硬件上安装虚拟化软件，将硬件资源虚拟化。由于使用了裸金属架构，FusionCompute 可为用户带来接近服务器性能的、高可靠和可扩展的虚拟机。

FusionCompute 的架构和 KVM 非常相似，其中 VRM 相当于 KVM 中的管理工具，管理员和用户可以通过图形化的 Portal 对 FusionCompute 进行管理和使用。它是基于 Linux 操作系统的，所以我们登录 VRM 后，很多 Linux 命令都可以使用。

VRM 主要提供以下功能：

- 管理集群内的块存储资源。
- 管理集群内的网络资源（IP/VLAN），为虚拟机分配 IP 地址。
- 管理集群内虚拟机的生命周期以及虚拟机在计算节点上的分布和迁移。
- 管理集群内资源的动态调整。
- 通过对虚拟资源、用户数据的统一管理，对外提供弹性计算、存储、IP 等服务。
- 通过提供统一的操作维护管理接口，操作维护人员通过 WebUI 远程访问 FusionCompute，对整个系统进行操作维护，包含资源管理、资源监控、资源报表等。

CNA 相当于 KVM 中 QEMU+KVM 模块，主要提供虚拟化功能，通常是以集群的方式部署，将集群内的计算、存储和网络资源虚拟化成资源池供用户使用。同样，CNA 也是基于 Linux 操作系统的。

CNA 主要提供以下功能：

- 提供虚拟计算功能。

- 管理计算节点上的虚拟机。
- 管理计算节点上的计算、存储、网络资源。

VRM 和 CNA 都有管理的作用，CNA 管理的是本节点上的虚拟机和资源，而 VRM 是从集群或者整个资源池的层面进行管理。如果 VRM 对某个虚拟机进行修改或者其它生命周期的操作时，需要将命令下发给 CNA 节点，再由 CAN 去执行。操作完成后，CNA 再把结果返回给 VRM，由 VRM 记录到数据库中。所以尽量不要到 CNA 上执行虚拟机或其它资源的修改操作，以避免造成 VRM 数据库中的记录与实际不匹配的情况。

FusionCompute 除了支持华为自己的硬件产品外，还支持其它基于 x86 硬件平台的多种服务器，并兼容多种存储设备，可供企业灵活选择。目前单个集群最大可支持 64 个主机，3000 台虚拟机。FusionCompute 具有完善的权限管理功能，可根据不同的角色、权限等，提供完善的权限管理功能，授权用户对系统内的资源进行管理。

本课程将以 FusionCompute 为例，进行虚拟化功能和特性的实验，具体请参考对应的实验手册。

根据实验手册安装完 FusionCompute，我们要验证如下几个问题：

1 前面说过 FusionCompute 6.3 是基于 KVM 开发的，那么到底是不是呢?

2 如果是基于 KVM 开发的，那么有没有用到 QEMU 和 Libvirt 呢?

3 云计算中的网络基础知识

网络是由各种网络设备组成，在传统 IT 中，网络设备几乎都是物理设备，是可以真实看到的，大部分的流量是可控的，比如两个交换机上的主机需要通信，必须使用网线或者光纤将两个交换机连起来。到了云计算中，除了传统的物理网络设备外，还有很多网络设备是虚拟化的，运行在服务器内部，打通虚拟网络设备的不再是真实的网线，有可能是转发表里的一条条目，所以管理员面对的是前所未有的挑战，本章节主要介绍云计算中与网络相关的物理设备和虚拟设备的知识。

3.1 虚拟化中的网络架构

3.1.1 虚拟化中的网络流量

相对于传统 IT，在云计算和虚拟化中，数据中心的流量分为南北向流量和东西向流量，判断某流量是南北向还是东西向，需要一定的参照物。一般情况下，南北向流量和东西向流量是以路由器为分界点，无论是物理路由器或者虚拟路由器都适用，经过路由器的流量为南北向流量，不经过路由器的流量为东西向流量。以物理路由器为例，如下图 3-1 所示，该路由器被部署在 IDC 机房的边界处，向上连接外网（可以是互联网，也可以是企业自己定义的外部网络），向下连接 IDC 机房的业务网络（如邮件系统、办公系统等），当外网访问 IDC 机房业务时，该流量为南北向流量；如果 IDC 机房内运行了员工的个人虚拟机，该虚拟机访问业务时，不需要经过路由器，同样是访问业务的流量，但该流量为东西向流量。

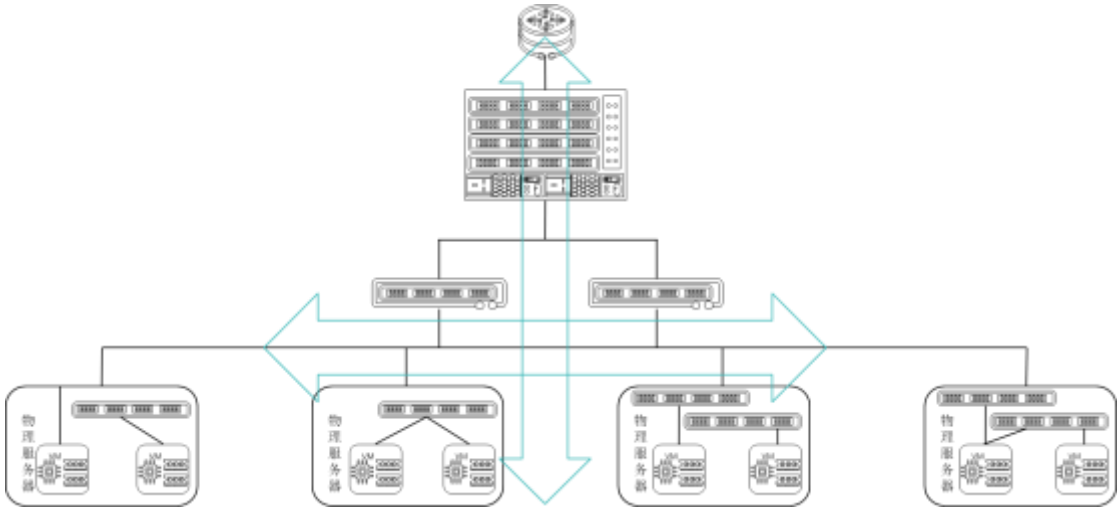


图 3-1 云计算流量示意图

随着云计算的发展，越来越多的运算都集中在 IDC 机房的集群中，有时候客户只发送了一条指令，大量的运算就会在 IDC 的不同业务之间进行，如果使用桌面云的话，客户端甚至也运行在 IDC 机房，以及虚拟化本身的高可用特性使虚拟机可以自由地在数据中心不同的物理服务器上迁移，从而导致东西向流量越来越多，从之前的大约 20%发展到现在的大约 70%。这种情况也带来了网络结构的变化，以前采用的是核心层、汇聚层和接入层的架构，实践证明这种架构已经无法满足和适应云计算的部署。因此，三层的网络架构发展成了大二层架构，大二层可以很好地满足东西向流量较大的要求。

虚拟机也是构成网络的一部分，它们几乎都是通过网桥接入到网络中，为了更方便管理和配置，大多数情况下，我们会使用虚拟交换机，它是网桥的一种高级体现形式，后面再单独详细介绍。一个虚拟机可以有多个虚拟网卡，通过虚拟网卡，虚拟机可以与一台虚拟交换机相连，也可以同时与多个虚拟交换机相连。

3.1.2 网络基本概念

- 广播和单播

广播和单播都是网络中通讯的方式，除了广播和单播，还有一种方式叫做组播，在本课程中暂时不讨论。

广播，顾名思义就是“一个人说，剩余的人听”。在网络中，两台设备第一次进行通信时，通信的发起方会使用广播的方式来找通信的接收方，该广播会在整个广播域中扩散，同域内的所有网络设备都会接收到该广播，同时会检查这个广播数据包中的内容，如果发现广播中的接收方是自己，则会给发起方回一个单播的消息，如果发现接收方不是自己，则会将这个广播包丢弃。

除了第一次通信时需要使用广播，还有很多应用使用的也是广播，比如 DHCP 等，不同的业务之间使用不同的广播地址来区分，比如在 192.168.1.0/24 网段中，广播地址为 192.168.1.255，再比如 DHCP 客户机在一开始搜索 DHCP 服务器时使用的广播地址为 255.255.255.255.

再通俗点讲，拿村委的大喇叭打比方，十几年前，村委一般会通过广播站进行找人或发布一些消息，如果是找人大多是广播站工作人员在话筒前讲，这时的广播地址就是讲话人的声音；如果有重大消息需要村长亲自讲，这时的广播地址就是村长的声音。不同的事由不同的人广播，发出的声音也不一样。

与广播不同，单播是“一个人说，另外一个人听”。在真实的网络中，单播是指网络设备进行一对一的通信，如果一台设备进行消息的发送，另外一台只负责接收，叫做半双工，如果两台设备同时在进行发送和接收，叫做全双工。

网络上绝大部分的数据都是以单播的形式进行传输的，比如我们在发送邮件、查看网页以及玩网络游戏时，都需要先与邮件服务器、网站 web 服务器和游戏服务器建立联系。

网络设备进行广播时，通常不会包含很多信息在数据包中，整个数据包中有用的信息可能只有源地址和目的地址，而网络设备进行单播时，会在数据包中放入对通信双方有价值的信息，所以，如果网络中充斥了太多广播包的话，在带宽一定的情况下，会造成单播的堵塞，反应到用户侧，用户就会感觉上网网页打不开，邮件发送不出去，或者在打游戏时老掉线。除了占用带宽外，广播也存在着一定的安全隐患，由于广播包所有人都可以看到，如果有人利用广播包中的信息进行伪装和冒充，有可能造成信息泄露或者网络瘫痪。但是广播是进行单播前的必要条件，不能被禁止。因此，网络设计人员就想到，广播只在广播域内有效，所以能不能把广播域划小点

呢？这样的话，网络堵塞和安全问题就都可以被缓解。那么，广播域之间通信使用什么呢？接下来我们看一下路由和默认网关。

- 路由和默认网关

早些年，手机还没有普及的时候，我们大多使用固定电话通讯，如果是长途电话需要在拨号前加区号，有时还需要查黄页或其它材料去找对应的区号，区号的作用是进行号码路由，让通话能够到达对应的区市，而用来查询区号的黄页就相当于是路由表。如果把广播域之间的通信看做为打长途电话的话，那么拨电话的人通过黄页找到对方的电话号码的方式就叫路由。

如果广播域较多，那么路由表里的条目也会很多，每次通信时都需要进行路由的查找，会给保存路由表的设备带来负担，也会影响网络通信的效率，这时就会用到默认网关。默认网关的作用和默认路由是一样的，还是拿打电话举例，以前我们有个 114 查号台，如果不知道对方的电话号码，可以打 114 进行查号获取对方的电话号码，默认网关的作用与 114 类似，但也有不同的地方，114 会返回给查号者一个电话号码，然后由查号者再次拨打电话，而默认网关收到通信请求时，如果自己的路由表中存在目的地址的网段，则会替通讯的发起者进行路由的转发，如果自己的路由表中没有目的地址，它会给发起者返回一个目的地址不可达的消息。

默认网关是路由的一种特殊形式，它是路由转发时的最后选择，如果没有其它的路由条目进行转发，则使用默认网关进行转发。

- VLAN

解决了广播域之间通信的问题，那么该如何将广播域划小呢？经常使用的技术是 VLAN（Virtual Local Area Network，虚拟局域网）。VLAN 是将一个物理的 LAN 在逻辑上划分成多个广播域的通信技术。同一个 VLAN 内的主机间可以直接通信，而不同 VLAN 内的主机间不能直接通信，从而将广播报文限制在一个 VLAN 内。具体如下图 3-2 所示：

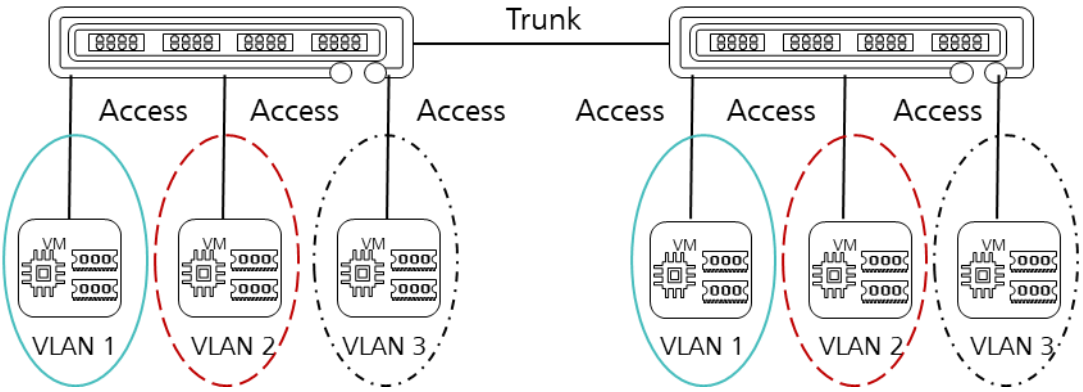


图 3-2 VLAN 的作用

使用 VLAN，可以带来以下好处：

- 限制广播域：广播域被限制在一个 VLAN 内，节省了带宽，提高了网络处理能力。
- 增强局域网的安全性：不同 VLAN 内的报文在传输时是相互隔离的，即一个 VLAN 内的用户不能与其它 VLAN 内的用户直接通信。
- 提高了网络的健壮性：故障被限制在一个 VLAN 内，本 VLAN 内的故障不会影响其它 VLAN 的正常工作。
- 灵活构建虚拟工作组：用 VLAN 可以将不同的用户划分到不同的工作组，同一工作组的用户也不必局限于某一固定的物理范围，网络构建和维护更方便灵活。

简单介绍一下 VLAN 的工作原理。VLAN 是在传统的以太网数据帧中加入了 4 字节的 802.1Q Tag，不同的 VLAN 就用这个标签来区分。具体如下图 3-3 所示：

6bytes	6bytes	2bytes	46-1500bytes	4bytes
Destination address	Source address	Length/Type	Data	FCS

图 3-3 传统数据帧格式

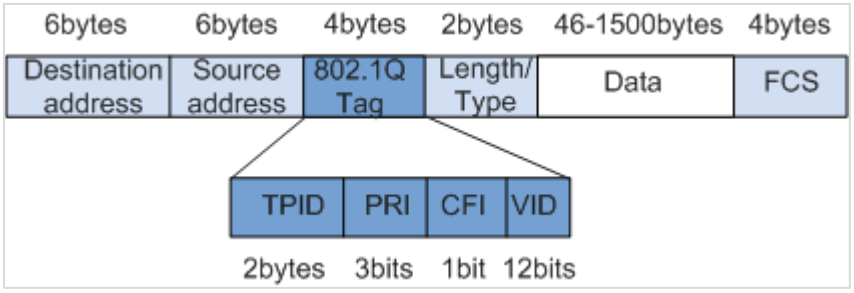


图 3-4 加入 Tag 后数据帧格式

每台支持 802.1Q 协议的交换机发送的数据包都会包含 VLAN ID，以表明交换机属于哪一个 VLAN。因此，在一个 VLAN 交换网络中，以太网帧有以下两种形式：

- 有标记帧（tagged frame）：加入了 4 字节 802.1Q Tag 的帧
- 无标记帧（untagged frame）：原始的、未加入 4 字节 802.1Q Tag 的帧

操作系统或交换机的端口都有为数据帧加入 Tag 标签的功能，一般情况下，这个标签的添加和去除都由交换机完成，所以 VLAN 中有以下两种链路类型：

- 接入链路（Access Link）：用于连接用户主机和交换机的链路。通常情况下，主机并不需要知道自己属于哪个 VLAN，主机硬件通常也无法识别带有 VLAN 标记的帧。因此，主机发送和接收的帧都是 untagged 帧。
- 干道链路（Trunk Link）：用于交换机间或交换机与路由器之间的连接。干道链路可以承载多个不同 VLAN 的数据，数据帧在干道链路传输时，干道链路两端的设备需要识别数据帧属于哪个 VLAN，所以在干道链路上传输的帧都是 Tagged 帧。

802.1Q 定义了 VLAN 帧后，设备的有些接口可以识别 VLAN 帧，有些接口则不能识别 VLAN 帧。根据接口对 VLAN 帧的识别情况，可分为两类：

- Access 接口：Access 接口是交换机上用来连接用户主机的接口，它只能连接接入链路。仅允许唯一的 VLAN ID 通过本接口，这个 VLAN ID 与接口的缺省 VLAN ID 相同，Access 接口发往对端设备的以太网帧永远是不带 Tag 标签的帧。

- Trunk 接口：Trunk 接口是交换机上用来与其它交换机连接的接口，它只能连接干道链路，允许多个 VLAN 的数据帧（带 Tag 标签）通过。

每种类型的接口都可以配置一个缺省 VLAN，对应的 VLAN ID 为 PVID（Port Default VLAN ID)。接口类型不同，缺省 VLAN 的含义也有所不同。几乎所有交换机出厂时的默认 VLAN 都为 1。

各类型接口对数据帧的处理方式如下表：

表 1 数据帧处理方式

接口类型	对接收不带Tag的报文处理	对接收带Tag的报文处理	发送帧处理过程
Access接口	接收该报文，并打上缺省的VLAN ID。	当VLAN ID与缺省VLAN ID相同时，接收该报文。 当VLAN ID与缺省VLAN ID不同时，丢弃该报文。	先剥离帧的PVID Tag，然后再发送。
Trunk接口	打上缺省的VLAN ID，当缺省VLAN ID在允许通过的VLAN ID列表里时，接收该报文。 打上缺省的VLAN ID，当缺省VLAN ID不在允许通过的VLAN ID列表里时，丢弃该报文。	当VLAN ID在接口允许通过的VLAN ID列表里时，接收该报文。 当VLAN ID不在接口允许通过的VLAN ID列表里时，丢弃该报文。	当VLAN ID与缺省VLAN ID相同，且是该接口允许通过的VLAN ID时，去掉Tag，发送该报文。 当VLAN ID与缺省VLAN ID不同，且是该接口允许通过的VLAN ID时，保持原有Tag，发送该报文。

下面我们以华为交换机为例，来具体看一下数据帧的处理过程。

- 配置一：

```
Port link-type access
Port default vlan 10
```

配置一表明，该接口属于 access 类型，此接口上的默认 VLAN 为 10，如果带 VLAN 10 标签的数据帧经过该端口时，会被剥离标签进行转发；如果经过的数据帧不带标签，会被打上 VLAN 10 的标签，然后进行转发；如果数据帧带的标签不是 VLAN 10，则会被丢弃。

- 配置二：

```
Port link-type trunk
Port trunk pvid 10
Port trunk allow-pass vlan 16 17
```

配置二表明，该接口属于 trunk 类型，此接口上的默认 VLAN 为 10，允许通过的 VLAN 为 16 和 17，如果经过该端口的数据帧不带任何标签，会被打上 VLAN 10 的标签；如果经过的是带有 VLAN 10 标签的数据帧，则会被去掉标签进行转发；如果经过的数据帧带有 VLAN 16 和 17 的标签，会不做任何修改进行转发；如果非以上情况，则数据帧会被丢弃。

VLAN 除了可以基于交换机端口划分，还可以基于 MAC、子网和策略划分，另外还可以在同一个 VLAN 中继续划分 VLAN，这些都属于 VLAN 的高级功能，我们在这里就不再介绍了，如果有兴趣的可以关注华为数通方向的认证课程。

3.2 虚拟化中的物理网络介绍

在虚拟化中，承载业务的都是虚拟机，而虚拟机运行在物理服务器内部，如何使虚拟机接入到网络中，首先要解决的是将物理服务器连接到网络中，在这个过程中，我们需要用到的设备有路由器、三层交换机、二层交换机以及服务器自身的网卡。

在讲物理设备之前，我们先简单了解一下 TCP/IP 四层模型。

在讲互联网发展时，我们说过，TCP/IP 是目前网络中最通用的协议栈，它把整个网络通信过程分为四层，分别是应用层、传输层、网络层和链路层，如下图 3-5 所示。

图 3-5 TCP/IP 协议栈

其中，我们前面提到的路由工作在传输层（三层），VLAN 工作在网络层（二层），所以如果某台设备具备了路由功能，可以查看路由表，我们就把它当成三层设备，如果仅可以划分 VLAN，就把它当成二层设备，比如 Hub，看上去和交换机的作用一样，但它不能划分 VLAN，只能起到分线器的作用，所以它只是一个链路层（一层）设备。工作在传输层的设备有路由器和三层交换机，工作在网络层的设备一般是二层交换机，而物理服务器的网卡、连接网卡的网线以及光纤等线路则属于链路层的设备。

路由器和三层交换机都工作在传输层，三层交换机也具备路由的功能，并且很多路由器加上交换板卡后也可以划分 VLAN，那么，三层交换机和路由器是否可以互相取代呢？答案肯定是否定的。

首先，交换机和路由器的作用不一样，交换机通过 ASIC 芯片负责高速的数据交换，而路由器通过维护路由表进行不同网段之间的路由寻址，同时其天生具备隔离广播域的特性。但就算路由器具备了交换功能，或者交换机具备了路由功能，它们最主要的功能仍然没有变，新增加的功能只不过是一个附加功能，不能作为主要用途来使用。

第二，交换机主要适用于局域网，路由器一般适用于广域网。局域网的特点主要是数据交换频繁、网络接口单一、数量较大，交换机能够提供快速的数据转发，一般提供网线接口（RJ45）和光纤接口两种，并且每台交换机一般都有较多接口，完全可以满足局域网的需求。对广域网来说，网络类型和接口类型众多，路由器的路由功能通常非常强大，不仅适用于同种协议的局域网间，更适用于不同协议的局域网与广域网间。它的优势在于选择最佳路由、负荷分担、链路备份及与其它网络进行路由信息的交换等等路由器所具有的功能。

第三，三层交换机和路由器的性能不一样。从技术上讲，路由器和三层交换机在数据包交换上存在明显区别。路由器一般由基于微处理器的软件路由引擎执行数据包交换，而三层交换机通过硬件执行数据包交换。三层交换机在对第一个数据流进行路由后，会产生一个 MAC 地址与 IP 地址的映射表，当同样的数据流再次通过时，将根据此表直接从二层通过而不会再次路由，从而避免了路由器进行路由选择而造成的网络延迟，提高了数据包转发的效率。

同时，三层交换机的路由查找是针对数据流的，它利用 ASIC 技术的缓存机制，可以很容易实现快速转发，大大节约成本。而路由器的转发采用最长匹配的方式，通常使用软件来实现，实现方式复杂，转发效率较低。 因此，从整体性能上比较，三层交换机的性能要远优于路由器，非常适用于数据交换频繁的局域网；而路由器虽然路由功能非常强大，但它的数据包转发效率远低于三层交换机，更适用于不同网络类型但数据交换不是很频繁的互联，如局域网与互联网的互联。如果把路由器，特别是高端路由器用于局域网中，在相当大程度上可以说是一种浪费（就其强大的路由功能而言），而且还不能很好地满足局域网通信性能的需求，影响子网间的正常通信。

在云计算和虚拟化中，路由器一般部署在企业或单位网络出口的位置，连接互联网和内网。如果内网需要访问互联网，一般都需要路由器完成路由转发和 NAT（网络地址转换），如果通过互联网访问内部网络，同样也需要经过路由器。

路由器解决了内部网络与互联网的通信问题，剩下就是内部网络的规划了。我们的最终目的是将物理服务器接入到网络中，在数据中心机房，服务器放置在独立的机柜中，为了更好的制冷效果，机柜会按列摆放，所以目前最主流的两种服务器接入网络的方式分别是 ToR（Top of Rack，柜顶接入）和 EoR（End of Row，列头接入）。ToR 和 EoR 都是描述机柜布线的专有名词。ToR，顾名思义，就是将连接服务器接入网络的交换机放在机柜顶部，如果机柜中的服务器密度较高且流量较大，则会在每个机柜都放一个 ToR 交换机，如果密度和流量都一般，则会考虑几个机柜共用一个 ToR 交换机。ToR 交换机在选型时可以根据实际情况考虑使用千兆或者万

兆的设备。ToR 交换机下联服务器网口，上联汇聚或核心交换机，从而将物理服务器连接到网络。ToR 布线方式如下图 3-5 所示。

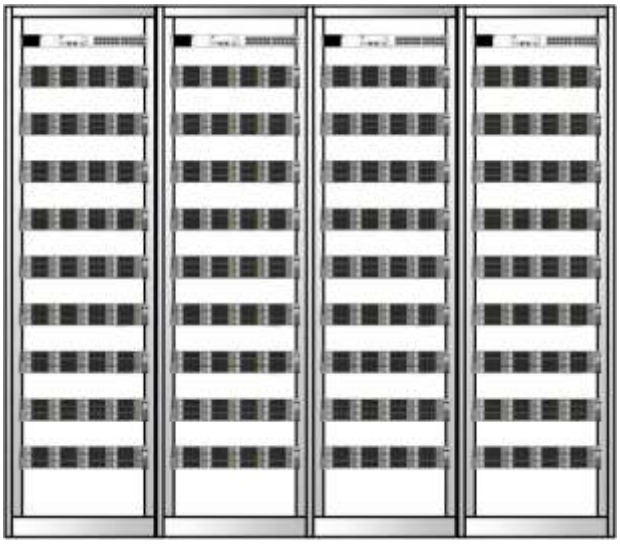


图 3-6 ToR 示意图

EoR 全称叫 End of Row，Row 指的是“列”，EoR 的布线方式与 ToR 不同，是在一列机柜中单独放置用来连接服务器接入网络的交换机，这个交换机也叫 EoR 交换机。虽然 EoR 中的“E”指的是“End”，但很多情况下，为了减少服务器到交换机的布线长度，一般会将 EoR 交换机放在一列机柜的中间，而其它机柜会预留好网线配线架和光纤配线架，服务器的网口直接连接到配线架上，然后再跳接到 EoR 交换机上。具体如下图 3-7 所示：

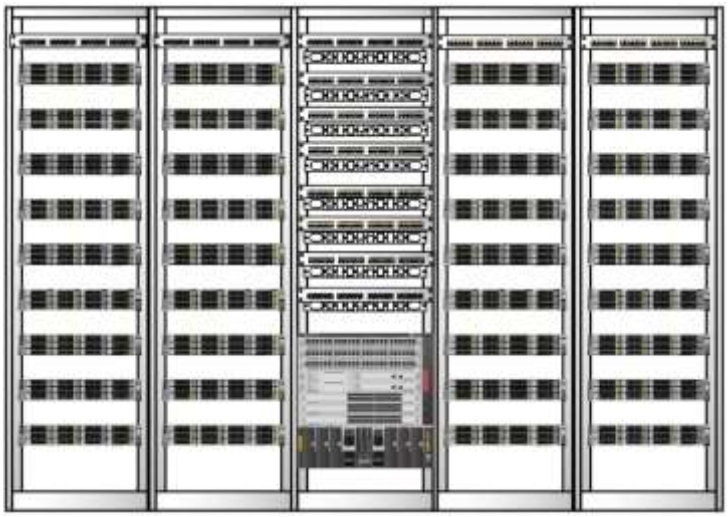


图 3-7 EoR 示意图

ToR 和 EoR 都有自己的使用场景，也有一定的局限性。

EoR 布线方式的缺点：从服务器机柜到网络机柜的线缆需要提前布放好，所以对机房服务器的设计要求较高，包括每个机柜中有多少台服务器、每个服务器有多少个网口以及可能会有的网口类型等等，且距离网络机柜越远的服务器机柜，其布线线缆越长。同时，为了机房的整洁，线缆都需要捆扎好，一旦线缆数量与实际数量不匹配或线缆出现故障，就需要重新布线，因此线缆管理维护工作量大、灵活性差。

ToR 布线的缺点：每个服务器机柜中电源的输出功率是一定的，那么可部署的服务器数量就会受到限制，所以可能只会用到交换机的 12-16 个端口，可是目前绝大数交换机至少有 24 个端口，因此导致机柜内交换机的接入端口利用率较低。如果几个服务器机柜共用 1-2 台接入交换机，虽然可解决交换机接入端口利用率低的问题，但这种方式就相当于是小型的 EoR 方式，增加了线缆管理和维护的工作。

在设计网络综合布线时，最好可以结合两种方式的优缺点，找一个最适合当前情况的解决方案。

服务器接入到网络后，我们可以根据网络流量的用途来分类，一般会分为业务流量、管理流量和存储流量。业务流量和存储流量对用户来说比较重要，用户通过业务流量来访问所需的业务，如果业务数据没放在服务器本地，而是放在了专业的存储设备上，那么服务器访问存储设备时就会产生存储流量。管理流量主要是用户用来管理服务器、虚拟机及存储设备等时产生的流量。现在几乎每个物理设备都会单独配一个管理口，如果管理流量和业务流量分开，使用不同的物理线路及接口，这种方式叫带外管理（out-of-band），如果管理流量和业务流量使用同一物理通道，就叫带内管理（in-band）。

在云计算数据中心，设计网络时，会使用高端的三层交换机作为整个网络的核心，所有流量所对应网段的默认网关全部设置在上面，这意味着所有跨广播域相互访问的流量都会经过该交换机。这样做的原因有以下几点：

- 高端三层交换机有很高的转发性能，可以满足全网流量的转发要求。

- 高端三层交换机使用模块化设计，可保证自身的高容错能力和易扩展性。除了如电源、风扇等必要模块外，高端三层交换机的核心部件——引擎板采用 1+1 热备的方式大大提升了设备的可用性；交换板卡支持热插拔，方便用户随时对网络进行扩容。
- 高端三层交换机可以提供多种接口密度的板卡，比如 10G、40G、100G 等，能够支持大容量的高密服务器接入和 ToR 上行汇聚，满足数据中心网络高性能、超大容量的要求。
- 高端三层交换机除了可以提供基本的路由交换功能外，还支持其它更符合云计算要求的特性，比如大二层、堆叠以及虚拟化等。

所有流量在接入核心交换机前，一般先要接入二层交换机，接入的方式我们前面讲过——EoR 和 ToR。按照接入流量的类型，接入交换机可以分为管理交换机、存储交换机和业务交换机。如果是超大流量的数据中心，在设计网络结构时，建议使用不同的物理交换机来承载不同流量，也就是说，每种流量都使用单独的交换机，如下图 3-8 所示。如果是一般流量的数据中心，可以使用同一个物理交换机并利用 VLAN 对不同流量进行逻辑隔离。

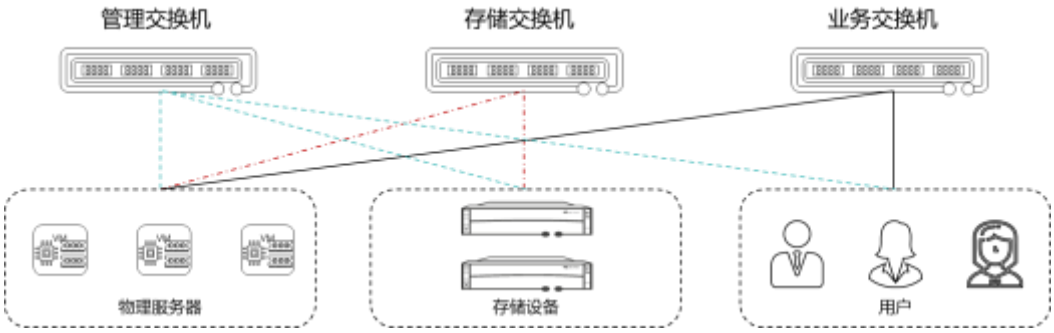


图 3-8 接入层交换机

物理服务器通过自身的物理网卡连接到网络中，所有虚拟机的流量都是通过各种类型的网口进入整个网络。关于物理网卡，有一个很关键的知识点——端口（链路）聚合。端口聚合通过将多条以太网物理链路捆绑在一起成为一条逻辑链路，实现增加链路带宽的目的，同时，这些捆绑在一起的链路通过相互间的动态备份，可以有效提高链路的可靠性。

随着网络规模的不断扩大，用户对骨干链路的带宽和可靠性提出越来越高的要求。在传统技术中，一般是通过更换高速率的设备的方式来增加带宽，但这种方案需要付出高额的费用，而且不够灵活。

链路聚合技术可以在不进行硬件升级的情况下，通过将多个物理接口捆绑为一个逻辑接口，来达到增加链路带宽的目的。并且在增大带宽的同时，链路聚合采用备份链路机制，可以有效提高设备间链路的可靠性。

链路聚合技术主要有以下三个优势：

- 增加带宽

链路聚合接口的最大带宽可以达到各成员接口的带宽之和。

- 提高可靠性

当某条活动链路出现故障时，流量可以切换到其它可用的成员链路上，从而提高聚合链路的可靠性。

- 负载分担

在一个链路聚合组内，可实现各成员活动链路间的负载分担。

根据是否启用 LACP（Link Aggregation Control Protocol，即链路聚合控制协议），链路聚合分为手工负载分担模式和 LACP 模式。

手工负载分担模式下，链路的创建、成员接口的添加由手工配置，不需要链路聚合控制协议的参与。该模式下所有活动链路都参与数据的转发，平均分担流量，因此称为负载分担模式。如果某条活动链路故障，链路聚合中的剩余活动链路会自动平均分担流量。当需要在两个直连设备间提供一个较大的链路带宽而设备又不支持 LACP 协议时，可以使用手工负载分担模式。

手工负载分担模式无法检测到链路层故障、链路错连等，为了能够提高链路的容错性及备份功能，保证成员链路的高可靠性，有了链路聚合控制协议 LACP，LACP 模式就是采用 LACP 的一种链路聚合模式。LACP 为交换数据的设备提供了一种标准的协商方式，以供设备根据自身配

置自动形成聚合链路并启动聚合链路来收发数据。形成聚合链路后，LACP 负责维护链路状态，在聚合条件发生变化时，自动调整或解散链路聚合。

3.3 虚拟化中的虚拟网络介绍

3.3.1 虚拟网络的架构

随着云计算和虚拟化的普及，真正接入网络层的已经不再是以前的二层交换机了，它需要进入到服务器内部，与虚拟机进行对接，虚拟交换机就应声而出，成为真正接入网络层的设备。

我们在前面讲了云计算和虚拟化的优势，正是这些优势，使云计算和虚拟化变的越来越普及，并成为目前主流的 IT 技术，但随着新技术的诞生也出现了一些新的问题，虚拟化越来越普及，物理服务器不再是业务的承载体，以前物理服务器至少会有一根网线连接到交换机上，运行在此服务器上的业务独享这根网线，而现在，一台物理服务器内运行了多台虚拟机，它们会共享这一根网线，这样一来这根网线就会承载多种流量，如何管理这些流量？如何查看这些流量的状态？这些都成了新的问题。

此外，以前我们的岗位分为主机工程师、网络工程师和程序开发工程师，大家各司其职，都有明显的责任划分，而现在虚拟交换机虽然属于网络工程师维护的设备，但它却运行在服务器内部，服务器内部按照以前的划分应该由主机工程师负责。因此，在云计算和虚拟化中，一旦出现了虚拟机交换机的问题，就会比较尴尬，因为网络工程师想管却管不到，主机工程师能管却管不了。这些情况都是因为虚拟交换机成为了真正的网络接入设备造成的。这时候，无论是主机工程师还是网络工程师，都非常有必要了解虚拟化中的网络架构。

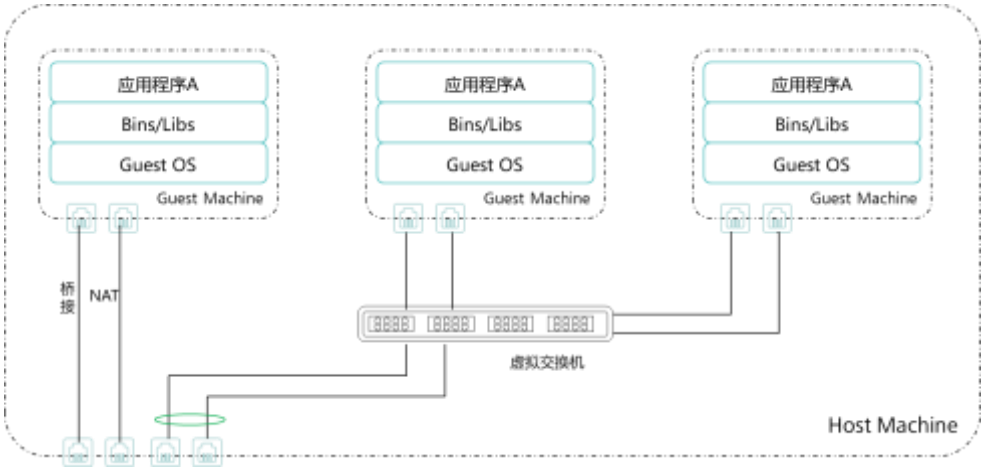


图 3-9 虚拟网络架构

如上图 3-9 所示，平时我们使用的虚拟化大多是这个架构，在个人或者小型的虚拟化中，虚拟机会以桥接或者 NAT 的方式与物理网卡绑定，而在企业级的大规模场景下，虚拟机都是通过虚拟交换机连接到物理网络的。

网桥不是一个新技术，简单来说，网桥就是把一台机器上的若干个网络接口“连接”起来，其中一个网口收到报文时会复制给其它网口，这样网口之间就能够正常通信了。

在虚拟化中，将所有网口连接到一起的工作由操作系统完成，如下图 3-10 所示，为 Linux 的桥接示意图：

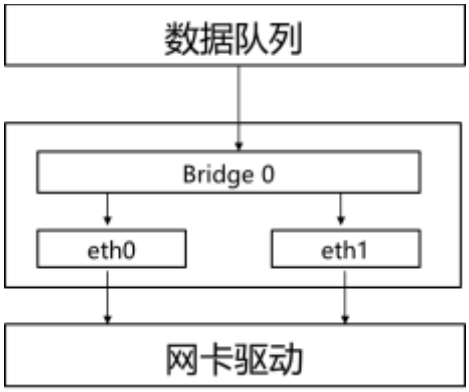


图 3-10 Linux 桥接示意图

网桥设备 Bridge0 绑定了 eth0 和 eth1，对于上层的网络协议栈来说，只能看到 Bridge0，因为桥接是在数据链路层实现的，上层并不需要关心桥接的具体细节。于是上层协议栈将需要发送的报文发送到 Bridge0，由网桥来判断报文该转发到 eth0 还是 eth1，或者两者都

转发；反过来，从 eth0 或 eth1 接收到的报文被转发给网桥，由网桥来判断该报文该转发、丢弃、或发送给上层协议栈。

网桥主要有以下两个重要功能：

1． MAC 学习：学习 MAC 地址，起初，网桥是没有任何地址与端口的对应关系的，它发送数据，还是得像 HUB 一样，但是每发送一个数据包，它都会关心对应源 MAC 地址的数据包是从自己的哪个端口来的，通过学习建立地址-端口的对照表（CAM 表）。

2． 报文转发：每发送一个数据包，网桥都会提取它的目的 MAC 地址，再从地址-端口对照表（CAM 表）中查找由哪个端口把数据包发送出去。

当有了虚拟化后，每个虚拟机都会有一个虚拟网卡，Linux 操作系统会在用户态生成一个 TAP 设备，同时，在内核态生成一个 tun/tap 的设备驱动和虚拟网卡驱动。虚拟机发出的数据包经过用户态的 tun/tap 字符设备文件，然后经过内核态的 tun/tap 设备驱动和虚拟网卡驱动，发给 TCP/IP 协议栈，再转发到用户态的虚拟网卡 tap，tap 此时直接连在网桥 Bridge0 上，上图中的 eth0 和 eth1 就会变成虚拟机的网卡 tap，具体如下图 3-11 所示：

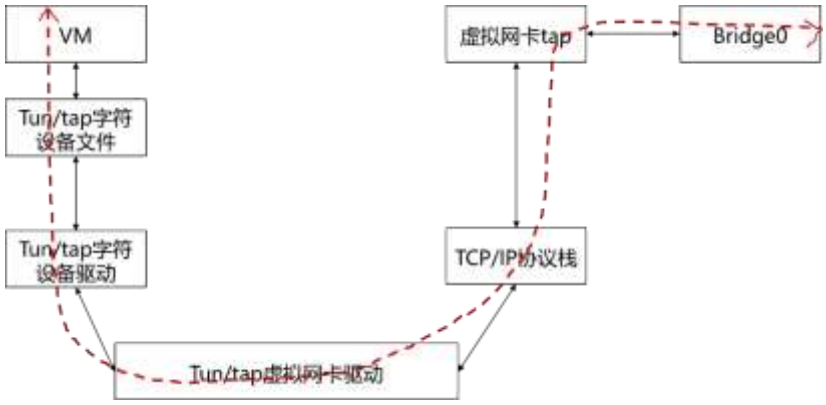


图 3-11 网桥流量转发

注：我们到 CNA 中查看一下 tap 设备，登录 CNA 后输入 ifconfig *命令，在最后会看到两个 tap 的配置文件，具体如下：*

```
tap00000001.0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       ether fe:6e:d4:88:b5:06  txqueuelen 5000  (Ethernet)
       RX packets 2633199  bytes 662047312 (631.3 MiB)
       RX errors 0  dropped 0  overruns 0  frame 0
       TX packets 3682209  bytes 1012247487 (965.3 MiB)
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tap00000002.0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether fe:6e:d4:88:c6:29  txqueuelen 5000  (Ethernet)
        RX packets 9  bytes 2102 (2.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35  bytes 2202 (2.1 KiB)
        TX errors 0  dropped 514913 overruns 0  carrier 0  collisions 0
```

我们可以看到两个 tap 设备的 MAC 地址分别为 fe:6e:d4:88:b5:06 和 fe:6e:d4:88:c6:29，*登录到 VRM 页面，查看当前虚拟机的 MAC 地址，可以发现它们是对应的，如下 3-12 所示:*

图 3-12 虚拟机 MAC 地址

如果仅使用网桥，虚拟机与外部通讯时有两种方式——桥接和 NAT。简单地说，如果是桥接，网桥就相当于是一个交换机，虚拟网卡连接的是交换机的一个端口；如果是 NAT，网桥就相当于是一个路由器，虚拟网卡连接的是路由器的一个端口。

连接在交换机上比较容易理解，虚拟网卡和网桥使用广播通信，使用相同的 IP 地址配置。如果连接的是路由器，虚拟网卡和网桥的 IP 地址不是同一网段，系统会自动生成一个网段地址，虚拟网卡与其它网络包括网桥通讯时使用三层路由转发，并且在网桥上进行地址转换。在 Linux 中，默认生成的网段是 192.168.122.0/24，如下图 3-13 所示:

```
virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:cb:7e:97  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vnet0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::fc54:ff:feeb:e118  prefixlen 64  scopeid 0x20<link>
        ether fe:54:00:eb:e1:18  txqueuelen 1000  (Ethernet)
        RX packets 685272  bytes 74578135 (71.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1232006  bytes 1260248954 (1.1 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 3-13 Linux 中 NAT 网桥地址

地址转换使用的是 NAT 技术。使用 NAT 后，当虚拟机与外部网络进行通信时，经过 NAT 网关，也就是上图的 virbr0 时，会将 IP 包中的源 IP 地址转换成物理网桥的地址，并且会有一个记录，当外部网络访问虚拟机时，NAT 网关会根据这个记录将数据包转发给对应的虚拟机。

NAT 是一种应用非常广泛的技术，使用 NAT 主要有以下优势：

- 当物理网桥的网段地址不足时，使用一个新的网段来弥补这个缺陷；
- 隐藏源地址。由于虚拟机在访问外部网络时，地址在 NAT 网关处进行了转换，所以外部网络是无法直接看到要访问的内部虚拟机的 IP 的，在某种程度可以保护内部虚拟机的安全；
- 负载均衡。NAT 有重定向的作用，多个相同应用的虚拟机主备部署后，可以通过 NAT 转换成同一个地址与外部通信，然后再通过负载均衡的软件，进行业务访问的负载均衡；

桥接和 NAT 适用于个人及小规模场景，使用网桥无法查看虚拟网卡的状态，也无法监控经过虚拟网卡的流量，同时，桥接仅支持在 GRE 隧道，功能受限。另外，目前 SDN（Software Define Network，即软件定义网络）非常普及，而网桥却不支持 SDN，因此在大规模场景下，我们会使用虚拟交换机来实现虚拟机间的网络通信，虚拟交换机是升级版的网桥，可以弥补网桥的缺陷。

目前，各个虚拟化厂家都有自己的虚拟交换机产品，如 VMware 的 vSwitch、思科的 Nexus 1000V、华为的 DVS 等，我们以开源的 Open vSwitch 为例来介绍虚拟交换机。

Open vSwitch（以下简写为 OVS）是一款开源的、高质量的、支持多层协议的虚拟交换机，使用开源 Apache2.0 许可协议，由 Nicira Networks 开发，主要实现代码为可移植的 C 代码。其目的是使大规模网络自动化可以通过编程扩展，同时仍然支持标准的管理接口和协议（例如 NetFlow， sFlow， SPAN， RSPAN， CLI， LACP， 802.1ag），Open vSwitch 支持多种 Linux 虚拟化技术，比如 Xen 和 KVM 等等。

以下为 OVS 官网上解释为什么需要 OVS 的理由：

- 状态可迁移。虚拟机在不同主机上时，与其相关联的所有网络状态应当易被识别和迁移，这个状态包括传统的“软状态”（比如虚拟机的二层的相关缓存表）、三层转发状态、路由策略、ACL、QoS 规则以及和监控相关的配置（比如 NetFlow、IPFIX、sFLOW）等。OVS 支持跨实例配置和迁移网络状态。比如，如果虚拟机在迁移时，不仅迁移与其相关联的网络配置（SPAN 规则、ACL、QoS），还会迁移虚拟机此刻的网络状态（包括很难重现的网络状态）。此外，OVS 的状态可以使用开发者的数据模型输出和备份。
- 响应动态网络。高频率的变化是虚拟化的特征之一，众多虚拟机同时发生迁移是常有的事，每次迁移都会改变逻辑网络环境。当网络发生变化时，OVS 的很多特性都支持通过网络管理工具自地的控制和适应。常见的网络管理工具有 NetFlow、IPFIX 和 sFlow，其中最有用的是 OVS 中的一个模块——OVSDB 支持远程触发更新，因此，网络各个方面的信息可以被监控，一旦发生变化，通过编排软件就可以进行实时的响应。这个在当前经常会用到，比如，可以跟踪并响应虚拟机的迁移。OVS 还支持通过 OpenFlow 来远程接入进行流量控制。使用 OpenFlow 后，还有很多其它的功能，比如全局网络发现或者流量的链路状态检测（比如，LLDP、CDP、OSFP 等）。

- 支持逻辑标签维护。分布式虚拟交换机一般通过在网络数据包中附加或操作标记来维护网络中的逻辑上下文。 这可以用于唯一地标识 VM（以抵抗硬件欺骗的方式），或者用于保持仅在逻辑域中相关的一些其它上下文。 构建分布式虚拟交换机的大部分问题是如何高效且正确地管理这些标签。Open vSwitch 包含多种用于指定和维护标记规则的方法，所有这些方法都可供业务流程的远程进程访问。此外，在许多情况下，这些标记规则以优化的形式存储，因此它们不必与重量级网络设备耦合。例如，这允许配置，更改和迁移数千个标记或地址重新映射规则。与此类似，Open vSwitch 支持 GRE 实现，可以处理数千个同时发生的 GRE 隧道，并支持隧道创建，配置和拆除的远程配置。例如，这可以用于连接不同数据中心的私有 VM 网络。

注：我们多次提到 GRE 隧道，它规定了如何使用一种网络协议来封装另外一种网络协议。比如我们在局域网中使用的是 OSPF 网络协议，在广域网中使用的 EGP 网络协议，如果使用了 GRE，我们可以将 OSPF 封装在 EGP 中进行传播。打个通俗的比方，在寄国际信件时，信件是使用自己国家的语言写的，但是信封上的地址需要使用英文来写，自己国家的语言就相当于是 OSPF，英文就相当于是 EGP，而装信件的信封就相当于是 GRE 技术。

- 支持与硬件集成。Open vSwitch 的转发路径（内核数据路径）旨在将数据包处理，“卸载”到硬件芯片组，无论是安装在传统硬件交换机机箱中还是安装在终端主机 NIC 中。这允许 Open vSwitch 控制路径能够控制纯软件实现或硬件开关。目前很多厂商正在努力将 Open vSwitch 移植到硬件芯片组。其中包括多个商用芯片组（Broadcom 和 Marvell），以及许多特定供应商的平台。硬件集成的优势不仅在于虚拟化环境中的性能。如果物理交换机还公开 Open vSwitch 控制抽象，则可以使用相同的自动网络控制机制管理裸机和虚拟主机环境。

总之，在许多方面，Open vSwitch 的目标设计空间与之前的虚拟机管理程序网络堆栈不同，侧重于在大规模基于 Linux 的虚拟化环境中对自动和动态网络控制的需求。

Open vSwitch 的目标是使内核中的代码尽可能小（如性能所需），并在适用时重用现有的子系统（例如 Open vSwitch 使用现有的 QoS 堆栈）。 从 Linux 3.3 开始，Open vSwitch 作为内核的一部分包含在内，用户空间实用程序的打包在大多数流行的发行版中都可用。

OVS 的主要组件包括：

- ovs-vswitchd：OVS 的主要模块，实现 switch 的 daemon，包括一个支持流交换的 Linux 内核模块；
- ovsdb-server：一款轻量级数据库服务器，提供 ovs-vswitchd 获取配置信息；
- ovs-dpctl：用来配置 switch 内核模块；
- scripts and specs：辅助 OVS 安装在 Citrix XenServer 上，作为默认 switch；
- ovs-vsctl：用来查询和更新 ovs-vswitchd 的配置；
- ovs-appctl：用来发送命令消息，运行相关 daemon。

此外，OVS 还提供了一些工具：

- ovs-ofctl：查询和控制 OpenFlow 交换机和控制器；
- ovs-pki：OpenFlow 交换机创建和管理公钥框架。

如下图 3-14 所示，演示了 OVS 的数据包转发流程：

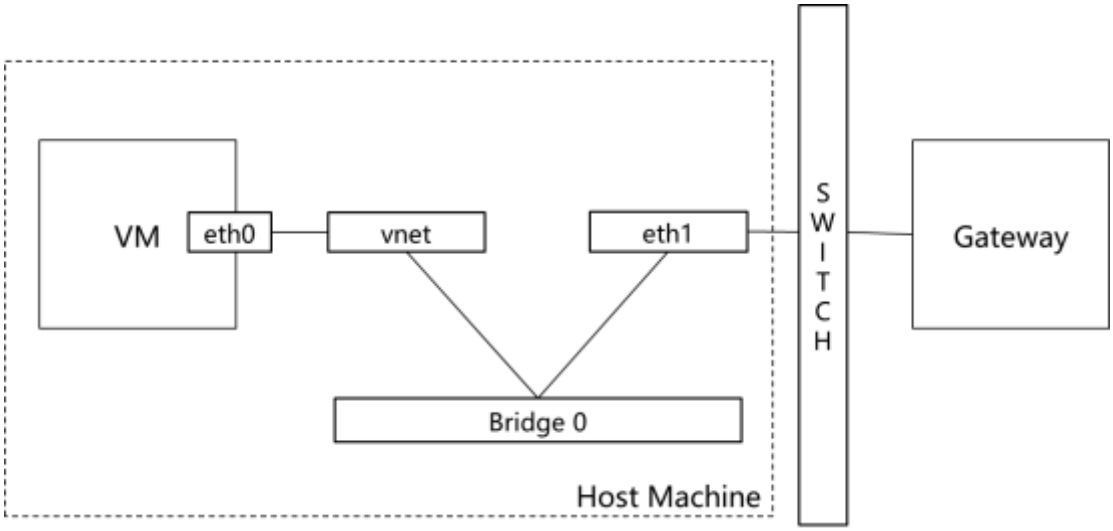


图 3-14 OVS 流量转发示意图

- 虚拟机产生的数据包先发送到虚拟网卡 eth0 中；
- 数据包会首先发送到 tun/tap 设备上，就是图中的 vnet；
- 然后再由 vnet 发送到网桥上；

- 最后经过网桥设备转发到同样在网桥上桥接的物理机网卡 eth1 上，并由 eth1 将该数据包转发到物理的二层交换机上。

虚拟交换机分两种类型，一种是普通虚拟交换机，一种是分布式虚拟交换机。普通虚拟交换机只运行在一台单独的物理主机上，所有与网络相关的配置只适用于此物理服务器上的虚拟机；分布式虚拟交换机分布在不同的物理主机上，通过虚拟化管理工具，可以对分布式虚拟交换机进行统一地配置。虚拟机能够进行热迁移的条件之一就是要有分布式虚拟交换机。

华为虚拟化产品使用的虚拟交换机为分布式虚拟交换机，我们以华为 FusionCompute 为例来介绍分布式虚拟交换机。

3.3.2 华为虚拟化产品的网络特性

3.3.2.1 华为虚拟化产品中的网络方案

华为虚拟交换提供集中的虚拟交换和管理功能。集中管理提供统一的 Portal，进行配置管理，从而简化用户的管理。

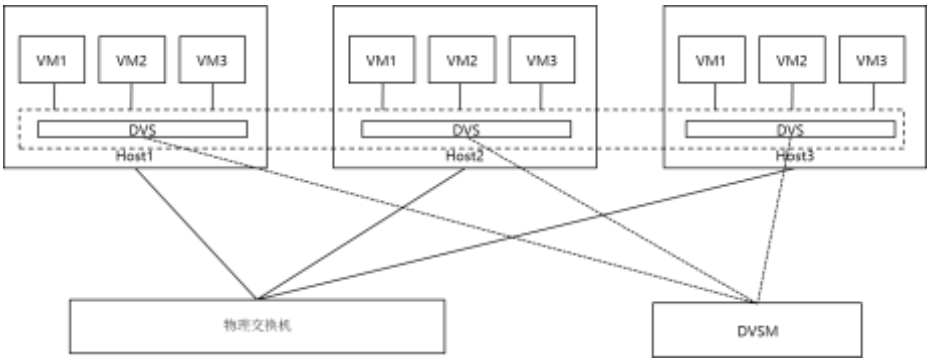


图 3-15 DVS 示意图

通过分布在各物理服务器的虚拟交换机，提供虚拟机的二层通信、隔离、QoS 能力。

分布式交换机模型基本特征：

1） 虚拟化管理员可以配置多个分布式交换机，每个分布式交换机可以覆盖集群中的多个 CNA 节点；

2） 每个分布式交换机具有多个分布式虚拟端口 VSP，每个 VSP 具有各自的属性（速率，统计），为了管理方便采用端口组 Port Group 来管理具有相同属性的一组端口，相同端口组的 VLAN 相同；

3） 虚拟化管理员或业务系统（例如 VDI/IDC）可选择管理/存储/业务使用的不同物理接口；每个分布式交换机可以配置一个 UpLink 端口或者一个 Uplink 端口聚合组，用于 VM 对外的通信。Uplink 端口聚合组可以包含多个物理端口，端口聚合组可以配置负载均衡策略；

4） 每个 VM 可以具有多个 vNIC 接口，vNIC 可以与交换机的 VSP 一一对接；

5） 虚拟化管理员或业务系统可根据业务需求，在一个集群中选择允许进行 2 层迁移的服务器创建虚拟二层网络，设置该网络使用的 VLAN 信息；

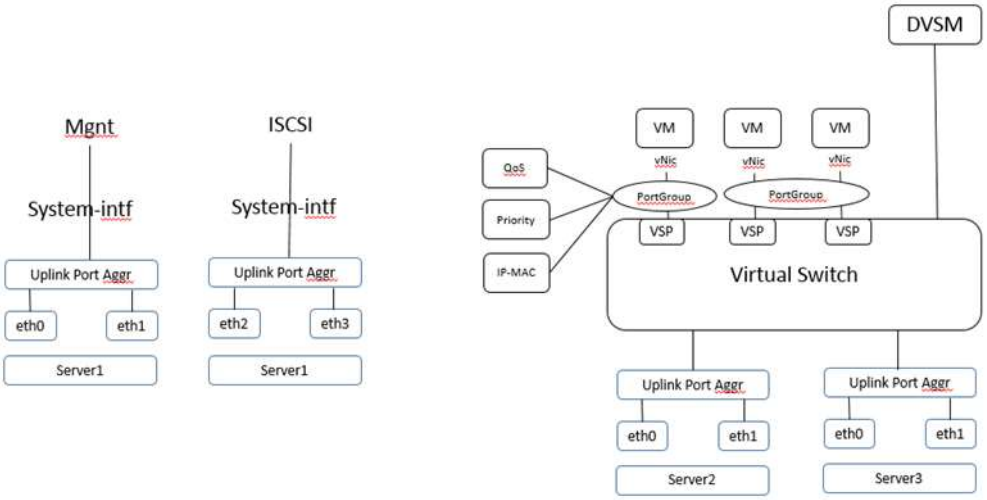


图 3-16 虚拟交换机内部结构

虚拟化管理员可通过定义端口组属性（安全/QoS）来简化对虚拟机端口属性的设置；设置端口组属性，不影响虚拟机正常工作；

端口组：端口组是网络属性相同的一组端口的集合。管理员可以通过配置端口组属性（带宽 QOS、2 层安全属性、VLAN 等）来简化对虚拟机端口属性的设置。设置端口组属性，不影响虚拟机正常工作。

上行链路：分布式交换机关联的服务器物理网口，管理员可以查询上行链路的名称、速率、模式、状态等信息。

上行链路聚合：分布式交换机关联的服务器绑定网口，绑定网口可以包含多个物理网口，这些物理网口可以配置主备或负载均衡策略。

华为的分布式虚拟交换支持基于开源 Open vSwitch 的纯软件的虚拟交换功能。如下图 3-17 所示，为华为分布式虚拟交换机的架构：

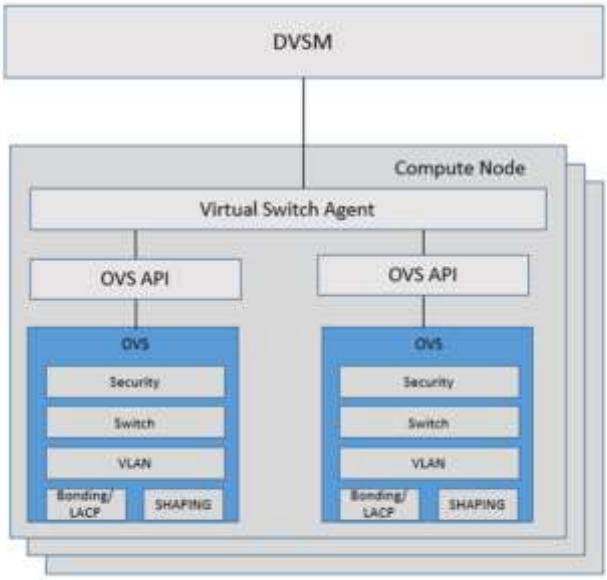


图 3-17 华为 DVS 内部结构

华为分布式虚拟交换机的特点也很明显，具体如下：

1） 集中的管理：统一的 Portal 和集中的管理，简化用户的管理和配置；

2） 开源 Open vSwitch：集成开源 Open vSwitch，充分利用和继承了开源社区虚拟交换的能力；

3） 提供丰富的虚拟交换二层特性，包括交换、QoS、安全隔离等。

3.3.2.2 分布式交换机流量走向描述

- 虚拟机运行在相同主机，但是端口组不同

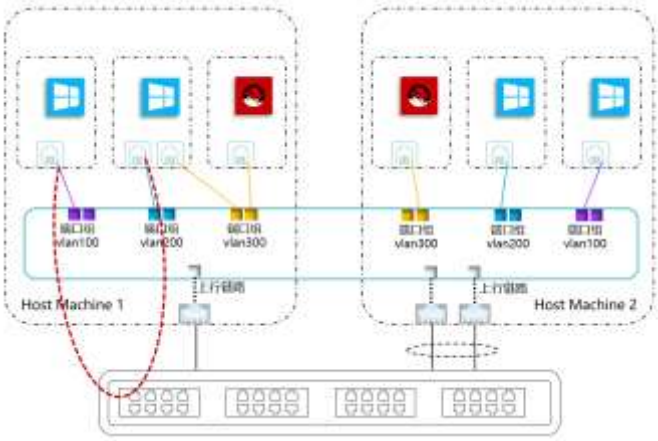


图 3-18 相同主机不同端口组流量走向

前面我们说过，虚拟交换机本质是一台二层交换机，端口组里有一个很重要的参数叫 VLAN ID，如果两个虚拟机不在同一端口组，就代表它们不在同一 VLAN，所以无法通过广播找到对方。一般情况下，属于不同 VLAN 的虚拟机，我们会给它配不同网段的 IP 地址，所以，如果它们之间需要通信就需要使用三层设备，比如三层交换机或路由器，而在华为虚拟化产品 FusionCompute 架构中，三层功能只能由物理的三层设备提供，因此，这两台虚拟机的访问流量需要从主机内部传出到物理的接入交换机，转发到三层设备经过路由后再进入到主机内部，才能完成通信。

- 虚拟机运行在相同主机，且端口组相同

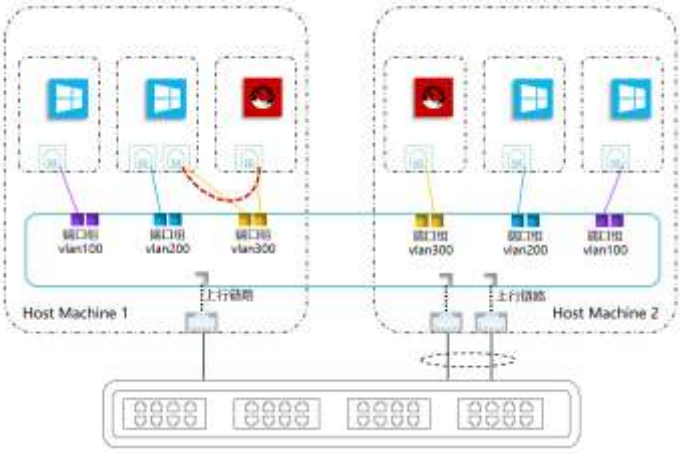


图 3-19 相同主机相同端口组流量走向

属于同一端口组，则虚拟机属于同一广播域，而虚拟交换机支持广播。所以，如果是同主机相同端口组的虚拟机之间互相通信，可以直接通过虚拟交换机完成，通信的流量不需要传出到物理网络中。

- 虚拟机运行在不同主机，但端口组相同

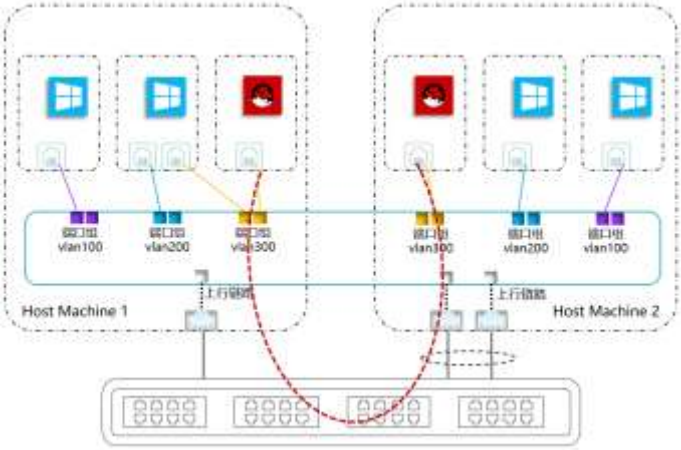


图 3-20 不同主机相同端口组流量走向

虽然虚拟机属于同一端口组，可以通过广播找到彼此，但是不同的物理服务器需要使用物理交换机才能接入到网络中（除非两个物理服务器直接互连，属于特殊情况，暂不考虑）。因此，如果虚拟机使用相同端口组，但是运行在不同物理服务器时，流量需要通过物理服务器的网口传出到物理交换机上，然后才能完成通信。与不同物理服务器不同端口组不一样，此时两台虚拟机之间可以不经过三层设备即可正常通信。

- 同一物理服务器运行不同的 DVS

在实际工作中，经常会根据实际需求在同一物理服务器上运行不同的 DVS，如果有两台虚拟机分别与不同的 DVS 相连，流量该怎么走呢？大多数情况下，如果使用了不同的 DVS，两台 DVS 上的端口组的 VLAN ID 不会相同，这就意味着虚拟机会使用不同的 IP 地址，所以与第一种情况类似，虚拟机之间的通信，数据包会先到三层设备进行路由，然后才能正常通信。

3.3.2.3 安全组

用户根据虚拟机的安全需求创建安全组，每个安全组可以设定一组访问规则。当虚拟机加入安全组后，即受到该访问规则组的保护。用户通过在创建虚拟机时选择要加入的安全组来对虚拟

机进行安全隔离和访问控制。安全组是一个逻辑上的分组，这个分组是由同一个地域内具有相同安全保护需求并相互信任的虚拟机组成。位于同一个安全组的所有虚拟机网卡都将使用该安全组规则进行网络通信。每块虚拟机网卡只能加入一个安全组中。

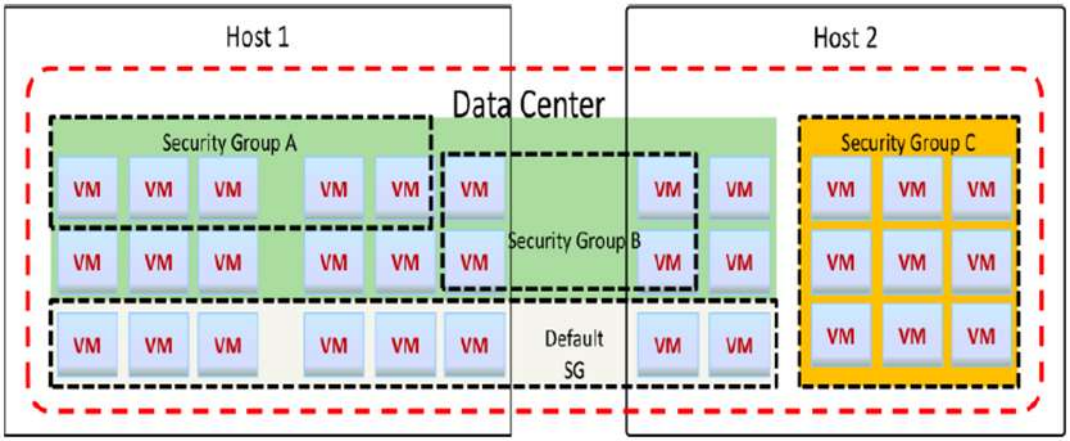


图 3-21 安全组示意图

安全组的作用与防火墙类似，在功能上可以参考防火墙，都是使用 iptables 的包过滤来实现安全控制的。

Netfilter/iptables（简称为 iptables）组成 Linux 平台下的包过滤防火墙。其中，iptables 是一个 Linux 用户空间（userspace）模块，位于/sbin/iptables，用户可以使用它来操作防火墙表中的规则。真正实现防火墙功能的是 netfilter，它是一个 Linux 内核模块，做实际的包过滤。

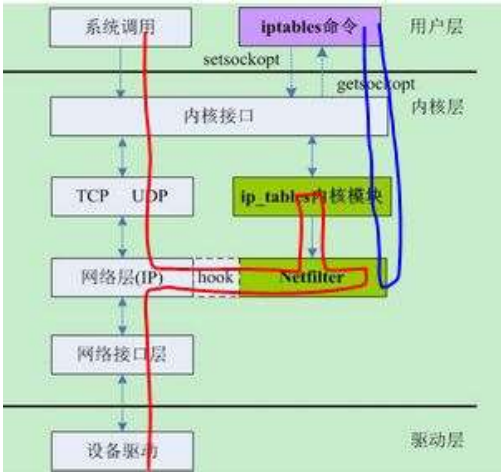


图 3-22 安全组工作原理

上图中的 Netfilter 是一套数据包过滤框架，在处理 IP 数据包时 hook 了 5 个关键钩子，业务可通过这 5 个关键点来挂载自己的处理函数以实现各种功能。

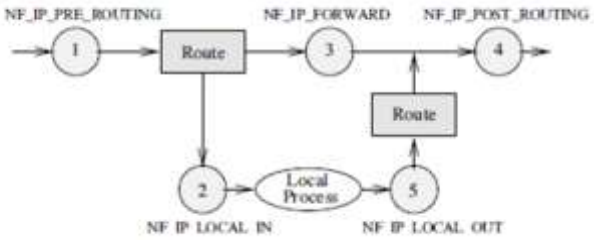


图 3-23 规则过滤示意图

iptables 是通过配置表中的链与规则来进行处理的，进入链处理的网络报文会依序匹配链中的规则，匹配某个规则时会根据规则中指定的操作进行下一步处理。iptables 包含 4 个表，分别是 RAW，Filter（包过滤），NAT（网络地址转换），Mangle（TCP 头修改），这些表根据需要在上述 5 个关键点生成自己的处理链，并将链处理的入口函数挂载在对应的关键点上。

3.3.2.4 虚拟交换模式

华为虚拟交换机提供三种虚拟交换模式：1）普通模式；2）SR-IOV 直通模式；3）用户态交换模式。

- 普通交换

普通模式下，虚拟机有前后端两个虚拟网卡设备，其中前端网卡连接在虚拟交换机的虚拟端口上。虚拟机的网络数据包通过环形缓冲区和事件通道在前后端网卡之间传输，并最终通过后端网卡连接的虚拟交换机实现转发。具体如下图 3-24 所示：

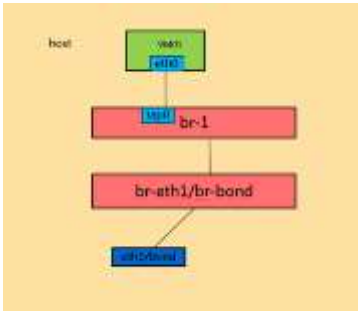


图 3-24 普通交换模式示意图

- SR-IOV 直通

SR-IOV（Single Root I/O Virtualization）技术是 Intel 在 2007 年提出的网络 I/O 虚拟化技术，目前已是 PCI-SIG 的规范。简单来说，支持 SR-IOV 的物理网卡可以虚拟出多个虚拟网卡以供虚拟机使用，对于虚拟机来说就像是使用一块单独的物理网卡一样，相比软件虚拟化提升了网络 I/O 性能，相对于硬件直通（PCI Passthrough）又减少了硬件网卡数量上的需求。如下图 3-25 所示：

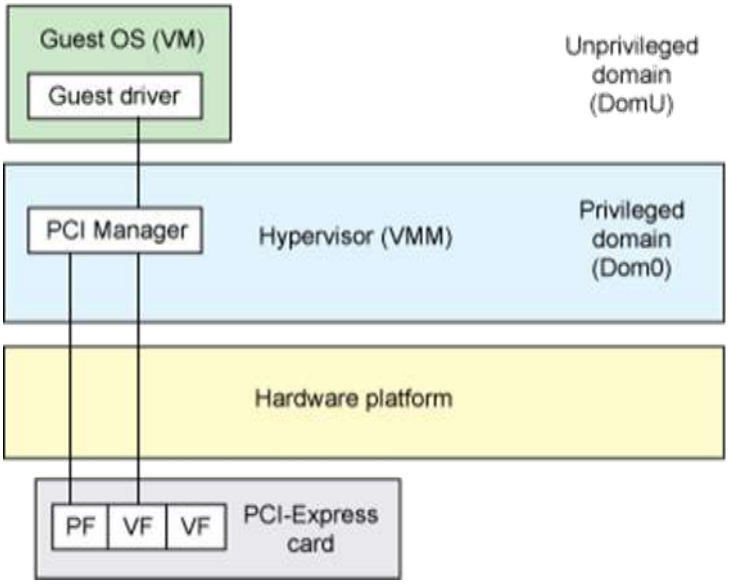


图 3-25 SR-IOV 直通模式示意图

- 用户态交换

通过对虚拟端口加载用户态驱动，在 vswitchd 中启动线程接管内核态收发包功能，从网卡收到的数据包直接在 vswitchd 的线程中接收，接收到数据包后，查询 vswitchd 中的精确流表匹配，然后执行 openflow 的动作和指令，把数据从指定端口发送出去。优势在于使用 DPDK 技术提升了端口 I/O 性能。另外，收发包和基于 openflow 的数据转发都在用户态完成，减少了内核态与用户态间切换带来的开销，带来网络 I/O 性能的提升，相较于 SR-IOV 技术，支持热迁移、热添加网卡等更多高级特性。如下图 3-26 所示：

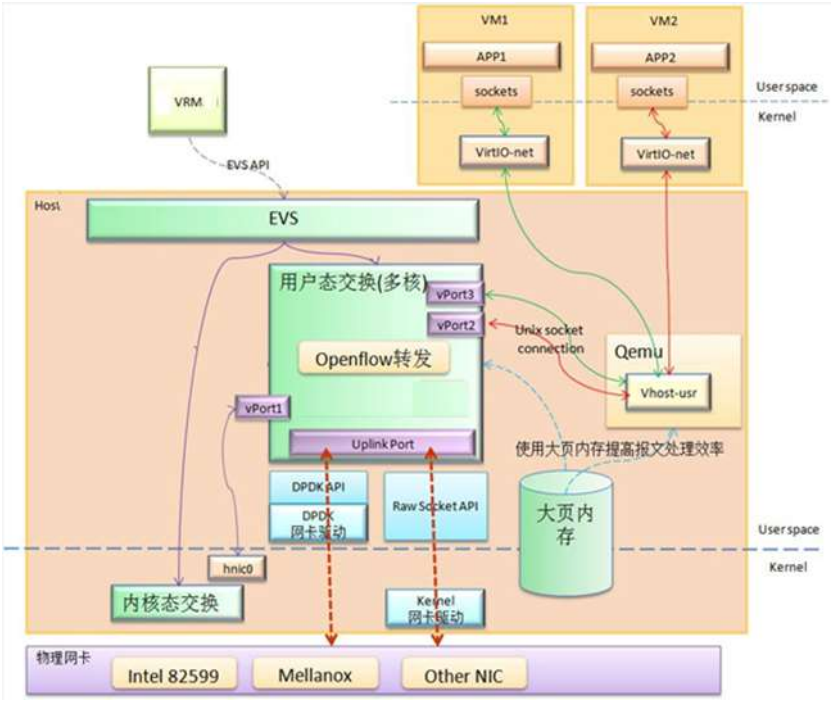


图 3-26 用户态交换模式示意图

完成实验手册中 3.2.4 后，我们验证几个问题：

3 端口组的作用是什么?

4 VRM 是如何收集每个端口产生的流量信息的?

4 云计算中的存储基础知识

日常生活中，我们在买个人电脑时，无论是笔记本电脑还是台式机，硬盘是必不可少的一部分，可能我们只会关注两项参数——容量和类型，容量是 500G 还是 1T，目前最主流的类型是机械硬盘和固态硬盘。在云计算中，硬盘也是必不可少的一部分，但是云计算中的硬盘和普通 PC 不一样，是看不到物理实体的，用户可能只需要关注性能和容量就可以，但是作为一个云计算工程师，除了需要知道如何实现客户的需求，更需要知道如何将物理的硬盘一步步转换为按照用户需求生成的云硬盘。

图 4-1 云硬盘规格

如下图 4-2 所示，为虚拟化中存储的架构。

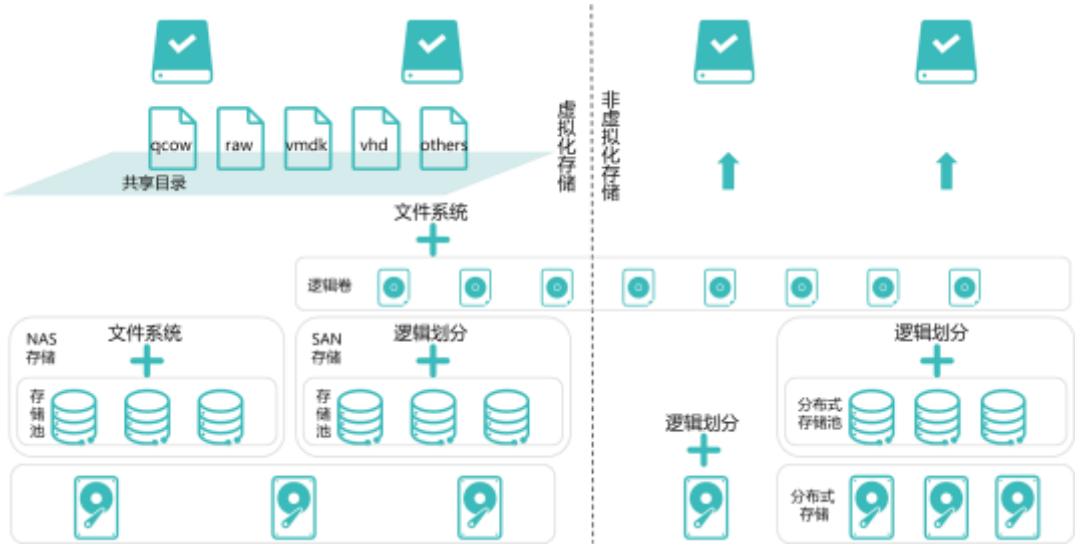


图 4-2 虚拟化中存储架构

在这个架构图中，最下层是物理磁盘，最上层是云硬盘，中间经过了一系列的逻辑划分、文件系统格式化等操作，这些操作都是需要我们重点关注和理解的。另外还需要关注的是虚拟化存储和非虚拟化存储的区别，我们从下往上进行学习，首先需要了解常用的物理磁盘。

4.1 常见的物理磁盘类型

4.1.1 机械硬盘简介

机械硬盘（hard disk drive，HDD）的历史始于 1956 年，世界上第一个磁盘存储系统 IBM 305 RAMAC 由 IBM 公司发明，它拥有 50 个 24 英寸的盘片，重量约 1 吨，容量为 5MB。1973 年，IBM 公司研制成功了一种新型的硬盘 IBM 3340。这种硬盘拥有几个同轴的金属盘片，盘片上涂着磁性材料。它们与可以移动的磁头共同密封在一个盒子里，磁头能从旋转的盘面上读取磁信号的变化。这就是我们今天使用的与硬盘最接近的祖先，IBM 把它叫作温切斯特硬盘。因为 IBM 3340 拥有两个 30MB 的存储单元，而当时有一种很有名的“温切斯特来复枪”口径和装药也恰好包含了两个数字“30”，于是这种硬盘的内部代号就被定为“温切斯特”。1980 年，希捷公司制造出了个人计算机上的第一块“温切斯特”硬盘，这个硬盘与当时的软驱体积相仿，容量为 5MB。如下图 4-3 所示，为温切斯特硬盘：

图 4-3 温切斯特硬盘

硬盘的读取速度在当时受到硬盘转速的限制。提高转速可以加快存取数据的速度，但是硬盘的磁头和盘片是相互接触的，过高的转速会导致磁盘损坏，于是技术人员想到让磁头在盘片上方

“飞行”。盘片高速旋转会产生流动的风，因此只要磁头的形状合适，它就能像飞机一样在磁盘表面飞行，盘片就能很快的旋转而不必担心摩擦会造成故障，这就是温切斯特技术。

温切斯特硬盘采用了创新的技术，磁头被固定在一个能沿盘片径向运动的臂上，磁头并不与盘片接触。当磁头与盘片相对运动时，磁头能感应到盘片表面的磁极，并记录或改变磁极的状态来完成数据的读写。由于磁头相对于盘片高速运动，并且二者距离很近，这时哪怕是一点点灰尘也会造成磁盘的损坏，因此硬盘需要封装在一个密封的盒子里，来保持一个清洁的内部环境，确保磁头和盘片能高效可靠地工作。

在现代的计算机系统中，常见的存储介质有硬盘、光盘、磁带、固态硬盘等，硬盘容量大，价格低廉、读取速度可观、可靠性高，有着其它介质无法代替的作用，仍然被人们认为是重要的存储设备。

我们通常说的硬盘主要是指机械硬盘，它主要由盘片和主轴组件、浮动磁头组件、磁头驱动机构、前驱控制电路和接口等组成，如下图 4-4 所示：

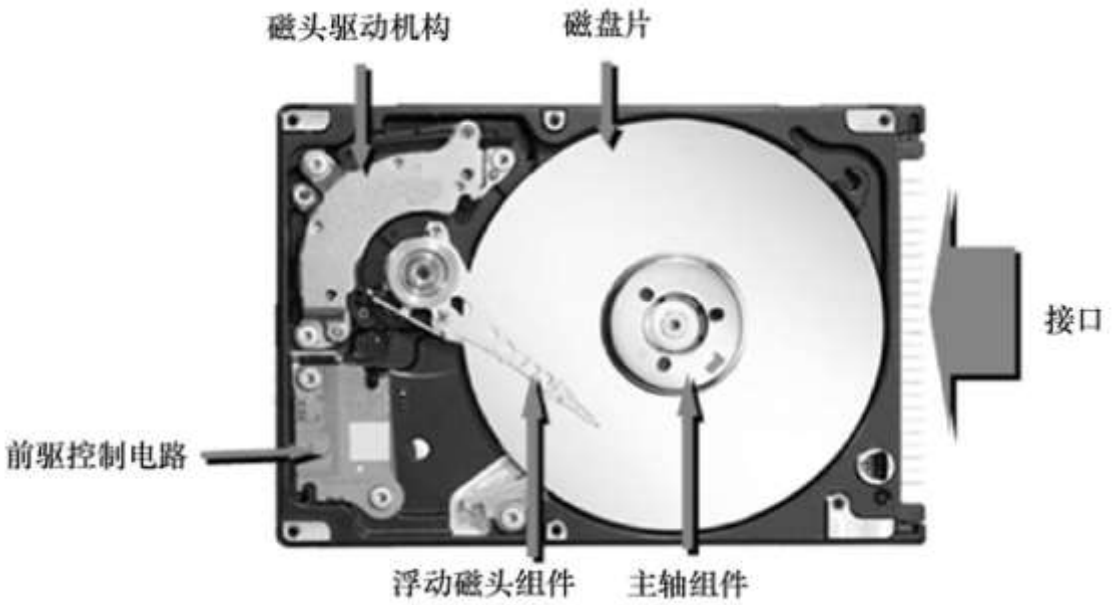


图 4-4 机械硬盘组件

1 盘片和主轴组件。盘片和主轴组件是两个紧密相连的部分，盘片是一个圆形的薄片，上面涂了一层磁性材料用以记录数据。主轴由主轴电机驱动，带动盘片高速旋转。

2 浮动磁头组件。浮动磁头组件由读写磁头、传动手臂和传动轴三部分组成。在盘片高速旋转时，传动手臂以传动轴为圆心带动前端的读写磁头在盘片旋转的垂直方向上移动，磁头感应盘片上的磁信号来读取或改变磁性涂料的磁性，以达到写入信息的目的。

3 磁盘驱动机构。由磁头驱动小车、电机和防震机构组成，其作用是对磁头进行驱动和高精度的定位，使磁头能迅速、准确地在指定的磁道上进行读写工作。

4 前驱控制电路。前驱控制电路是密封在屏蔽腔体内的放大线路，主要作用是控制磁头的感应信号、主轴电机调速、驱动磁头和磁头定位等。

5 接口。通常包含电源接口与数据传输接口。目前主流的接口类型有 SATA 和 SAS，稍后会详细介绍。

硬盘内部用于存储数据的盘片，是一张表面涂有磁性材料的金属圆盘。盘片表面被划分出一圈圈磁道，当盘片在马达的驱动下高速旋转时，设置在盘片表面的磁头便受到精确的控制，沿着磁道读取和写入数据。当系统向硬盘写入数据时，磁头中便产生随着数据内容而变化的电流，这股电流会产生磁场，使盘片表面磁性物质的状态改变，并且这一状态在电流磁场消失后仍能持久地保持下来，这就相当于是将数据保存了下来。当系统从硬盘中读取数据时，磁头经过盘片指定区域，盘片表面的磁场使磁头产生感应电流或线圈阻抗产生变化，这一变化被捕捉下来，经过一定的处理，便能够还原出原本写入的数据。

前面讲过，目前我们最常见的、按照接口不同来区分的机械硬盘类型主要有 SATA 和 SAS，下面我们重点介绍一下。

- SATA 硬盘

说到 SATA，首先要从 ATA（advanced technology attachment）接口说起。ATA 接口实际上就是我们常说的 IDE（integrated drive electronics）接口。ATA 接口从 20 世纪 80 年代一直发展至今，且由于其价格低、兼容性好，曾经是市场上的主流配置。但随着时代的发展，其速度过慢，已不足以应用在现代计算机系统中。

SATA，即串行 ATA（serial ATA），如下图 4-5 所示，现已基本取代所有并行 ATA 接口。SATA，顾名思义，使用串行的方式发送数据。SATA 的显著特点就是比 ATA 快，目前普及的 SATA 3.0 可达 6.0Gbit/s 的传输速率，是并行 ATA 标准的数倍。

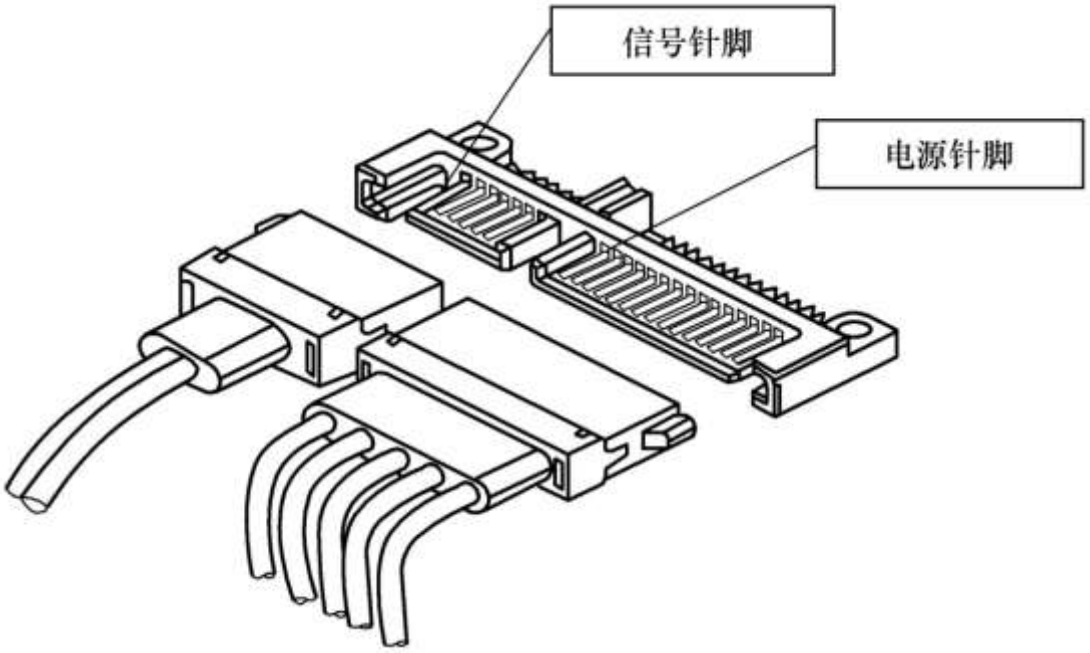


图 4-5 SATA 接口

在传输数据时，SATA 使用独立的数据接口和信号接口。并行 ATA 在传输时使用 16 位的数据总线，并且需要传输许多附加的支持和控制信号。又因为工艺的限制，易受噪音影响，需要使用 5V 电压才能工作。与之相对应，SATA 采用嵌入式时钟信号，具备了更强的纠错能力，且只需要使用 0.5V 的电压即可工作。

从总线结构上看，SATA 使用单通道进行点对点的传输，其中以串行方式按位传输，数据中嵌入了校验和信号位。这种传输方式既能保证速度，又能提高数据传输的可靠性。

SATA 硬盘采用点对点连接方式，支持热插拔，即插即用。SATA 接口通常为 7+15 针，与并行 ATA 相比，SATA 使用较细的线缆，便于弯曲，同时最长可达 1m，极大地改善了机箱内的散热。SATA 接口的硬盘如下图 4-6 所示:

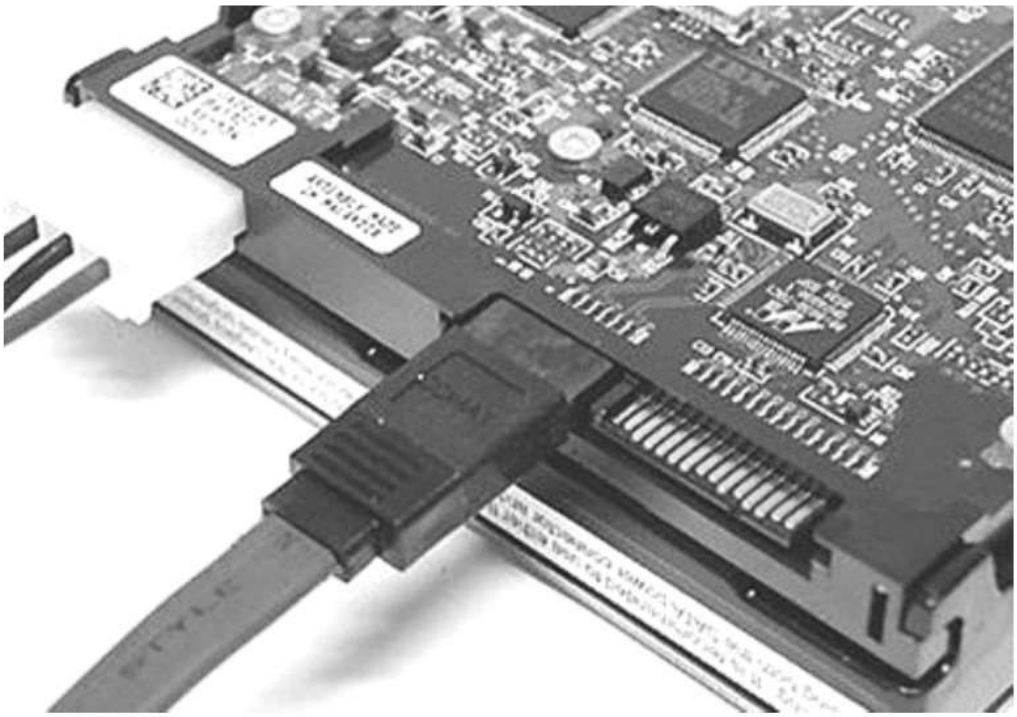


图 4-6 SAS 接口

- SAS 接口硬盘

SAS（serial attached SCSI），即串行连接 SCSI（small computer system interface，小型计算机系统接口）。与 SATA 类似，SAS 也是从对应的并行 SCSI 技术发展而来。

SCSI 以其高性能常常应用于企业级存储领域。SCSI 硬盘分为 50 针、68 针、80 针，历经数十年的发展，当前主流的 SCSI 技术 Ultra 320 SCSI 支持 320MB/s 的传输速度。

SAS 作为 SCSI 技术的分支，与 SATA 类似，通过采用串行传输以得到更高的性能，目前主流的 SAS 传输速率为 6Gbit/s。同时由于采用串行技术可以使用细而长的线缆，不仅可以实现更长的连接距离，还能提高抗干扰能力。SAS 接口正反面如下图 4-7 所示：

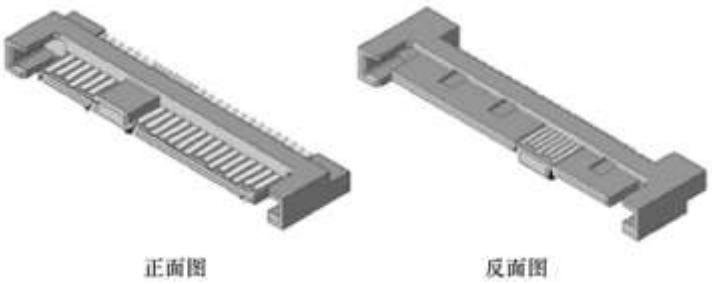


图 4-7 SAS 接口

SAS 向下兼容 SATA，SAS 控制器可以与 SATA 硬盘相连接，这为企业提供了低成本和优秀的灵活性。

在传输方式上，SAS 采用点对点连接方式。与 SATA 类似，SAS 不像并行 SCSI 一样需要终止信号，也不会出现同步问题。SAS 最多可以支持 65536 个设备，不像 SCSI 只能支持 8 个或 16 个设备。

- NL-SAS 接口硬盘

NL-SAS 全称为“near line SAS”，是一种介于 SATA 和 SAS 之间的硬盘。SAS 接口硬盘的读写速度要优于 SATA 硬盘，但是价格也相对要高一些，同时 SAS 硬盘的容量一般会比 SATA 小，所以就产生了 NL-SAS 硬盘，它是用 SAS 接口和 SATA 级盘体组成的硬盘，如下图 4-8 所示，为用 SAS 接口和 SATA 级盘体组成的硬盘。

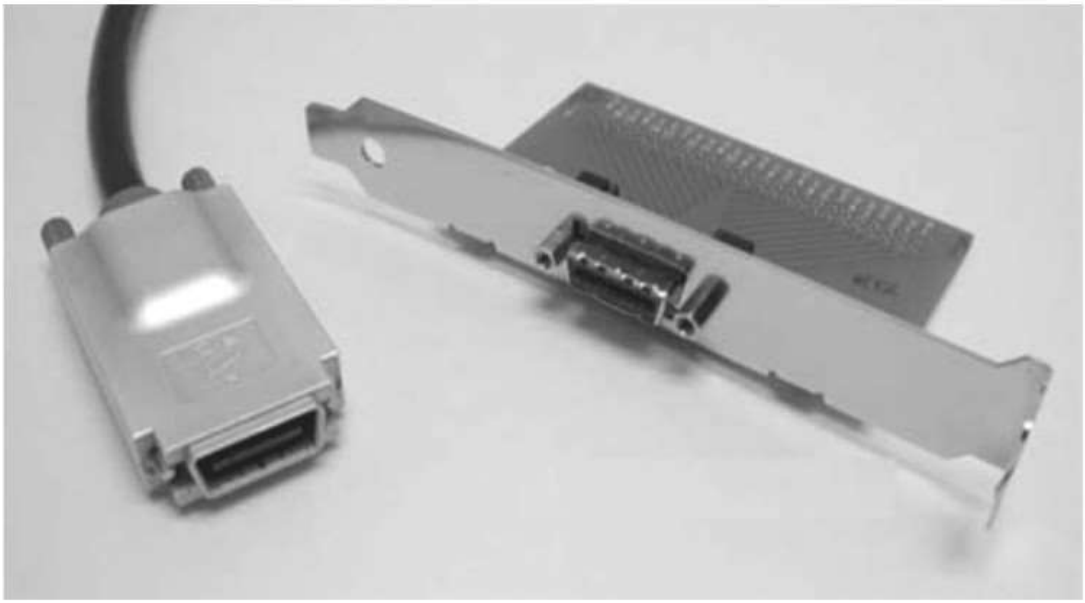


图 4-8 NL-SAS 示意图

NL-SAS 的 IOPS 相比 15000 转速的 SAS 硬盘差了近一半，但是从整个 RAID 的运行情况来看，其性能与 SATA 硬盘相比有明显的提升，而且 NL-SAS 硬盘有 SATA 盘的容量和价格，SAS 硬盘的可靠性，因此 NL-SAS 在市场上很受欢迎。

4.1.2 固态硬盘简介

世界上第一款固态硬盘（Solid State Drive，SSD）出现于 1989 年。当时的固态硬盘价格极为昂贵，但其性能却远低于当时的普通硬盘，因此没有得到广泛应用，但由于固态硬盘独有的抗震、静音、低功耗等特性，却能应用于非常特殊的市场，如医疗工作以及军用市场，因此在这些领域，固态硬盘得到了一定程度的发展。

随着固态硬盘技术的日趋成熟、制造工艺的提升、生产成本的降低，它开始逐渐进入消费领域。2006 年，三星发布了第一款带有 32GB 固态硬盘的笔记本电脑。2007 年初，SanDisk 发布了两款 32GB 的固态硬盘产品。2011 年，泰国发生大洪水，诸多机械硬盘厂商诸如西部数据、希捷等，在泰国的工厂都被迫关闭，导致当年机械硬盘产量大幅下降，价格猛增。这在很大程度上刺激了人们对固态硬盘的需求，从而带来了固态硬盘的黄金时期。如今，固态硬盘在容量、成本、传输速率以及使用寿命上，相比于最初的产品，都有了极大的提升。当前市场上常见的固态硬盘的容量已经达到 128GB～256GB，而每 GB 的价格只有当时的几分之一，让很多消费者都能承担得起。在超薄笔记本与平板领域，固态硬盘更是必不可少的存储设备之一。在未来几年，可以预见固态硬盘仍将受到人们的极大关注。

固态硬盘由主控芯片和存储芯片组成，简单地说，就是用固态电子芯片阵列构成的硬盘。固态硬盘的接口规范、定义、功能及使用方法与普通硬盘完全相同，在产品外形和尺寸上也完全与普通硬盘一致，包括 3.5’、2.5’、1.8’多种类型。由于固态硬盘没有普通硬盘的旋转结构，因而抗震性极佳，同时工作温度范围很大，扩展温度的电子硬盘可工作在−45℃～+85℃，广泛应用于军事、车载、工控、视频监控、网络监控、网络终端、电力、医疗、航空、导航设备等领域。传统机械硬盘都是磁碟型的，数据就储存在磁盘扇区里，而常见的固态硬盘的存储介质是闪存（Flash）。固态硬盘是未来硬盘发展的趋势之一。固态硬盘内部结构如下图 4-9 所示：

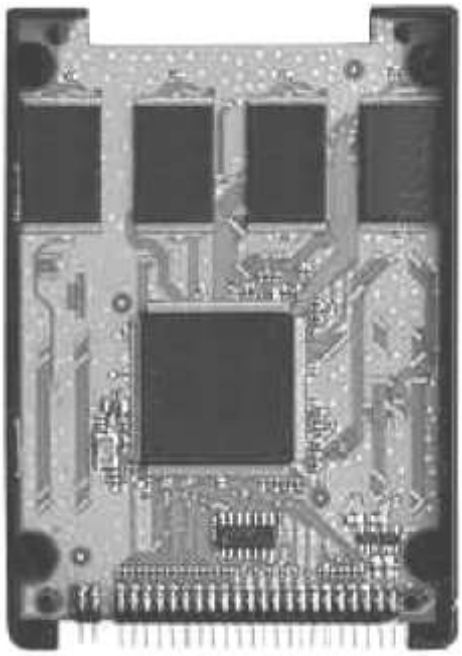


图 4-9 SSD 内部结构图

固态硬盘由主控芯片、存储芯片构成。存储芯片负责存放数据，主控芯片则控制数据的读/写过程协调。存储芯片按介质分为两种，最常见的一种是采用闪存（Flash 芯片）作为存储介质，另一种是采用动态随机存取存储器（DRAM）作为存储介质。

（1）基于闪存的固态硬盘

最为常见的固态硬盘采用闪存芯片作为存储介质。闪存芯片根据使用方式不同，可以被制作成多种电子产品，如固态硬盘、存储卡、U 盘等，这些设备都具有体积小、便携性高等特点。本章节所讨论的固态硬盘，都是基于闪存的固态硬盘。

（2）基于 DRAM 的固态硬盘

这一类固态硬盘采用 DRAM（动态随机存取存储器）作为存储介质。这种存储介质目前广泛应用于内存，性能非常好，而且使用寿命很长。美中不足的是，它只有在供电状态下才能保存数据，一旦失去供电，DRAM 存储的信息就会丢失，因此它需要额外的电源来保护。目前这类固态硬盘价格很高，应用范围较窄。

相比于传统硬盘，固态硬盘在很多方面都更具优势，具体如下：

- 读取速度快

由于固态硬盘是以闪存芯片为介质，没有磁盘与马达的结构，因此在读取数据时节省了寻道时间，在随机读取时尤其能体现速度的优势。同时，固态硬盘的性能不会受到磁盘碎片的影响。

- 抗震性好

固态硬盘内部不存在任何机械活动部件，不会发生机械故障，也不怕碰撞、冲击、振动。这样即使在高速移动，甚至伴随翻转倾斜的情况下，也不会影响正常使用，而且在笔记本电脑意外掉落或与硬物碰撞时，能够将数据丢失的可能性降到最低。

- 无噪音

固态硬盘内部没有机械马达，因此是真正的无噪音静音硬盘。

- 体积小，重量轻

一块很小的电路板上就可以集成一块固态硬盘。

- 工作温度范围更大

典型的硬盘驱动器只能在 5℃～55℃范围内工作。而大多数固态硬盘可在-10℃～70℃温度范围内工作，一些工业级的固态硬盘还可在-40℃～85℃，甚至更大的温度范围下工作。

然而固态硬盘也有两个很大的缺点，导致它目前无法成为机械硬盘的替代品。一个缺点是成本较高。目前固态硬盘每单位容量的价格是传统硬盘的 10 倍左右，大容量固态硬盘在市场上仍然相当少见，因此对于那些对数据读写速度不敏感的应用，机械硬盘仍是第一选择。另一个缺点是固态硬盘的寿命有限，一般高性能的闪存可以擦除 1 万～10 万次，普通消费级的闪存只能擦除 3 千～3 万次。随着制造工艺的不断提升，存储单元的尺寸越做越小，闪存的最大擦除次数还将进一步降低。好在通常情况下，固态硬盘的主控芯片都具有平衡芯片损耗的功能，可以使存储芯片更加均匀地被消耗，从而提高使用寿命。

固态硬盘作为相比于传统硬盘拥有更高读写速度的存储介质，如今已受到人们的广泛关注。由于其原理与传统硬盘不同，没有任何机械的成分，因此固态硬盘在性能上提升很快，同时它还具有抗震、体积小、无噪音、散热小等传统硬盘不具有的优点，因此被很多人寄于希望取代传统硬盘，成为新一代的存储设备。然而，固态硬盘的成本目前还远远高于传统硬盘，加之现在硬盘

的性能已经能满足大部分的需求，因此在很长一段时间内，传统硬盘与固态硬盘还将共存，共同发展。

我们来看看机械硬盘和固态硬盘的对比情况。

表 2 各类硬盘对比

	固态硬盘	SAS	NL-SAS	SATA
性能	非常高	高	较高	较高
可靠性	一般	高	较高	一般
价格	很高	高	较便宜	便宜
能耗	一般	高	较低	较低
推荐场景	适合数据访问非常大的用户使用	适合数据较为离散的高中端用户使用	适合较大数据块、业务I/O压力不大的用户使用	适合大数据块、业务压力不大的用户使用

4.2 集中式存储和分布式存储

4.2.1 集中式存储简介

所谓集中式，指的是所有的资源都集中在某一个中心，然后通过统一接口向外提供服务。集中式存储指的是将所有的物理磁盘集中在硬盘框中，然后通过控制器向外提供存储服务，这里的集中式存储主要指的是磁盘阵列。

集中式存储按照技术架构可以分为 SAN 和 NAS，其中 SAN 又可以细分为 FCSAN、IPSAN、FCoESAN。目前 FCSAN 和 IPSAN 技术都比较成熟，FCoESAN 目前还处于发展初期。

磁盘阵列将多个磁盘组成一个逻辑上更大的磁盘，当做单一的存储资源来使用。在外形上，磁盘阵列包括了控制框和硬盘框两大部分。控制框与硬盘框两者有机结合共同为用户提供一个高可用、高性能和大容量的智能化存储空间。

- SAN 存储简介

存储区域网络（Storage area network，SAN）是一种独立于业务网络系统之外，以块级数据为其基本访问单位的高速存储专用网络。这种网络的主要实现形式有光纤通道存储区域网络（FC-SAN）、IP 存储区域网络（IP-SAN）和 SAS 存储区域网络（SAS-SAN）。不同的实现形式分别采用不同的通信协议和连接方式在服务器和存储设备之间传输数据、命令和状态。

在 SAN 出现以前，使用最多的是 DAS。DAS 已有近 40 年的历史，早期的数据中心使用磁盘阵列以 DAS 的方式扩展存储能力，每一个服务器的存储设备只为单个应用服务，形成了一种孤立的存储环境，然而这些孤立的存储设备难以共享和管理，随着用户数据的不断增长，这种扩展方式在扩展及灾备等方面的弊端也日益明显。而 SAN 的出现解决了这些问题，SAN 将这些存储孤岛用高速网络连接起来，这些存储设备通过网络能被多个服务器共享，实现了数据的异地备份并获得了优异的扩展能力。这些因素都使得这种存储技术快速发展起来。

SAN 作为新兴的存储解决方案，以其加快数据传输速度、提供更大灵活性、减少网络复杂性的优势缓解了传输瓶颈对系统的影响，并大大提高了远端灾难备份的效率。

SAN 是一个由存储设备和各种系统部件构成的网络架构，包括要使用存储设备资源的服务器、用于连接各存储设备的主机总线适配器（host bus adapter，HBA）卡以及 FC 交换机等。

在 SAN 网络中，所有与数据存储相关的通信都在一个与应用网络隔离的独立网络上完成，这也意味着数据在 SAN 中传输时，不会对现有的应用系统数据网络产生任何影响，所以，SAN 可以在不降低原有应用系统数据网络效率的基础上提高网络整体的 I/O 能力，同时增加了对存储系统的冗余链接，并提供了对高可用集群系统的支持。

随着 SAN 存储网络技术的不断发展，至今形成了提到的三类存储区域网络体系：以 FC 为基础的 FC-SAN 光纤通道存储区域网络、以 IP 为基础的 IP-SAN 存储区域网络和以 SAS 总线为基础的 SAS-SAN 网络。这里我们主要了解一下 FC-SAN 和 IP-SAN。

在 FC-SAN 中，存储服务器上通常配置两个网络接口适配器：一个用于连接业务 IP 网络的普通网卡（network interface card，NIC），服务器通过该网卡与客户机交互；另一个网络接口

适配器是与 FC-SAN 连接的主机总线适配器（hoat bus adaptor，HBA），服务器通过该适配器与 FC-SAN 中的存储设备通信。FC-SAN 架构如下图 4-10 所示：

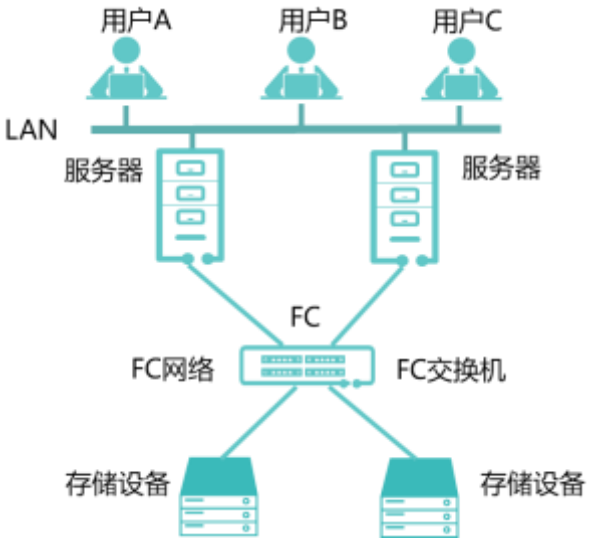


图 4-10 FC-SAN 架构图

IP-SAN 是近年来十分流行的一种网络存储技术。在早期的 SAN 环境中，数据以块为基本访问单位在光纤通道中传播，即早期的 SAN 都是 FC-SAN。由于 FC 协议与 IP 协议不兼容，要实现 FC-SAN，必须单独采购部署 FC-SAN 的设备和组件，其高昂的价格、复杂的配置也让众多中小用户望而却步。因此，FC-SAN 主要应用于对性能、冗余度和可用性等都有较高要求的中高端存储需求。为了提高 SAN 的普及程度和应用范围，并充分利用 SAN 本身所具备的架构优势，SAN 的发展方向开始考虑与已经普及并且相对廉价的 IP 网络进行融合。因此，使用已有 IP 网络构架的 IP-SAN 应运而生，IP-SAN 是标准的 TCP/IP 协议和 SCSI 指令集相结合的产物，是基于 IP 网络来实现块级数据存储的方式。

IP-SAN 与 FC-SAN 的区别在于传输协议和传输介质不同。常见的 IP-SAN 协议有 iSCSI、FCIP、iFCP 等，其中，iSCSI 是发展最快的协议标准，平时我们所说的 IP-SAN 是指基于 iSCSI 协议的 SAN。

基于 iSCSI 的 SAN 的目的就是要使用本地 iSCSI Initiator（启动器，通常为服务器）通过 IP 网络与 iSCSI Target（目标器，通常为存储设备）来建立 SAN 网络连接。IP-SAN 架构如下图 4-11 所示：

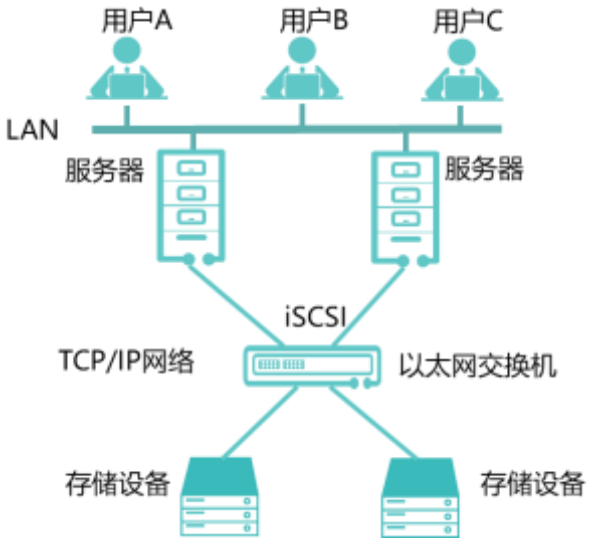


图 4-11 IP-SAN 架构示意图

对比 FC-SAN，IP-SAN 主要有以下几方面的优点：

- 接入标准化。不需要专用的 HBA 卡和光纤交换机，只需要普通的以太网卡和以太网交换机就可以实现存储和服务器的连接。
- 传输距离远。理论上只要是 IP 网络可达的地方，就可以使用 IP-SAN，而 IP 网络是目前地球上应用最为广泛的网络。
- 可维护性好。大部分网络维护人员都有 IP 网络基础，IP-SAN 自然比 FC-SAN 更容易被人接受。另一方面，IP 网络维护工具已经非常发达，IP-SAN 充分发扬了拿来主义。
- 后续带宽扩展方便。因为 iSCSI 是承载于以太网的，随着 10GB 以太网的迅速发展，IP-SAN 单端口带宽扩展到 10GB 已经是发展的必然。

这些优势使得产品的总体拥有成本（TCO）降低，比如建设一个存储系统，总体拥有成本包括需要购买磁盘阵列、接入设备（HBA 和交换机）、人员培训、日常维护、后续扩容、容灾扩展

等。IP-SAN 因为 IP 网络的广泛应用优势，可以大幅降低单次采购的接入设备采购成本、减少维护成本，而且后续扩容和网络扩展成本也大幅降低。

IP-SAN 和 FC-SAN 其它方面的对比如下表：

表 3 IP-SAN 和 FC-SAN 对比

描述	IP SAN	FC SAN
网络速度	1Gb、10Gb、40Gb	4Gb、8Gb、16Gb
网络架构	使用现有IP网络	单独建设光纤网络和HBA卡
传输距离	理论上没有距离限制	受到光纤传输距离的限制
管理、维护	与IP设备一样操作简单	技术和管理较复杂
兼容性	与所有IP网络设备都兼容	兼容性差
性能	目前主流1Gb，10Gb正在发展	非常高的传输和读写性能
成本	购买与维护成本都较低	购买（光纤交换机、HBA卡、光纤磁盘阵列等）与维护（培训人员、系统设置与监测等）成本高
容灾	本身可以实现本地和异地容灾，且成本低	容灾的硬件、软件成本高
安全性	较低	较高

- NAS 简介

网络附加存储（network attached aToRage，NAS）是一种将分布、独立的数据整合为大型、集中化管理的数据中心，以便于不同主机和应用服务器进行访问的技术。通常 NAS 被定义为一种特殊的专用文件存储服务器，包括存储设备（如磁盘阵列、CD/DVD 驱动器、磁带驱动器或可移动的存储介质）和内嵌系统软件，可提供跨平台文件共享功能。

NAS 的出现与网络的发展密不可分，Internet 的雏形 ARPANET 出现后，现代网络技术得到了迅猛的发展，人们在网络中共享数据的需求越来越多。但是在网络中共享文件面临着跨平台访问和数据安全等诸多问题。早期的网络共享如下图 4-12 所示：

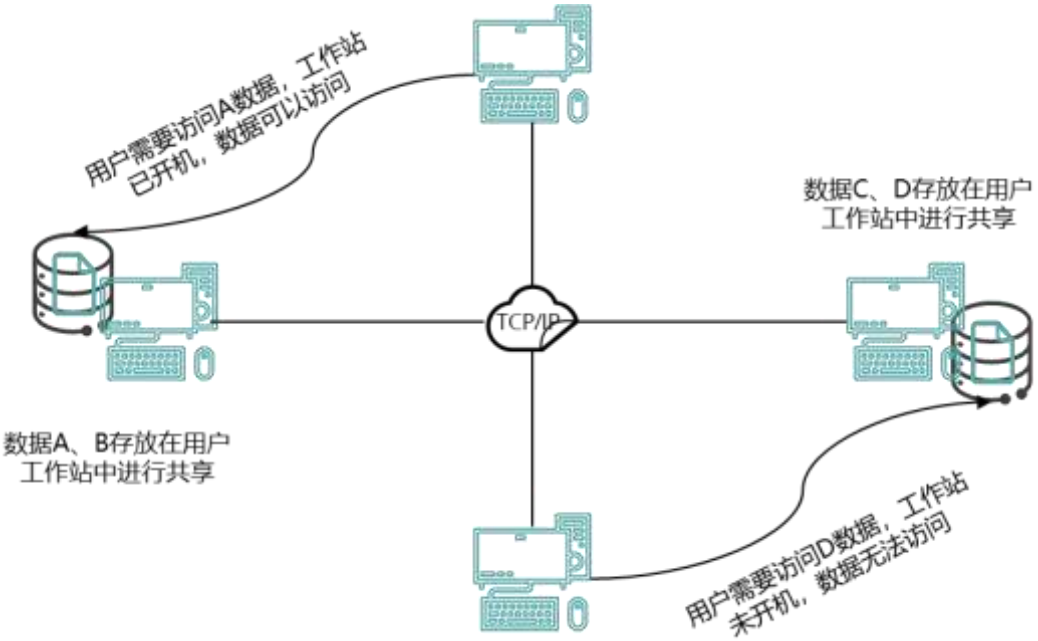


图 4-12 早期网络共享

为了解决这个问题，可以设置一台专门的计算机来保存大量的共享文件，这台计算机连接到现有的网络上，并允许整个网络上的所有用户共享其存储空间。正是通过这种办法，早期的 UNIX 网络环境演化为依赖“文件服务器”共享数据的方式。

使用专门的服务器来提供共享数据存储，拥有大量的存储磁盘空间，保证数据的安全可靠是必须的。同时，单台服务器承担着众多服务器的访问需求，需要对文件共享服务器进行文件 I/O 方面的优化。除此之外，操作系统的额外开销是不必要的。因此，在这种方式下使用的计算机应当配有只具备 I/O 功能的“瘦”操作系统连接到现有的网络中，除此以外的功能，都不是这类服务器必需的。网络中的用户能够像访问自己工作站上的文件一样访问这台特殊服务器上的文件，从根本上实现整个网络中所有用户对文件共享的需求。早期 UNIX 环境下的 TCP/IP 网络示意图如下图 4-13 所示：

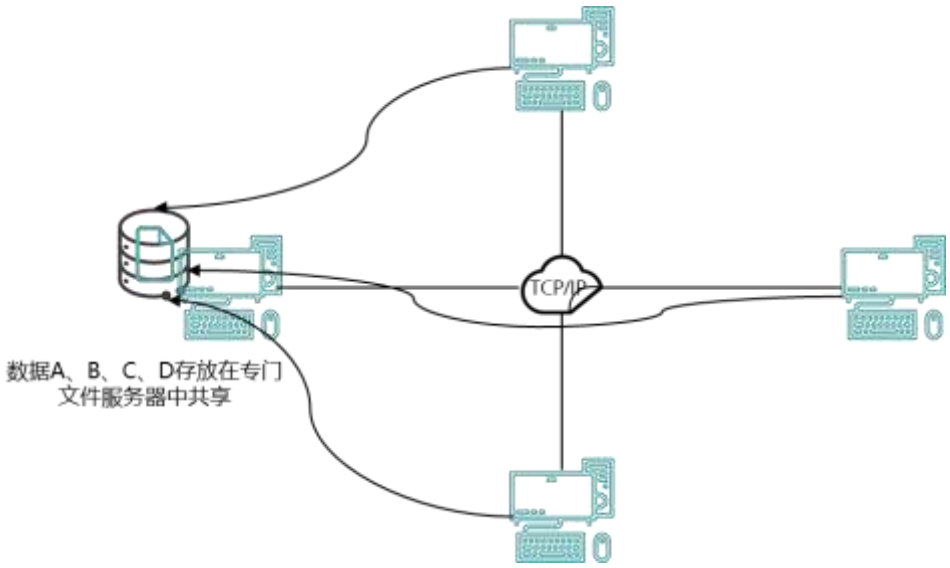


图 4-13 TCP/IP 网络共享示意图

随着网络的发展，网络中不同计算机间的数据共享需求越来越多。大多数情况下，人们希望网络中的系统和用户可以连接到特定的文件系统并访问数据，从而可以像处理本地操作系统中的本地文件那样来处理来自共享计算机的远程文件，进而可以为用户提供一个虚拟的文件集合，这个集合中的文件并不存在于本地计算机的存储设备中，其位置实际上是虚拟的。这种存储方式的发展方向之一就是与支持 Windows 操作系统的传统客户机/服务器环境相集成。这涉及诸如 Windows 的网络能力、专用协议以及基于 UNIX 的数据库服务器等问题。在最初的发展阶段中，Windows 网络由一种至今仍在使用的网络文件服务器组成，并且使用一种专用的网络系统协议。早期的 Windows 文件服务器示意图如下图 4-14 所示：

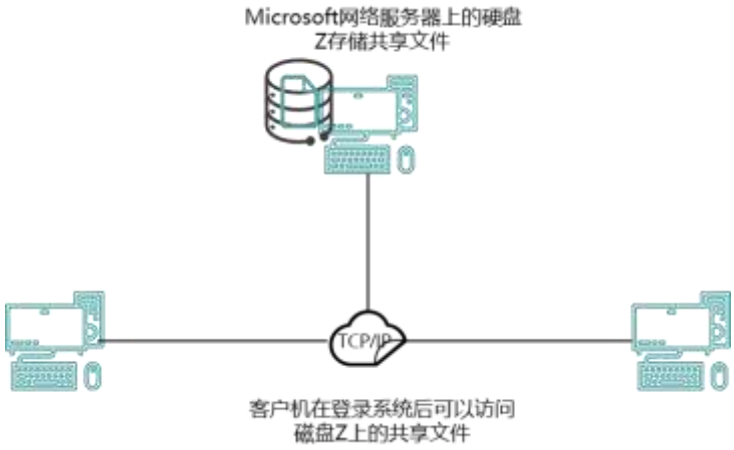


图 4-14 Windows 文件服务器示意图

文件共享服务器的出现使数据存储趋于向集中式存储发展，这种趋势使得集中的数据和业务量也飞速增长。因此，专注于文件共享服务的 NAS 产品应运而生。

NAS 通常在一个 LAN 上拥有自己的节点，无需应用服务器的干涉，NAS 允许用户通过网络直接存取文件数据，在这种配置中，NAS 将集中管理和处理网络上的所有共享文件，将负载从应用或企业服务器上释放出来，有效降低总体拥有成本（total cost of ownership，TCO），保护用户的投资。简单来说，NAS 设备就是连接在网络上，具备文件存储功能的设备，因此也称为“网络文件存储设备”。它是一种专用文件数据存储服务器，以文件为核心，实现了集中文件存储与管理，将存储设备与服务器彻底分离，从而释放带宽，提高性能，保护用户的投资，并降低 TCO。

从本质上讲，NAS 是存储设备而不是服务器。NAS 不是精简版的文件服务器，它具有某些服务器没有的功能特性。服务器的作用是处理业务，存储设备的作用是存储数据，在一个完整的应用环境中应将两种设备有机地结合起来使用。

NAS 的内在价值在于其拥有利用数据中心现有的资源，以快速且低成本的方式提供文件存储服务的能力。现在的解决方案可以在 UNIX、Linux 以及 Windows 环境之间实现兼容，并且能够轻易提供与用户的 TCP/IP 网络相连接的能力。NAS 系统示意图如下图 4-15 所示:

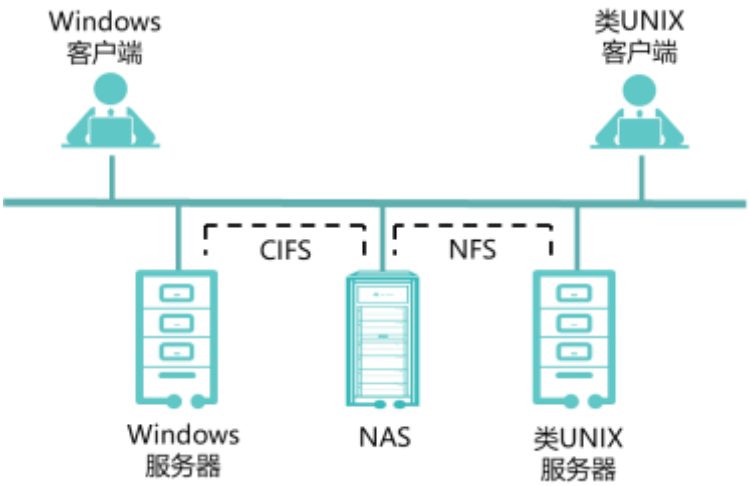


图 4-15 NAS 示意图

NAS 要求能够实现大量数据的存储与备份，在此基础上还需要提供稳定而高效的数据传输服务，这样的要求仅仅依靠硬件是无法完成的，NAS 还需要一定的软件来实现这样的要求。NAS 的软件按照功能可以划分为操作系统、卷管理器、文件系统、网络文件共享和 Web 管理 5 个模块，具体如下图 4-16 所示：

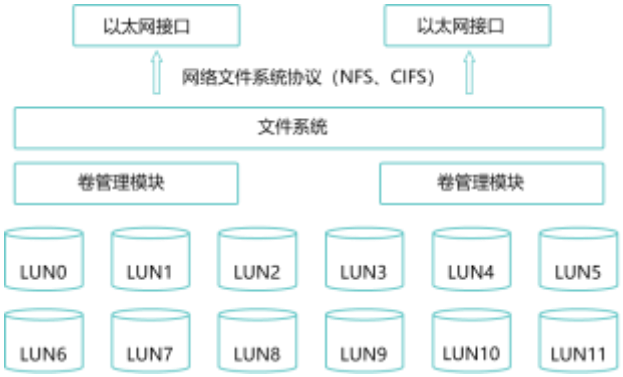


图 4-16 NAS 架构

NAS 设备支持对公用互联网文件系统（common internet file system，CIFS）或网络文件系统（network file system，NFS）进行读写，也支持同时对两者进行读写。

CIFS（Common Internet File System）是由微软的 SMB（Server Message Block）发展而来的一个公共、开放的文件系统。SMB 是微软基于 NetBIOS 设定的一套文件共享协议。通过 CIFS，用户可以访问远程计算机上的数据。此外，CIFS 提供了一定的机制来避免读写冲突与写写冲突，从而支持多用户访问。

为了让 Windows 和 UNIX 计算机达成资源共享，让 Windows 客户不需要更改设置，就能像使用 Windows NT 服务器一样使用 UNIX 计算机上的资源，最好的办法是在 UNIX 中安装支持 SMB/CIFS 协议的软件。当所有主流的操作系统都支持 CIFS 之后，计算机之间的交流就方便了。Samba 帮助 Windows 和 UNIX 用户实现了这一愿望。人们建立基于 CIFS 的共享服务器，将资源共享给它的目标计算机，目标计算机在自己的系统中通过简单的共享映射，将 CIFS 服务器上的共享资源挂载到了自己的系统中，把它当成自己本地文件系统资源一样来使用。通过一个简单的映射，计算机客户就从 CIFS 服务器上得到了它想要的一切共享资源。

NFS（Network File System）是由 Sun 公司开发的，NFS 使用户能够共享文件，它的设计是为了在不同的系统之间使用，所以其通信协议设计与主机及作业系统无关。当用户想用远程文件时，只需要使用挂载命令，就可把远程的文件系统挂载在自己的文件系统之下，使用远程文件和使用本机的文件没有什么区别。

NFS 与平台无关的文件共享机制是基于 XDR/RPC 协议实现的。

外部数据表示（external data representation，EDR）可以转换数据格式。通常，EDR 将数据转换到一种统一的标准数据格式，从而保证在不同的平台、操作系统与程序设计语言中，数据表示的一致性。

远程过程调用（remote procedure call，RPC）请求远程计算机给予服务。用户通过网络将请求传送到远程计算机，由远程计算机完成请求的处理。

NFS 利用虚拟文件系统（virtual file system，VFS）机制，将用户对远程数据的访问请求，通过统一的文件访问协议和远程过程调用，发送给服务器处理。NFS 不断发展，从 1985 年出现至今，已经经历了 4 个版本的更新，被移植到了几乎所有主流的操作系统中，成为分布式文件系统事实上的标准。NFS 出现在一个网络状态不太稳定的时代，起初是基于 UDP 传输的，而并未采用可靠性较高的 TCP。虽然 UDP 在可靠性较好的局域网中工作良好，但在可靠性较差的广域网如互联网上运行时，则不能胜任。当前，随着 TCP 的改进，运行于 TCP 上的 NFS 可靠性高、性能良好。

CIFS 和 NFS 对比如下表：

表 4 CIFS 和 NFS 对比

	CIFS	**NFS**
传输特点	基于网络，可靠性要求高	独立传输
易用性	无需额外软件	需要安装专用软件
安全性	无法进行错误恢复	可以进行错误恢复

文件转换	不保留文件格式特性	保留文件格式特性

4.2.2 RAID 机制

集中式存储把所有的磁盘集中到硬盘框，使用控制框进行统一控制，可以实现存储容量的动态扩展，同时增强系统的容错能力，提高存储系统的读写性能，而用到的技术就是 RAID 机制。

RIAD 全称是 Redundant Arrays of Independent Disks，冗余磁盘阵列。RAID 技术经过不断的发展，现在已拥有了 RAID-0～RAID-6 共 7 种基本的 RAID 级别。另外，还有一些基本 RAID 级别的组合形式，如 RAID-10（RAID-1 与 RAID-0 的组合）、RAID-50（RAID-5 与 RAID-0 的组合）等。不同 RAID 级别代表不同的存储性能、数据安全性和存储成本。这里我们只介绍 RAID0、RAID1、RAID5 和 RAID6。

- RAID0

RAID-0 也称为条带化（stripe），其原理是将多个物理磁盘合并成一个大的逻辑磁盘，它代表了所有 RAID 级别中最高的存储性能，不具有冗余，不能并行 I/O，但速度最快。在存放数据时，根据构建 RAID-0 的磁盘个数对数据进行分段，然后同时将这些数据并行写进磁盘中，因此在所有级别中，RAID-0 的速度是最快的。但是 RAID-0 没有冗余功能，如果一个物理磁盘损坏，则所有的数据都会丢失。

从理论上讲，磁盘个数和总磁盘性能应该成倍数关系，总磁盘性能等于“单一磁盘性能×磁盘数”。但实际上受限于总线 I/O 瓶颈及其它因素的影响，RAID 性能随磁盘个数的增加不再是倍数关系，也就是说，假设一个磁盘的性能是 50MB/s，两个磁盘的 RAID-0 性能约为 96MB/s，3 个磁盘的 RAID-0 也许是 130MB/s 而不是 150MB/s，所以两个磁盘的 RAID-0 最能明显感受到性能的提升。

如下图 4-17 所示，为 RAID0 的示意图，图中有 Disk 1 和 Disk 2 两个磁盘，RAID-0 的做法是将要储存的内容（D1，D2……）根据磁盘数目分成两部分同时储存。D1 和 D2 分别储存到 Disk 1 和 Disk 2 中，等 D1 储存完成后，开始将 D3 储存进 Disk 1 中，其余数据块同理。这样

可以将两个磁盘看成一个大磁盘，并且两侧同时进行 I/O。不过如果某块数据坏掉，整个数据就会丢失。

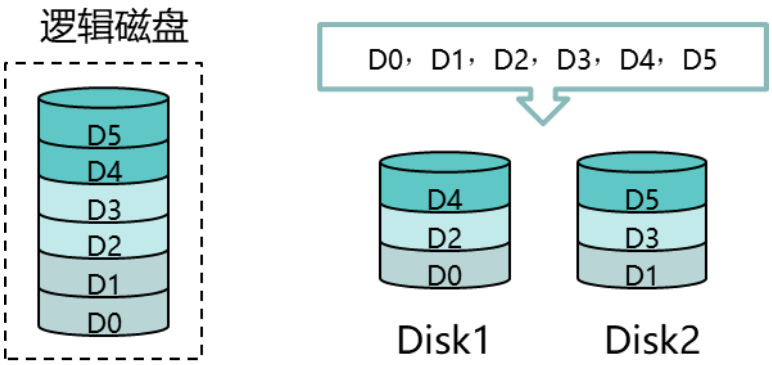


图 4-17 RAID-0 示意图

RAID-0 的读写性能较好，但是没有数据冗余，因此 RAID-0 适用于对数据访问具有容错能力的应用，以及能够通过其它途径重新形成数据的应用，如 Web 应用以及流媒体。

- RAID1

RAID-1 又称为 Mirror 或 Mirroring（镜像），其目的是最大限度地保证用户数据的可用性和可修复性。RAID-1 的原理是把用户写入硬盘的数据百分之百地自动复制到另外一个硬盘上。

RAID-1 在主硬盘上存放数据的同时，也在镜像硬盘上写同样的数据。当主硬盘损坏时，镜像硬盘代替主硬盘的工作。因为有镜像硬盘做数据备份，所以 RAID-1 的数据安全性是所有 RAID 级别中最好的。但是无论用多少磁盘做 RAID-1，有效数据空间大小仅为单个磁盘容量，是所有 RAID 级别中磁盘利用率最低的一个。

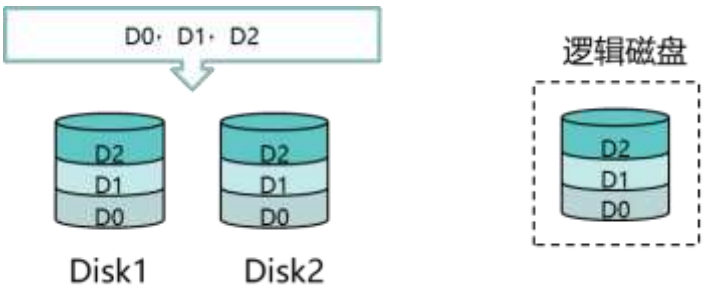


图 4-18 RAID-1

如上图 4-18 所示，为 RAID-1 的示意图，图中有 Disk 1 和 Disk 2 两个磁盘，在储存数据时，将要储存的内容（D1，D2……）储存进主磁盘 Disk 1，同时在 Disk 2 中再次将数据储存一遍，以达到数据备份的目的。

RAID-1 是所有 RAID 级别中单位存储成本最高的，但因其提供了几乎最高的数据安全性和可用性，所以 RAID-1 适用于读操作密集的 OLTP 以及其它要求数据具有较高读写性能和可靠性的应用，如电子邮件、操作系统、应用程序文件和随机存取环境等。

- RAID5

RAID-5 是高级 RAID 系统中最常见的一种 RAID 级别，由于其出色的性能与数据冗余平衡设计而被广泛采用。其全名为“独立的数据磁盘与分布式校验块”。RIAD 使用的是奇偶校验来进行校验和纠错。

RAID5 数据存储方式如下图 4-19 所示，图中以三个硬盘为例，P 为数据的校验值，D 为真实的数据。RAID 5 不对存储的数据进行备份，而是把数据和相对应的奇偶校验信息存储到组成 RAID5 的各个磁盘上，并且数据和相对应的奇偶校验信息分别存储于不同的磁盘上。当 RAID5 的一个磁盘数据发生损坏后，利用剩下的数据和相应的奇偶校验信息可以恢复被损坏的数据。因此，RAID-5 是一种存储性能、数据安全和存储成本兼顾的存储解决方案。

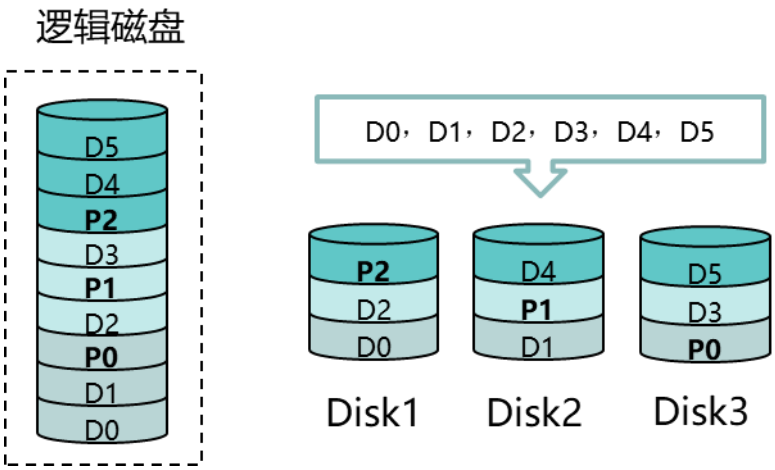


图 4-19 RAID5 示意图

RAID-5 尽管有一些容量上的损失，但是能够提供最佳的整体性能，因而也是被广泛应用的一种数据保护方案。它适合于 I/O 密集、高读/写比率的应用，如联机事务处理等。

- RAID6

RAID-6 是为了进一步加强数据保护而设计的一种 RAID 方式，与 RAID-5 相比，RAID-6 增加了第二种独立的奇偶校验信息块。这样一来，等于每个数据块都有两个校验保护屏障（一个分层校验，一个是总体校验），因此 RAID-6 的数据冗余性能非常好。但是，由于增加了一个校验，所以写入的效率较 RAID-5 差，而且控制系统的设计也更为复杂，第二块的校验区也减少了有效存储空间。

常见的 RAID-6 技术有 P+Q 和 DP 两种，这两种技术获取校验信息的方法不同，但是都可以允许整个阵列中两块磁盘数据丢失。

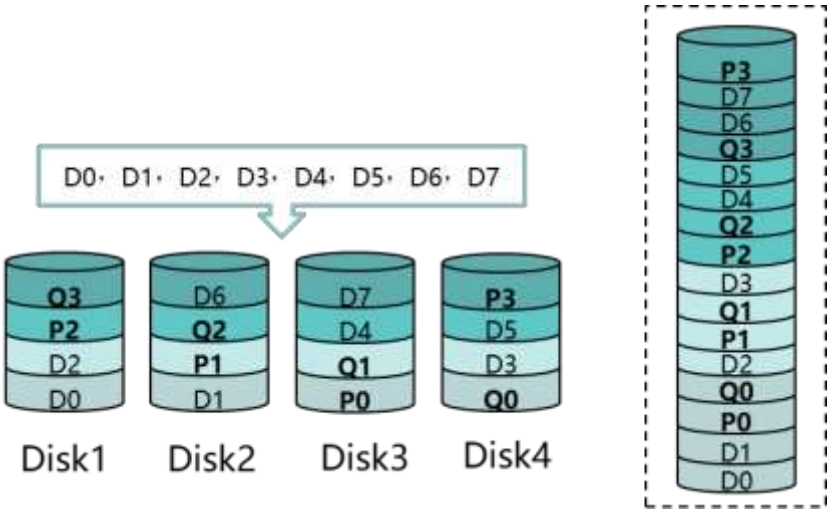


图 4-20 RAID6 示意图

RAID-6 的数据安全性比 RAID-5 高，即使阵列中有两个磁盘故障，阵列依然能够继续工作并恢复故障磁盘的数据。但是控制器的设计较为复杂，写入速度不是很高，而且计算校验信息和验证数据正确性所花的时间也比较多，当对每个数据块进行写操作时，都要进行两次独立的校验计算，系统负载较重，而且磁盘利用率相对 RAID-5 要低一些，配置也更为复杂，适用于对数据准确性和完整性要求更高的环境。

注：4.1 和 4.2 小节来自于华为认证系列丛书《信息存储与 IT 管理》。

4.2.3 分布式存储和副本机制

与目前常见的集中式存储技术不同，分布式存储技术并不是将数据存储在某个或多个特定的节点上，而是通过网络将企业每台机器上的磁盘空间集中起来，并将这些分散的存储资源构成一个虚拟的存储设备，数据分散地存储在企业的各个角落。

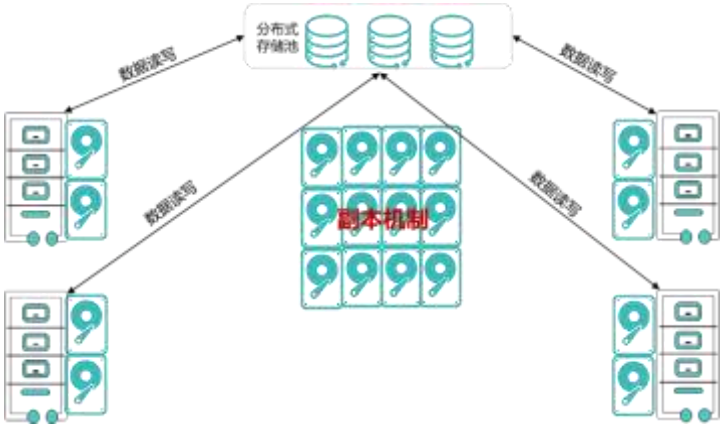


图 4-21 分布式存储

如上图 4-21 所示，分布式存储使用的设备一般是普通服务器，而非存储设备。分布式存储没有控制框和硬盘框，所有的硬盘存储资源都来自于通用的 x86 架构服务器，对硬盘的识别和管理都需要分布式存储本身提供的客户端，这些客户端负责数据路由的建立、I/O 读写的执行等等。

采用分布式存储客户端的方式，既有优势，也有一定的劣势。首先在扩容方面，只要给 x86 架构服务器安装了客户端，它就可以成为分布式存储的一份子，所以这种方式有极大的可扩展性；但是，除了服务器本身所承载的应用外，安装在服务器上的客户端软件也需要一定的计算资源，因此在规划分布式存储时，需要在提供存储资源的服务器上预留一定的计算资源，所以这种方式对服务器的硬件资源有一定的要求；最后，在传统的集中式存储中，I/O 读写都是通过控制器来完成，然而控制器的数量有限，在分布式存储中，安装了客户端的服务器可以进行 I/O 读写，从而可以突破传统集中式存储控制器数量的限制，在一定程度上也能提高 I/O 读写速度，但是每次读写时都需要计算数据读写的路径，如果客户端太多的话，计算路径就会比较复杂，因此客户端并不是越多越好，达到性能最优时，增加客户端的数量是无法继续提升性能的。

为了保证数据的高可用性和安全性，集中式存储使用的是 RAID 技术，RAID 可以通过硬件和软件的方式来实现，无论是软 RAID 或者硬 RAID，所有的硬盘都需要在一个服务器内（硬 RAID 需要统一 RAID 卡，软 RAID 需要统一操作系统），由于分布式存储的硬盘分布在不同的服务器上，所以无法再使用 RAID 机制。因此，分布式存储是通过副本机制来实现数据的高可靠性的。副本机制是指将数据复制成多份一模一样的内容，并分别保存在不同的服务器上，当某台服务器出现故障后，数据并不会丢失。分布式存储系统把所有服务器的本地硬盘组成若干个资源池，基于资源池提供创建/删除应用卷（Volume）、创建/删除快照等接口，为上层软件提供卷设备功能，具体如下图 4-22 所示：

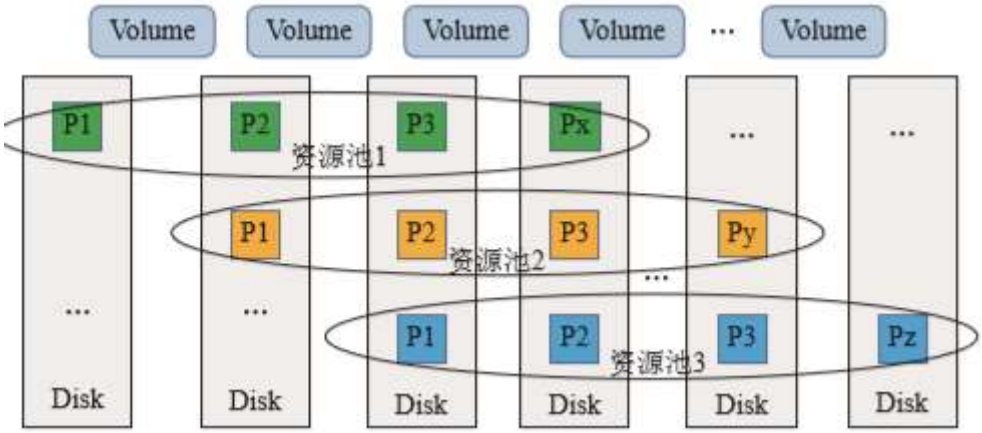


图 4-22 分布式存储架构

在分布式存储中，每块硬盘分为若干个数据分片（Partition），每个 Partition 只属于一个资源池，Partition 是数据多副本的基本单位，也就是说多个数据副本指的是多个 Partition。系统自动确保多个数据副本尽可能分布在不同的服务器上（服务器数大于数据副本数时），同时，系统自动保证多个数据副本之间的数据强一致性，然后 Partition 中的数据以 Key-Value 的方式存储。分布式存储直接对上层提供卷设备（Volume），没有 LUN 的概念，使用简单。系统自动确保每个硬盘上的主用 Partition 和备用 Partition 数量相当，避免出现集中的热点，所有硬盘都可以用做资源池的热备盘，单个资源池最多支持数百上千块硬盘。

我们来简单了解一下分布式存储的架构，它的主要模块如下图 4-23 所示：

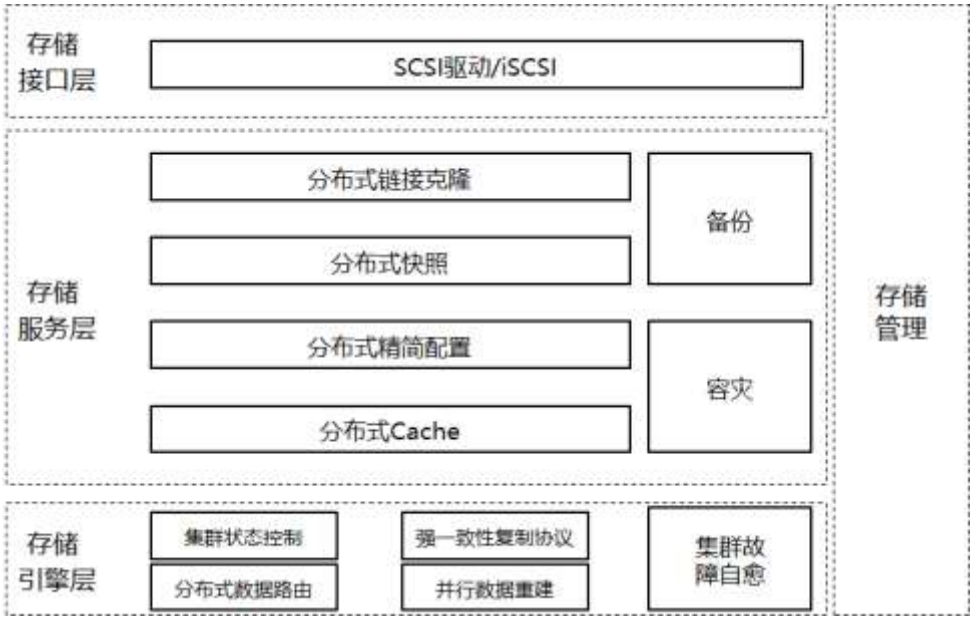


图 4-23 分布式存储模块

存储接口层：通过 SCSI 接口驱动向操作系统、数据库提供卷设备。

存储服务层：提供各种存储高级特性，如快照、链接克隆、精简配置、分布式 Cache、容灾备份等。

存储引擎层：分布式存储系统存储基本功能，包括管理状态控制、分布式数据路由、强一致性复制技术、集群故障自愈与并行数据重建等。

存储管理层：实现分布式存储系统软件的安装部署、自动化配置、在线升级、告警、监控和日志等 OM 功能，同时为用户提供 Portal 界面。

使用分布式存储进行数据写入时，应用程序不会看到底层的硬盘，只会看到分布式存储提供的存储池，应用程序的写入请求下发到存储池后，数据会被复制成多份（具体几份由用户手动设置），然后下发到不同的硬盘中，当所有写操作都返回完成的消息后，整个写操作才算完成。

在应用程序读数据时，并不会读取所有的副本数据，而是会读取主用副本中的数据，在主用副本不可用时，才会读取其它副本数据。

常见的分布式存储产品有 ceph（开源）、HDFS、FusionStorage（华为）、vSAN（VMware）等。

使用分布式存储有以下优势：

- 性能卓越

分布式存储系统通过创新的架构把分散的、低速的 SATA/SAS 机械硬盘组成一个高效的类 SAN 存储池设备，提供比 SAN 设备更高的 I/O，把性能发挥到极致。分布式存储系统一般支持用 SSD 替代 HDD 作为高速存储设备，用 InfiniBand 网络替代 GE/10GE 网络提供更高的带宽，为对性能要求极高的大数据量实时处理场景提供完美的支持。分布式存储系统采用无状态的分布式软件机头，机头部署在各个服务器上，突破集中式机头的性能瓶颈。单个服务器上软件机头只占用较少的 CPU 资源，却能提供比集中式机头更高的 IOPS。分布式存储系统实现了计算与存储的融合，缓存和带宽都均匀分布在各个服务器节点上。分布式存储系统集群内各服务器节点的硬盘使用独立的 I/O 带宽，不存在独立存储系统中大量磁盘共享计算设备和存储设备之间有限的带宽的问题。其将服务器的部分内存用作读缓存，NVDIMM 用作写缓存，数据缓存均匀分布在各节点上，所有服务器的缓存总容量远大于采用外置独立存储的方案。即使采用大容量低成本的 SATA 硬盘，分布式存储系统仍然可以发挥很高的 I/O 性能，整体性能提升 1～3 倍，同时提供更大的有效容量。

- 全局负载均衡

分布式存储系统的实现机制确保了上层应用对数据的 I/O 操作均匀分布在不同服务器的不同硬盘上，不会出现局部的热点，实现全局负载均衡。第一，系统自动将数据块打散后存储在不同服务器的不同硬盘上，冷热不均的数据会均匀分布在不同的服务器上，不会出现集中的热点。第二，数据分片分配算法保证了主用副本和备用副本在不同服务器和不同硬盘上的均匀分布，换句话说，每块硬盘上的主用副本和备副本数量是均匀的。第三，扩容节点或者故障减容节点时，数据恢复重建算法保证了重建后系统中各节点负载的均衡性。

- 分布式 SSD 存储

分布式存储系统通过支持高性能应用设计的 SSD 存储系统，可以拥有比传统的机械硬盘（SATA/SAS）更高的读写性能。特别是 PCIe 卡形式的 SSD，会带来更高的带宽和 I/O，采用 PCIe 2.0 x8 的接口，可以提供高达 3.0GB 的 I/O 读写带宽。SSD 的 I/O 性能可以达到 4KB 数

据块，100%随机，提供高达 600K 的持续随机读 IOPS 和 220K 的持续随机写 IOPS。前面说了这么多参数，大家可能不了解，只需要知道分布式存储使用 SSD 盘时有很高的读写速度，但是 SSD 普遍存在一个问题，那就是写寿命的问题，在采用 SSD 时，分布式 SSD 存储系统通过多种机制和措施增强了可靠性。

- 高性能快照

分布式存储系统提供了快照机制，将用户在某个时间点的逻辑卷数据的状态保存下来，作为后续导出数据、恢复数据使用。分布式存储系统的快照数据基于 DHT 机制来存储，快照不会引起原卷性能下降。对一块容量为 2TB 的硬盘来说，完全在内存中构建索引只需要几十 MB 空间，通过一次 Hash 查找即可判断有没有做过快照，以及最新快照的存储位置，因此效率很高。

- 高性能链接克隆

分布式存储系统可以基于增量快照提供链接克隆机制，基于一个快照创建出多个克隆卷，刚创建的克隆卷的数据内容与快照中的数据内容一致，后续对克隆卷的修改也不会影响原始的快照和其它克隆卷。分布式存储系统支持批量部署虚拟机卷，实现秒级批量创建上百个虚拟机卷。克隆卷可支持创建快照、从快照恢复以及再次作为母卷进行克隆操作。

- 高速 InfiniBand 网络

为突破分布式存储环境中存储交换瓶颈，分布式存储系统可以部署为高带宽应用设计的 InfiniBand 网络。

4.3 虚拟化存储和非虚拟化存储

我们这里讲的存储虚拟化是狭义的虚拟化，仅指集群是否有文件系统，如果有即为虚拟化存储，如果没有即为非虚拟化存储。这里的文件系统可以是 NFS 文件系统，也可以是虚拟化集群的文件系统。如果没有文件系统，虚拟化集群需要直接调动逻辑卷使用。

前面我们讲了最底层的物理磁盘以及由物理磁盘组成的集中式存储和分布式存储，无论是集中式存储，还是分布式存储，使用 RAID 或副本机制后，会形成一个物理卷，但大多数情况下都

不会把整个物理卷挂载给上层的应用（操作系统或者虚拟化系统，此处我们单指虚拟化系统）使用。因为如果把挂载整个物理卷后，上层应用会格式化所有空间，存储空间使用完后，虽然可以通过添加硬盘的方式进行扩容，但是扩容后需要重新格式化，数据可能丢失。所以，一般会将物理卷组成卷组，然后再将卷组划分成多个逻辑卷，上层应用使用逻辑卷的空间。

在云计算中，虚拟化程序会对逻辑卷进行格式化，各个厂商的虚拟化文件系统各不相同，VMware 使用的是 VMFS（Virtual Machine File System），华为使用的是 VIMS（Virtual Image Manage System），它们都属于高性能的集群文件系统，可以使虚拟化超出单个系统的限制，让多个计算节点同时访问一个整合后的集群式存储池。计算集群的文件系统可以保证某台服务器或某个应用软件不会完全控制对文件系统的访问。

以 VIMS 为例，它是一种基于 SAN 存储的集群文件系统，因此使用 FusionStorage 提供存储空间时，只能是非虚拟化存储。通过 VIMS，FusionCompute 以文件的形式管理虚拟机镜像及配置文件。VIMS 通过分布式锁机制来确保集群中数据读写的一致性。虚拟化程序使用的最小存储单位为 LUN（Logical Unit Number），而与 LUN 对应的是卷 Volume，卷是存储系统内部的管理对象，LUN 是 Volume 的对外体现。LUN 和卷都是从一个资源池（Pool）中划分出来的。

使用虚拟化后，LUN 可以分为 Thick LUN 和 Thin LUN。

Thick LUN 的中文名称是“传统非精简 LUN”，它是 LUN 的一种类型，支持虚拟资源分配，能够以较简便的方式进行创建、扩容和压缩操作。Thick LUN 创建完成后就会从存储池中分配满额的存储空间，即 LUN 的大小完全等于分配的空间。因此它拥有较高的、可预测的性能。

Thin LUN 的中文名称是精简 LUN，它也是 LUN 的一种类型，支持虚拟资源分配，能够以较简便的方式进行创建、扩容和压缩操作。Thin LUN 在创建时，可以设置初始分配容量。创建完成后，存储池只会分配初始容量大小的空间，剩余的空间还在存储池中。当 Thin LUN 已分配存储空间的使用率达到阈值时，存储系统才会从存储池中再划分一定的存储空间给 Thin LUN，直到 Thin LUN 等于设定的全部容量为止，因此它拥有较高的存储空间利用率。

Thick LUN 和 Thin LUN 的主要区别如下：

- 空间分配

Thick LUN 在创建时存储池会分配所有需要的存储空间。

Thin LUN 是一种按需分配的空间组织方法，它在创建时存储池不会分配所有需要的存储空间，而是根据使用情况动态分配。

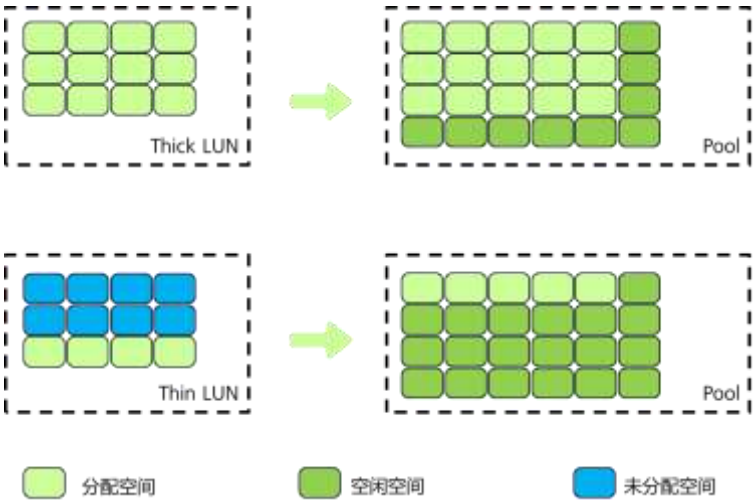


图 4-24 Thin LUN 示意图

- 空间回收

“空间回收”指的是释放存储池中的资源，并且这些释放的资源可以被其它 LUN 重新使用。Thick LUN 没有空间回收的概念，因为它在创建时就占用了所有存储池分配给它的存储空间，即使 Thick LUN 中的数据被删除，存储池分配给它的存储空间还是会被占用，不能被其它 LUN 使用。但是手动删除整个不再使用的 Thick LUN，则其对应的存储空间也会被回收。

Thin LUN 不仅能随空间占用率的增大，自动分配新的存储空间，而且当 Thin LUN 中的文件删除时也可以实现存储空间的释放，实现存储空间的反复利用，大大提升存储空间的利用率。

Thin LUN 空间回收如下图 4-25 所示：

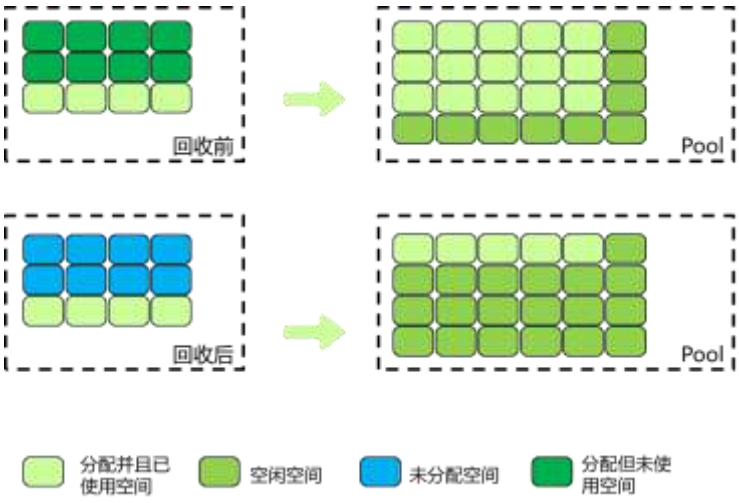


图 4-25 空间回收示意图

- 性能区别

Thick LUN 因为一开始就会拥有所有的分配空间，所以 Thick LUN 在顺序读写时拥有较高的性能，但是会造成部分存储空间的浪费。

Thin LUN 因为是实时分配空间，每次扩容时，需要重新增加容量，后台重新格式化，这时性能会受到一定的影响。而且每次分配空间后可能会导致硬盘中的存储空间不连续，这样一来，硬盘在读写数据时会在寻找存放位置上花费较多的时间，在顺序读写时对性能也有一定的影响。

- 使用场景

Thick LUN：

1. 对性能要求较高的场景。
2. 对存储空间利用率不太敏感的场景。
3. 对成本要求不太高的场景。

Thin LUN：

1. 对性能要求一般的场景。
2. 对存储空间利用率比较敏感的场景。
3. 对成本要求比较敏感的场景。

4. 在实际应用中很难预估存储空间的场景。

除虚拟化集群的文件系统外，常见的文件系统还有 NAS 存储文件系统（NFS 和 CIFS，在前文有介绍）和操作系统文件系统。

文件系统是大量文件的分层组织结构，操作系统有了文件系统后，我们看到的数据才能以文件或文件夹的形式体现，才可以随时地复制、粘贴、删除和恢复。文件系统使用目录的方式把数据组织成分层结构，目录就是保存文件指针的地方。所有的文件系统都要维护这个目录，操作系统只维护本机的目录，而集群需要维护 NAS 或者集群文件系统形成的共享目录。

常见的操作系统文件格式包括：FAT32（微软）、NTFS（微软）、UFS（UNIX）、EXT2/3/4（Linux）等。

如下图 4-26 所示，描述了操作系统文件系统的工作过程：

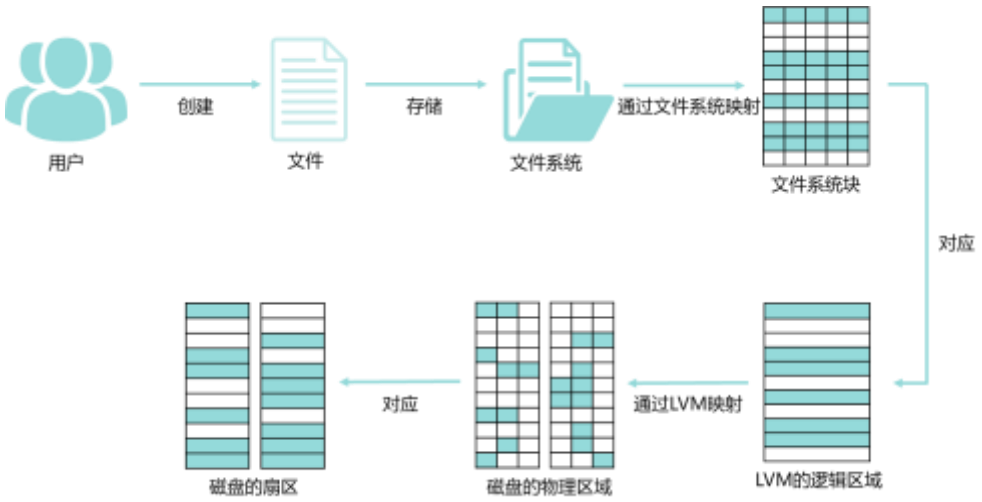


图 4-26 操作系统文件系统

首先，用户或者应用程序创建了文件或文件夹；

第二，这些文件或文件夹会被保存在文件系统上；

第三，文件系统将这些文件对应的数据映射到文件系统块上；

第四，文件系统块与由逻辑卷形成的逻辑区域进行对应；

第五，通过操作系统或 LVM 将逻辑区域映射到物理磁盘的物理区域，也就是我们前面讲过的逻辑卷与物理卷的对应；

最后，物理区域所对应的物理卷可能会包含一块或多块物理磁盘。

4.4 虚拟机磁盘

前面我们在讲虚拟化本质时讲过，虚拟机是由两部分组成的——配置文件和磁盘文件，每个虚拟机磁盘会对应一个磁盘文件，用来保存用户真实的数据。

如果使用虚拟化存储，所有的磁盘文件都会以文件的形式存放到文件系统形成的共享目录中，如果使用非虚拟化存储，每个磁盘文件对应一个 LUN。无论是文件还是 LUN，从用户和操作系统视角来看，它与普通硬盘一样，在系统的“硬件资源”中看到的都是一个硬盘。对管理员来说，创建虚拟机时需要同时为虚拟机创建磁盘来保存数据，磁盘文件在虚拟机配置文件中对应的是几行配置信息。

与其它文件一样，虚拟机磁盘文件也有自己的固定格式，常见的虚拟机磁盘格式如下：

表 5 常见虚拟机磁盘格式

虚拟机磁盘文件格式	支持的厂商
RAW	各厂商通用
VMDK	VMware
VHD	微软Hyper-V、华为FusionCompute
QCOW	QEMU或KVM虚拟化平台专用的格式
QED	

VDI	Oracle

各厂商都有自己的虚拟机磁盘格式转换工具，可以将其它的格式转换成自己产品可用的格式，例如，华为的 Rainbow 可将第三方或者开源的虚拟机磁盘格式转换成 VHD 格式。

4.5 华为虚拟化产品的存储特性

4.5.1 华为虚拟化产品存储架构

FusionCompute 使用的存储资源可以来自本地磁盘或专用的存储设备。专用的存储设备与主机之间应通过网线或光纤进行连接。

数据存储是 FusionCompute 对存储资源中的存储单元进行的统一封装。存储资源封装成数据存储并与主机关联后，能进一步创建成若干个虚拟磁盘，供虚拟机使用。

能够封装为数据存储的存储单元包括：

- SAN 存储（包括 iSCSI 或光纤通道的 SAN 存储）上划分的 LUN。
- NAS 存储上划分的文件系统。
- FusionStorage Block 上的存储池。
- 主机的本地硬盘（虚拟化）。

这些存储单元在华为 FusionCompute 中统称为“存储设备”，而向虚拟化提供存储空间的物理存储介质被称为“存储资源”，具体如下图 4-27 所示：

图 4-27 华为存储模型

注：在完成 FusionCompute 添加存储实验后，请注意观察华为对存储所定义的逻辑架构，并确定每个逻辑层上的设备是如何添加到系统中的。比如，存储资源是需要手动添加的，而存储设备是直接扫描出来的。

在使用数据存储前，需要手动添加存储资源。如果存储资源是 IPSAN、FusionStorage 或者 NAS 存储时，需要为集群内的主机添加存储接口，然后通过这个接口与集中式存储控制器的业务接口或者 FusionStorageManager 的管理地址进行通信。如果存储资源是 FCSAN，则不需要单独添加存储接口。

添加完存储资源后，需要在 FusionCompute 界面扫描存储设备，并最终将其添加为数据存储。

数据存储可以是虚拟化或非虚拟化的，也可以将 SAN 存储上的 LUN 直接作为数据存储，供虚拟机使用，而不再创建虚拟磁盘，此过程称为裸设备映射。目前仅支持部分操作系统的虚拟机，适用于搭建数据库服务器等对磁盘空间要求较大的场景。

4.5.2 华为虚拟机磁盘特性

添加完数据存储后，就可以为虚拟机创建虚拟磁盘了。根据不同用户对虚拟机磁盘的不同需求，比如要求共享同一个虚拟机磁盘，也可能要求尽可能地节省更多的物理空间等。华为虚拟机磁盘可分为以下几种：

- 按照类型，可将虚拟机磁盘分为普通和共享：
  - 普通：普通磁盘只能提供给单个虚拟机使用。

- 共享：共享磁盘可以绑定给多个虚拟机使用。

多个虚拟机使用同一个共享磁盘时，如果同时写入数据，有可能会导致数据丢失。若使用共享磁盘，需要通过应用软件来保证对磁盘的访问控制。

- 根据配置模式，可将虚拟机磁盘分为普通、精简和普通延迟置零：
  - 普通：该模式下，磁盘分配空间即为磁盘容量，在创建过程中会将物理设备上保留的数据置零。这种格式的磁盘性能要优于其它两种磁盘格式，但创建这种格式的磁盘所需的时间可能会比创建其它类型的磁盘长。
  - 精简：该模式下，系统首次仅分配磁盘容量设置值的一部分，后续根据使用情况，逐步进行分配，直到分配总量达到磁盘容量配置值为止。同时，在此模式下，数据存储支持超分配，但是建议超分配的比例不超过 50%，即假设“总容量”为 100G，“已分配容量”建议不要超过 150G。同理，如果发现“已分配容量”大于“实际容量”，则表示该磁盘模式为精简。
  - 普通延迟置零：该模式下，磁盘容量即为磁盘分配空间，但创建时不会擦除物理设备上保留的任何数据，而从虚拟机首次执行写操作时开始会按需将其置零。其创建速度比“普通”模式快；I/O 性能介于“普通”和“精简”两种模式之间。这种磁盘的配置模式只支持数据存储类型为“虚拟化本地硬盘”或“虚拟化 SAN 存储”。
- 根据磁盘模式，可将虚拟机磁盘分为从属、独立-持久和独立-非持久：
  - 从属：快照中包含该磁盘，更改将立即并永久写入磁盘。
  - 独立-持久：更改将立即并永久写入磁盘，持久磁盘不受快照影响。
  - 独立-非持久：关闭电源或恢复快照后，丢弃对该磁盘的更改。

若选择“独立-持久”或“独立-非持久”，则对虚拟机创建快照时，不对该磁盘的数据进行快照。使用快照还原虚拟机时，不对该磁盘的数据进行还原。

如果快照后，该磁盘被解绑定且未绑定其它虚拟机，则快照恢复的虚拟机会重新绑定该磁盘，但磁盘数据不进行还原。

如果快照后，该磁盘被删除，则快照恢复的虚拟机上不存在该磁盘。

磁盘类型和配置模式一旦设定后就不支持更改，而磁盘模式设定后还可以相互转换。

注：关于华为虚拟机磁盘的特性，请关注实验手册中的《5 场景一：虚拟机磁盘应用》

5 虚拟化特性介绍

前面我们说过，云计算是一种模式，而虚拟化是一种技术，我们也说过，云计算 1.0 时代是以虚拟化为主的，那么虚拟化究竟有什么特性可以使其成为云计算中很重要的部分呢?

在介绍虚拟化特性之前，我们先回顾一下虚拟化的特点：

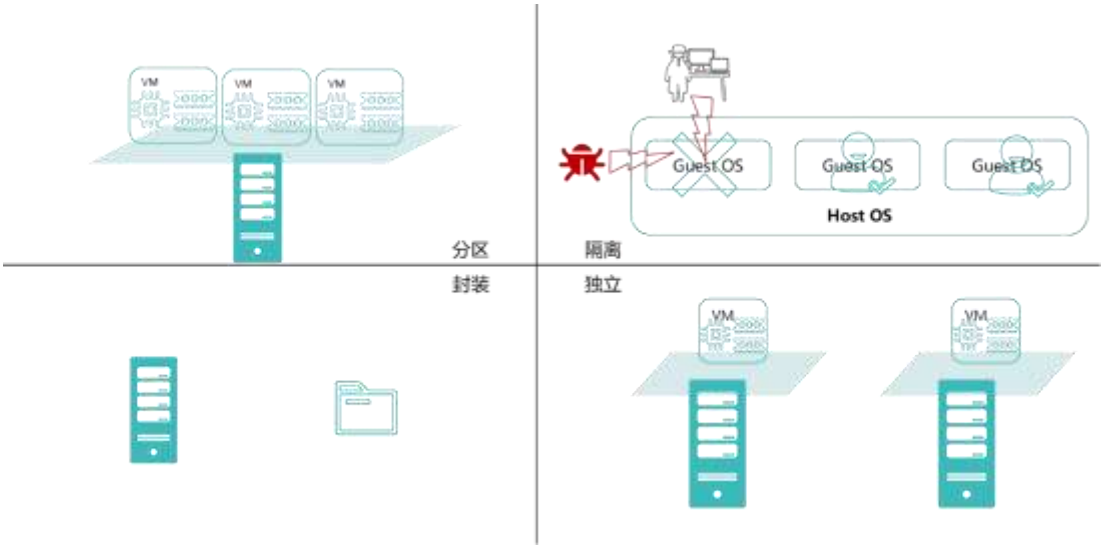


图 5-1 虚拟化特点

基于这些特点，虚拟化具备很多特性，我们从集群和虚拟机两个角度介绍。

5.1 虚拟化集群特性介绍

集群是一种把一组计算机组合起来作为一个整体向用户提供资源的方式，在虚拟化集群中可以提供计算资源、存储资源和网络资源，只有包含了这些资源以后，该集群才是完整的。

5.1.1 HA 特性

先分享一个李开复耳熟能详的故事。

早年，李开复博士曾在苹果电脑公司任职，专门负责新产品的研制和开发。

有一次，李开复与公司 CEO 史考利先生，受到美国当时最红的早间电视节目“早安美国”的邀请。在当时，能上这个收视率非常高的节目，不仅是苹果公司的荣誉，也是李开复展现个人魅力的机会。电视台方面提前和苹果公司沟通，希望他们能在电视直播中，演示苹果公司最新发明的语音识别系统，让更多消费者了解公司的新产品。

但是，该系统成功的概率大概是 90%，史考利希望把成功率提高到 99%。

李开复并没有修改程序，而是准备两台一样的电脑，如果一台电脑出了问题，立刻切换到另外一台电脑，那么根据概率原则，两台电脑出现故障的为 10%*10%=1%，那么成功率就到了 99%。

这个就是 HA 实现的基本原理：使用集群技术，克服单台物理主机的局限性，最终达到业务不中断或者中断时间减少的效果。在虚拟化中的 HA 只保证计算层面，具体来说，虚拟化层面的 HA 是虚拟机系统层面的 HA，即当一台计算节点出现故障时，在集群中的另外一台节点中能快速自动的将其启动起来。

虚拟化集群一般都会使用共享存储，我们前面讲过，虚拟机由配置文件和数据盘组成，而数据盘是保存在共享存储上的，配置文件则保存在计算节点上。当计算节点出现故障时，虚拟化管理系统（如 vCenter、VRM 等）会根据记录的虚拟机配置信息在其它节点重建出现故障的虚拟机。

在实现 HA 的过程中，有两个难点需要解决：

1 如何发现主机是否发生了故障?

2 如果虚拟机不能正常启动该怎么办?

首先看第一个问题。如果要检测到计算节点是否故障，管理员需要定期和集群内所有的节点建立通讯，一旦某个节点无法通讯，则证明该节点可能出现了故障。以华为的 FusionCompute

为例，CNA 主机和 VRM 通过心跳机制来保证 VRM 有效的感知 CNA 节点是否发生了异常，具体过程如下：

- CNA 主机侧有某个进程或服务承载着心跳机制的任务；
- 主机每间隔 3s 会向 VRM 主动上报心跳，如果连续 10 次，即 30s 内主机没有向 VRM 上报心跳，则会置此主机为“故障”状态，此时 FC-Portal 上会有“主机与 VRM 心跳异常”的告警出现；
- 主机每次向 VRM 上报心跳的时候都有超时机制，socket 连接、接收、发送超时时间均为 10s，如果 VRM 服务有异常或网络出现异常，都可能导致超时出现，而每次打印超时日志的时机=“超时探测时间间隔 3s”+“socket 超时时间 10s”= 13s 的日志时间戳；
- VRM 侧每收到一个主机侧发来的心跳就会将心跳频率 heartBeatFreq 变量设置为 10（默认为 10，此值可以通过修改配置文件修改），检测线程每 3s 会将该值减 1，同时对该参数当前值进行判断，如果<=0，则认为此值对应的主机节点异常，在 FC-Portal 上报告警，同时会把此主机异常的消息发送给 VRM 进行虚拟机 HA 机制判断。

接下来看一下第二个问题。虚拟机在其它主机上启动的时候，有可能会虚拟机上的业务无法自启动，甚至可能操作系统都无法正常启动，所以虚拟机层面的业务不能恢复的风险很大，同时业务恢复的时间也较长，这时候我们需要启用业务层面的 HA，一旦主用的虚拟机出现故障或者不能恢复时，业务会借助浮动 IP、Keepalived 等与高可用相关的技术，将业务在备用的虚拟机上恢复。

因此，虚拟机层面的 HA 一般会和应用层面的 HA 配合使用，可以缩短业务恢复的时间，提高业务恢复的几率。

除了以上问题外，如何防止脑裂也是需要考虑的问题。脑裂是由于共享存储有可能会同时被两个虚拟机执行写操作而造成的。在系统进行 HA 前，管理系统会通过心跳机制检测计算节点是

否故障，但是如果只是心跳网络出现了故障，就会造成管理系统误判计算节点故障的情况发生，这时候就可能造成脑裂现象。所以，系统在进行启动虚拟机以前会检测对应的存储是否有写操作，如果有的话，则证明主机有可能没有出现故障，这时候，系统就不会在继续执行虚拟机的启动操作了，而是直接在管理系统的页面显示 HA 成功。

如下图中的场景，当集群中第一台节点出现故障时，该节点上的虚拟机会自动迁移到其它工作正常的主机上。

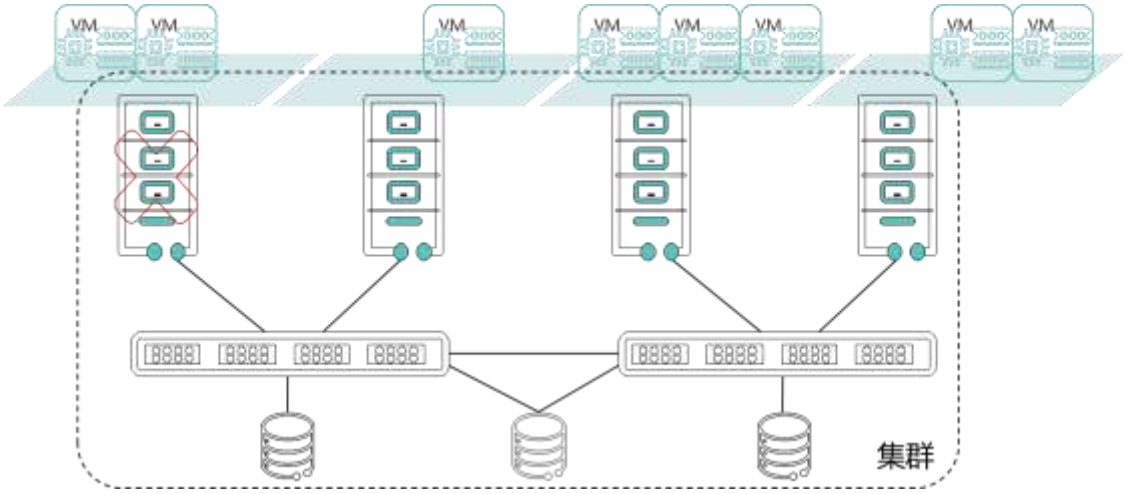


图 5-2 HA 示意图

5.1.2 负载均衡

负载均衡是一种集群技术，它将特定的业务（网络服务、网络流量等）分担给多台网络设备（包括服务器、防火墙等）或多条链路，从而提高了业务处理能力，保证了业务的高可靠性。

负载均衡有以下优势：

- 高性能：负载均衡技术将业务较均衡地分配到多台设备上，提高了整个系统的性能。
- 可扩展性：负载均衡技术可以方便地增加集群中设备或链路的数量，在不降低业务质量的前提下满足不断增长的业务需求。

- 高可靠性：单个甚至多个设备或链路发生故障也不会导致业务中断，提高了整个系统的可靠性。
- 可管理性：大量的管理工作都集中在应用负载均衡技术的设备上，设备群或链路群只需要常规的配置和维护即可。
- 透明性：对用户而言，集群等同于一个可靠性高、性能好的设备或链路，用户感知不到也不必关心具体的网络结构。增加和减少设备或链路均不会影响正常的业务。

在虚拟化环境中，负载均衡的对象一般是计算节点，判断依据为计算节点的 CPU 和内存的利用率。管理系统会在虚拟机创建和运行的过程中，感知整个集群所有物理资源的使用情况，并使用智能调度算法，确定适合虚拟机运行的最佳主机，并通过热迁移等手段将虚拟机迁移过去，从而提升全局业务体验。

如图 5-3，假设集群中有四台计算节点，现在已经运行了八台规格相同的虚拟机。管理员在创建虚拟机时，如果没有指定虚拟机运行的主机，系统会自动将该虚拟机创建在负载较少的节点上，即图中的第二台或者第四台的节点上。我们还可以看出，图中的第三台节点上运行了较多的虚拟机，在运行一段时间后，系统会自动检测出，第三台主机的负载较重，会自动将其上的虚拟机迁移到其它负载较轻的主机上。

负载的阈值可以由系统管理员指定，也可以使用系统自定义。比如，在华为 FusionCompute 中，系统自定义如果某台主机的 CPU 和内存的使用率超过 60%时，VRM 就会将此 CNA 主机上的虚拟机迁移到其它节点上，在迁移前，管理员可以将迁移任务设置为自动，也可以向管理员发通知，等管理员确认后，再进行迁移。

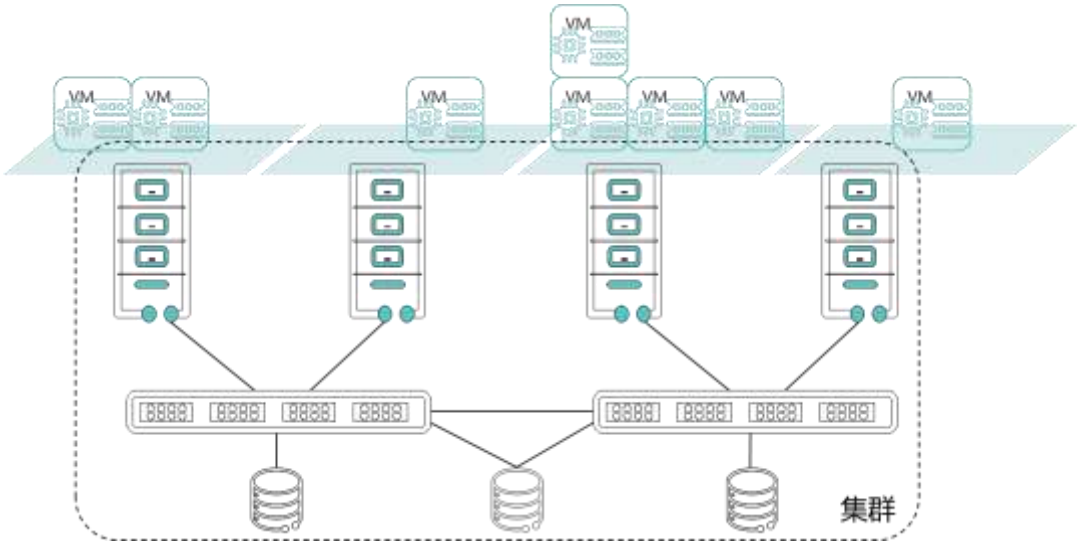


图 5-3 负载均衡示意图

5.1.3 易扩容

在传统非虚拟化的环境中，所有的业务都部署在物理机上，有可能在系统建设的初期，业务量不是很大，所以为物理机配置的硬件资源是比较低的，随着业务量的增加，原先的硬件无法满足需求，只能不停的升级硬件，比如将原先的一路 CPU 升级为两路，将 256G 的内存升级为 512G，这种扩容方式称为纵向扩容（Scale-up）。然而，物理机的所能承担的硬件是有上限的，如果业务量持续增加，最后只能更换服务器，停机割接是必然的。

在虚拟化中，将所有的资源进行池化，承载业务虚拟机的资源全部来自于这个资源池，当上面业务量持续增加的事情发生时，我们不需要升级单台服务器的硬件资源，只需要增加资源池中资源即可，具体在实施的时候，只需要增加服务器的数量即可，这种扩容方式称为水平扩容（Scale-out）。

集群支持水平扩容，所以相对传统的非虚拟化 IT，扩容更容易。如下图中，原先的集群中只有四台服务器和三台存储设备，在建设的初期，资源池中的资源是充足的，随着虚拟机慢慢的使用掉一部分资源后，管理员会发现，现有的资源不再足以支持新增的业务，这时，管理员可以在业务不繁忙的时候将所有虚拟机迁移到集中的几台物理机上，然后下电升级其它物理的硬件，也

可以在不影响任何业务的的情况下，随时在集群中增加新的服务器和存储设备。无论哪种方式，相比传统的非虚拟化环境都比较容易实施，对业务的影响也会小很多。

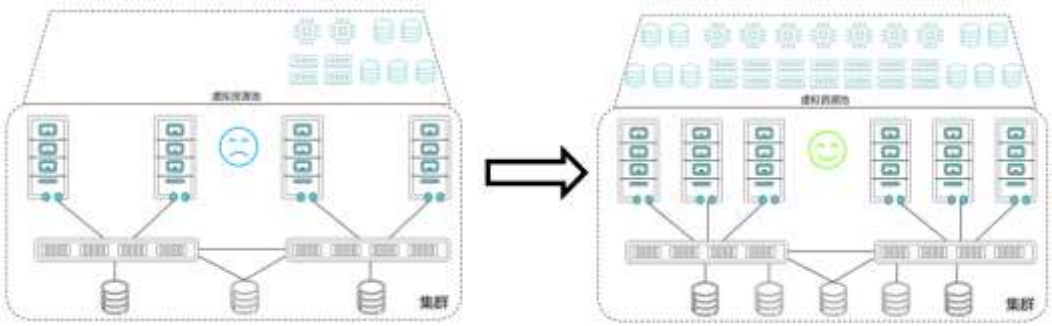


图 5-4 资源热添加

5.1.4 内存复用

内存复用是指在服务器物理内存一定的情况下，通过一定技术手段，使得虚拟机内存总和大于服务器物理内存总和，提高服务器中虚拟机密度。

内存复用的技术主要包括：内存气泡、内存置换、内存共享。一般情况下，这三种技术需要综合应用，同时生效。

内存共享：多台虚拟机共享数据内容相同的内存页。具体如下图所示。在图中，物理主机为 hypervisor 提供了 4G 的物理内存，分别分配给三台虚拟机，这三台虚拟机恰好会读取同一段物理内存中的数据，根据我们前面讲的内存虚拟化的实现方式，Hypervisor 在做内存映射时，会同时将这一段内存映射给不同的虚拟机，为了保证这一段内存里的数据不会被任意虚拟机修改，所有的虚拟机对该内存只有读操作权限，如果虚拟机需要对内存进行写操作，需要 Hypervisor 新开辟一段内存进行映射。通过内存共享技术，只有 4G 的物理内存可以分配出 6G 的虚拟内存给虚拟机。

图 5-5 内存复用

内存气泡：系统主动回收虚拟机暂时不用的物理内存，分配给需要复用内存的虚拟机。内存的回收和分配均为系统动态执行，虚拟机上的应用无感知。整个物理服务器上的所有虚拟机使用的分配内存总量不能超过该服务器的物理内存总量。具体如下图所示。

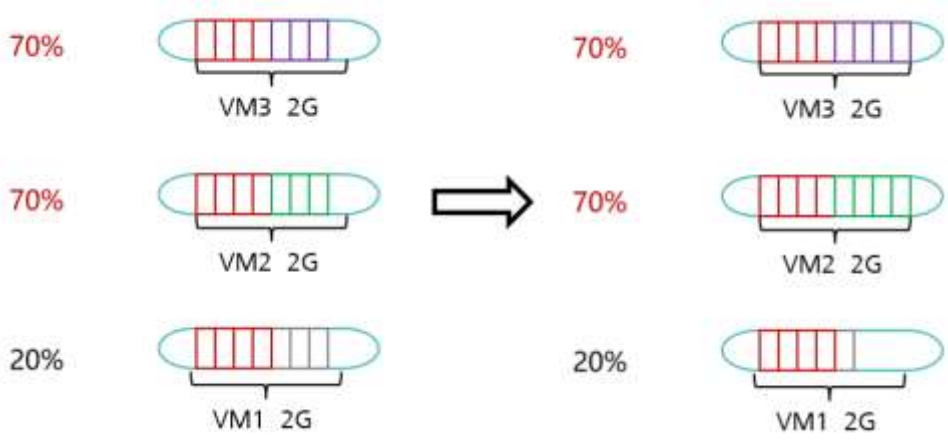


图 5-6 内存气泡

同样是三台虚拟机，每台虚拟机有 2G 的虚拟内存，然而 VM1 的内存利用率仅为 20%，而 VM2 和 VM3 的内存利用率都到了 70%，此时，系统会自动将分配给 VM1 的物理内存在后台映射给 VM2 和 VM3 以达到缓解内存压力的效果。

内存置换：将外部存储虚拟成内存给虚拟机使用，将虚拟机上暂时不用的数据存放到外部存储上。系统需要使用这些数据时，再与预留在内存上的数据进行交换。具体如下图所示。

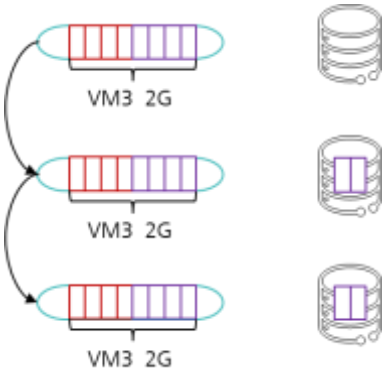


图 5-7 内存置换

内存置换类似于 windows 的虚拟内存和 linux 的 swap 分区的作用，都是使用存储模拟内存的功能，将一部分已被调用到内存但是使用频率很低的数据先放到磁盘中，当这些数据到用到的时候，这些数据会再被调回到内存中。

在部分虚拟化产品中，比如 FusionCompute 中，内存复用是基于集群设置的，打开内存复用功能后，由内存复用策略接管物理内存的分配，在内存不紧张时虚拟机可以使用全部物理内存。当出现竞争时，由内存复用策略为虚拟机实时调度内存资源，综合运用内存复用技术释放虚拟机的空闲内存，为其它虚拟机的内存需求提供条件。

使用了内存复用后，可在一定程度上降低客户的成本。

- 当计算节点的内存数量固定时，可以提高计算节点的虚拟机密度。
- 当计算节点的虚拟机密度固定时，可以节省计算节点的内存数量。

5.2 虚拟机特性介绍

5.2.1 虚拟机快速部署

我们在公有云上购买一台 ECS 时，后台很快就会为我们生成一台带有操作系统的虚拟机，所用时间会比我们自己为一台电脑安装操作系统少很多，这个就是利用了虚拟化中虚拟机可以快速部署的特性。

虚拟机的快速部署可以通过两种方式实现——按模板部署和虚拟机克隆。

模板的本质也是一台虚拟机，可以理解为虚拟机的一个副本，它同样包含了虚拟机磁盘和虚拟机的配置文件，使用模板创建虚拟机能够大幅节省配置新虚拟机和安装操作系统的时间。虚拟机模板创建后，不允许开机，也不允许启动，这样的设计是为了保证这个模板不会被其它随意的编辑而改变，同时它永远不占用集群的计算资源。使用模板部署出来的虚拟机和模板是相互独立的，如果想要更新或者重新编辑模板，需要先把模板转换为虚拟机才可，编辑完成后，再把虚拟机重新制作为模板。

虚拟机模板对于部署大量相似虚拟机是非常有用的，因为它们可以保持虚拟机的一致性，同时还可以自动将需要有差异的参数（如主机名、SID 等）进行修改。比如，现有一组测试人员需要对公司新开发的软件进行测试，那么管理员可以将安装了该软件的虚拟机制作为模板，然后快速部署出一批配置相同的虚拟机分配给不同的测试人员进行不同场景不同要求的测试，一旦测试过程中出现了任何问题，管理员可以删除故障的虚拟机，然后重新部署一台同样的虚拟机给测试人员。另外，不同的虚拟机模板中可以包含不同的软件，比如财务人员使用的模板可以预先安装好财务系统，销售人员的模板中可以安装 ERP 系统，使用这些不同的模板可以随时为对应部门的工作人员创建出符合他们需求的虚拟机。

除了模板部署虚拟机，使用虚拟机本身也可以快速的部署出一台虚拟机，这个功能称为虚拟机克隆。与使用模板部署不同，虚拟机克隆是在某个时间点对源虚拟机进行完全的复制，每个被克隆出来的虚拟机的所有设置，包括主机名、IP 地址等个性化数据，都和源虚拟机一模一样。我

们都知道，如果局域网中出现了两个完全一致的 IP 地址，系统会自动报错，所以，克隆出来的虚拟机最好不要同时开机。

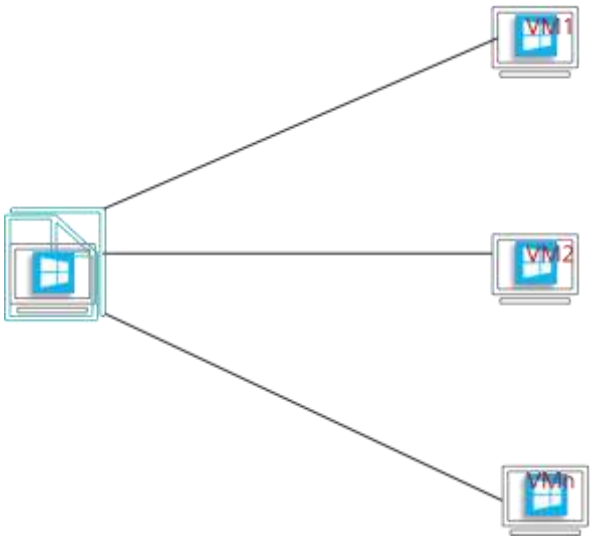


图 5-8 模板部署虚拟机

5.2.2 虚拟机资源热添加

此处的热添加指的是在虚拟机开机状态下为虚拟添加计算、存储和网络资源。

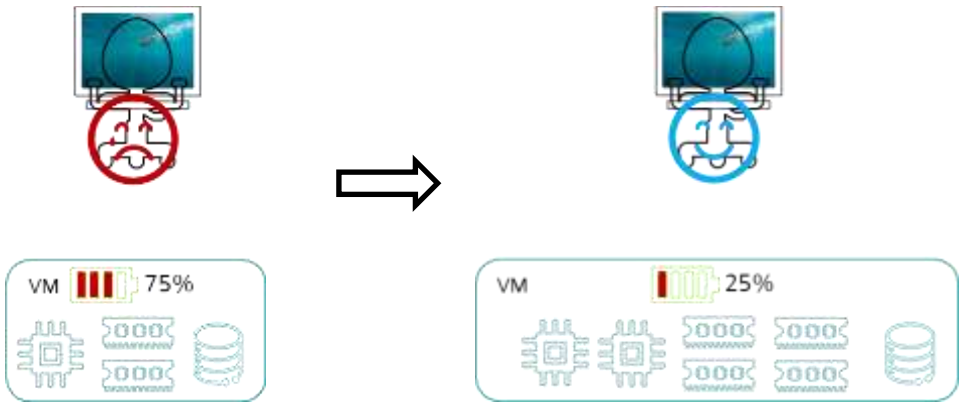


图 5-9 虚拟机资源热添加

从管理员的角度，虚拟机的 CPU、内存等参数都是配置文件里的一部分，我们可以通过修改配置文件中对应参数，来修改虚拟机的硬件配置。如图 5-8 所示，用户在使用虚拟机的过程中，CPU 和内存的使用率达到了 75%，用户侧的体验可能会非常不好，甚至有可能会影响正常的使用，这时，使用虚拟机资源热添加的功能，可以为该虚拟机在线添加 CPU 和内存资源，使客户侧的资源利用率快速降到正常水平。

除了 CPU 和内存，存储资源和网络资源也支持热添加。比如对虚拟机磁盘扩容、为虚拟机增加网卡等。

除了需要虚拟机本身支持热添加的功能，虚拟机的操作系统也要支持，才能使热添加上的资源立刻生效，否则，需要重启虚拟机，经过操作系统对硬件资源的识别后才可以使用。

大部分情况下，资源只支持热添加，而不支持减少。比如，管理员要将虚拟机的一个磁盘从 10G 扩容到 20G 可以，但是从 20G 减容到 10G 则不一定可以执行。

对存储资源的添加，除了可以对已有的磁盘进行扩容外，还支持为虚拟机增加不同的磁盘。

5.2.2.1 虚拟机 Console 控制

虚拟机不像物理机，可以使用显示器进行操作，一旦网络不通或者其它原因，很有可能就无法进行控制，这时就需要一种新的技术随时对虚拟机进行控制。

大部分不能通过接显示系统的硬件设备都会单独配置一个管理接口供用户进行管理操作，比如存储设备除了业务接口还有控制口、网络设备都会配置 Console 口等等，那么虚拟机能不能也使用同样的方式呢？答案是肯定的。

图 5-10 虚拟机远程控制

各个虚拟机厂商都为虚拟机配置了 Console 管理的功能，比如 VMware 使用的是 Remote Console，华为和 KVM 使用的 VNC 的方式。使用 Console 不代表不依赖网络，Console 登录虚拟机时，虚拟机可以不配置 IP 地址，但是虚拟机所在的计算节点需要配置 IP 地址，并且该计

算节点会作为服务端，为用户登录虚拟机提供服务，所以，用户端和服务端之间的网络需要通讯正常。

5.2.3 虚拟机快照

在日常的生活中，我们会使用拍照记录生活中的美好时刻，在虚拟化中，虚拟机的“快照”和我们生活中的拍照非常相似，可以记录某一时刻虚拟机的状态，照片可以留下那一刻的美好，快照也可以将虚拟机完全保留。通过照片我们可以找回曾经的岁月和记忆，通过快照，我们也可以将虚拟机恢复到那一刻的状态。

虚拟机快照一般应用在当对虚拟机进行升级、打补丁、测试等破坏性试验前，一旦虚拟机出现了故障，使用快照可以对虚拟机进行迅速恢复。虚拟机快照功能是通过存储系统来完成的，SNIA（存储网络行业协会）对快照（Snapshot）的定义是：关于指定数据集合的一个完全可用拷贝，该拷贝包括相应数据在某个时间点（拷贝开始的时间点）的映像。快照可以是其所表示的数据的一个副本，也可以是数据的一个复制品。图 5-11 为快照技术的示意图：

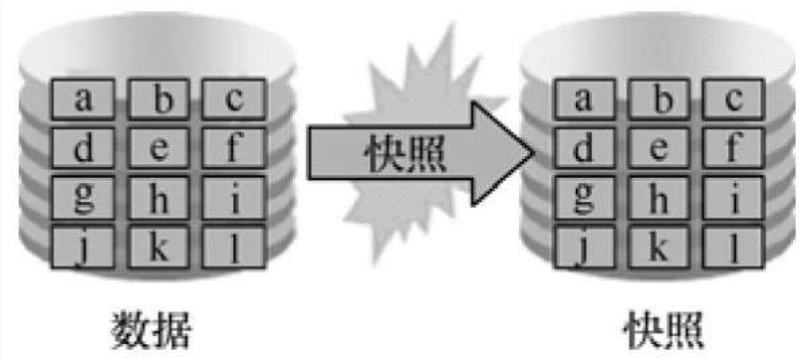


图 5-11 快照

快照技术有如下特点：

快照可迅速生成，并可用作传统备份和归档的数据源，缩小甚至消除了数据备份的窗口。

快照存储在磁盘上，可以快速直接存取，提高了数据恢复的速度。

基于磁盘的快照使存储设备有灵活和频繁的恢复点，可以快速通过不同时间点的快照简易恢复意外擦除或损坏的数据，对其进行在线数据恢复。

从具体的技术细节来讲，快照建立一个指针列表，指示读取数据的地址，当数据改变时，该指针列表能够在极短时间内提供一个实时数据，并进行复制。

常见的快照模式分为两种：写前拷贝（COW，Copy-On-Write）快照和写时重定向（ROW，Redirect-On-Write）快照。COW 和 ROW 都属于存储领域的知识，大部分厂商在创建虚拟机快照时使用的是 ROW 技术，无论是 COW 还是 ROW 都不会发生真正的物理拷贝的动作，只是做映射上的修改。我们简单了解一下 COW 和 ROW 的区别。

- COW

数据都是被记录在数据块中，当使用 COW 时，每次创建快照，系统都会产生一个新的空间，一旦数据块中的数据发生变化时，系统会做两个事情，第一是将原数据块中的数据拷贝到新的空间中，然后再将新的数据写到原先的数据块中。由于拷贝动作是在数据写之前发生，所以这种方式被称为写前拷贝。举个通俗的例子，现在有一个停车场，每个车位上都停了车子，当有新的车停进去的时候，需要先将现在车位上的汽车挪到其它车位，然后才能将新的汽车停进去。在生活中，如果你是原先车子的主人，当车场通知你为其它汽车腾位置的时候，你会不会觉的车场的管理有问题？为什么不让新的车直接停到新的车位中呢？所以就有了另外一种技术，也就是 ROW。

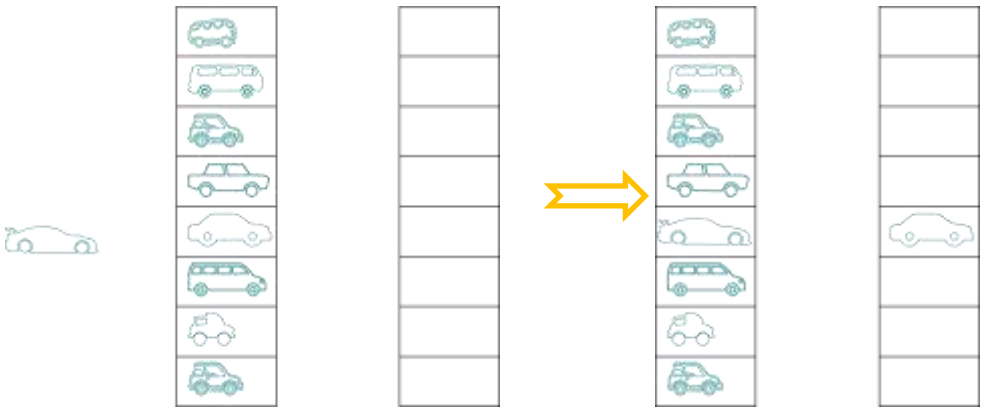


图 5-12 COW 示意图

- ROW

和 COW 一样，数据都是被记录在数据块中，在创建快照的时候也会产生一个新的空间，和 COW 不一样的是，如果有新的数据写进来的时候，原先的数据不会变化，而是将新数据写到新

的空间中。还拿停车场打比方，如果有新汽车停进来的时候，管理员直接将其引导到新的车位即可。

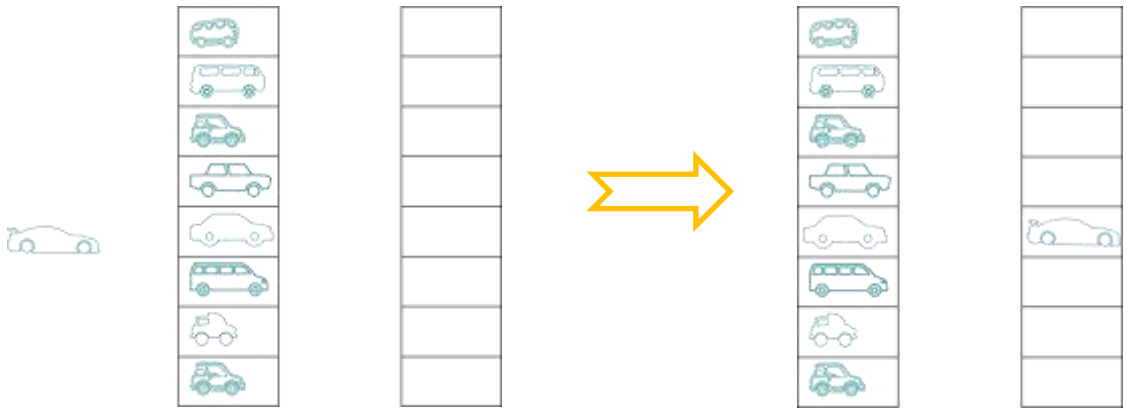


图 5-13 ROW 示意图

根据 COW 和 ROW 的简单介绍，我们会发现 COW 写操作会被进行两次，而 ROW 只进行一次，所以，单纯从对虚拟机创建快照来说，ROW 要比 COW 更加具备优势。

一台虚拟机可以创建多个快照，形成一个快照链，对其中的任意一个快照做操作时，都不会对其它的快照产生影响。

5.2.4 NUMA

NUMA，全称为非统一内存访问（Non Uniform Memory Access Architecture），它是一种可以提高数据读写速度的技术。

在近代，计算机单个 CPU 的运算速度已经到达瓶颈，所以设计者采用多路多核 CPU 的方式来提高计算机的运算速度。CPU 和内存是通过北桥的方式互相连接，由于 CPU 数量增多了，内存也相应的进行了增加，这就导致在北桥上的响应速度变慢，且越来越明显，于是，设计者就把内存平均的绑给每个 CPU，这样就可以避免共享北桥而出现的拥塞。如图 5-14 所示：

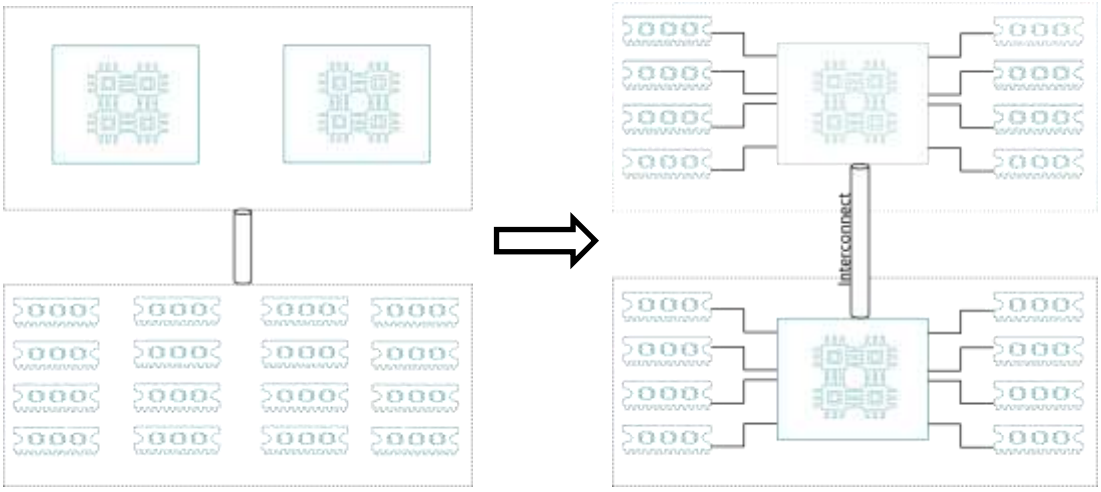


图 5-14 多路 CPU 排列对比

修改完以后，内存和 CPU 做了绑定，CPU 从绑定的内存（Local Access）中读取数据的响应时间较短，而如果跨 CPU 访问内存（Remote Access）读取数据的响应时间较长，既然 Local Access 速度快，那么就让程序在运行时全部使用一个 CPU 和与其相绑定的内存，这样就可以提高工作效率，这就是 NUMA。

使用了 NUMA 后，它会把 CPU 和与其绑定的内存当做一个 NUMA Node，每个 node 都有自己内部 CPU、总线和内存，如果跨 Node 进行访问，需通过 CPU 之间的 Interconnect。

对应到虚拟化中，使用 NUMA 技术可以让虚拟机使用同一 NUMA Node 上的硬件资源，以提高虚拟机的响应速度。

5.3 华为虚拟化产品特性

华为 FusionCompute 作为先进的虚拟化产品，同样支持我们前面介绍的集群和虚拟机的特性，并且在通用虚拟化特性的基础上做了增强。如果要使用全部的特性，需要为虚拟机安装 Tools。

虚拟机 Tools 有两部分组成——内核态的硬件驱动和用户态的 vm-agent 进程。通过内核态的硬件驱动，管理员可以为该虚拟机进行快照、热迁移、在线调整虚拟机规格和设置网卡

QOS 等操作；通过用户态的代理进程，VRM 可以获取到虚拟机的 IP 地址、状态等，还可以对虚拟机进行软关机、重启等操作。

Tools 在不同的操作系统上安装的方式不一样，在安装时请查看产品文档，按照对应的方式正确安装。安装完 Tools 以后需要对 Tools 镜像进行卸载，否则会影响虚拟机的热迁移、HA 等操作。

5.3.1 集群特性

在华为 FusionCompute 中，支持内存复用和 NUMA，所用到的技术和通用虚拟化一致，这里就不做过多的讲解了。

5.3.1.1 HA

FusionCompute 中，集群支持多种 HA 策略，用户可根据自己的需求进行设置。

- 主机故障处理策略
  - 原主机恢复虚拟机：当虚拟机所在的主机发生故障时，必须等待主机恢复后，系统才会在原主机重新启动虚拟机。
  - HA 虚拟机：当虚拟机所在的主机发生故障时，系统会按照集群中设置的虚拟机启动策略重新选择主机启动虚拟机。
- 主机数据存储故障策略
  - 主机数据存储故障处理策略：建议选择“不处理”。当选择“不处理”时，若数据存储故障时间超过“策略延迟”设置的时间，则虚拟机内部会收到 IO ERROR，导致操作系统异常。除了“不处理”，还可以选择“HA 虚拟机”，选择后若数据存储故障时间超过“策略延迟”设置的时间，系统会按照集群中设置的虚拟机启动策略重新选择主机启动虚拟机。
  - 策略延迟（分钟）：执行“主机数据存储故障处理策略”前的延迟时间。在延迟时间内，虚拟机的 IO 会在后端驱动进行重发处理。策略延迟时间建议设置一个比较大的数值，最大值不超过 14400。

- 虚拟机故障和响应策略
  - 不处理
  - 重启虚拟机
  - HA 虚拟机

注：在 FusionCompute 6.3 中，虚拟机故障只支持 Windows 蓝屏检测。

5.3.1.2 电源管理（DPM）

电源管理自动化功能会周期性地检查集群中服务器的资源使用情况，如果集群中资源利用率不足，则会将多余的主机下电节能，下电前会将虚拟机迁移至其它主机；如果集群资源过度利用，则会将离线的主机上电，以增加集群资源，减轻主机的负荷。

- 开启电源管理自动化时，必须同时开启计算资源调度自动化。待主机上电后，便自动对虚拟机进行负载均衡。
- 电源管理的分时阈值设置，可以满足不同时段的调度需求。在业务平稳运行的时段，建议将电源管理自动化的级别调低，以减小对业务的影响。
- 电源管理自动化会检测集群的资源利用率，当资源利用率低于轻载阈值的持续时间超过下电主机评估历史时间（默认为 40 分钟）时，才会对主机进行下电操作。同样，当资源利用率高于重载阈值的持续时间超过上电主机评估历史时间（默认为 5 分钟）时，才会对主机进行上电操作。下电主机评估历史时间和上电主机评估历史时间支持在高级选项中进行自定义配置。

DPM 在实际工作中可以实现绿色节能环保，比如某单位使用基于 FusionCompute 的桌面云，到了晚上十点时，可以将用户使用的虚拟机全部集中迁移到部分物理主机上，然后将其它的物理主机下电。到第二天早上七点时，在把全部的物理主机上电，然后将虚拟机通过负载均衡平均的分配到各个物理主机上。在晚上下电物理机时，如果设备集中在某个区域时，我们同时可以将配套的其它设备，如空调、新风系统，也进行下电，最大限度的进行节能。

5.3.1.3 DRS 规则

DRS 和上面提到的 DPM 都是负载均衡的一部分。DRS 可以通过一定的规则，为系统在进行负载均衡时提供迁移参考。

- 聚集虚拟机：列出的虚拟机必须在同一主机上运行，一个虚拟机只能被加入一条聚集虚拟机规则中。
- 互斥虚拟机：列出的虚拟机必须在不同主机上运行，一个虚拟机只能被加入一条互斥虚拟机规则中。
- 虚拟机到主机：关联一个虚拟机组和主机组并设置关联规则，指定所选的虚拟机组的成员是否能够在特定主机组的成员上运行。

如果不同的规则发生冲突时，规则的调度优先级如下：

- 第一优先级：规则类型为“虚拟机到主机”，规则是“必须在主机组上运行”和“禁止在主机组上运行”的。
- 第二优先级：规则类型为“聚集虚拟机”和“互斥虚拟机”的。
- 第三优先级：规则类型为“虚拟机到主机”，规则是“应该在主机组上运行”和“不应该在主机组上运行”的。

5.3.1.4 IMC

在 FusionCompute 中，设置集群的 IMC 策略，使虚拟机可以在不同 CPU 类型的主机之间进行迁移。

目前 IMC 策略仅支持 Intel 不同型号 CPU 的热迁移，其它厂商的 CPU 不能配置该功能。

IMC 可以确保集群内的主机向虚拟机提供相同的 CPU 功能集，即使这些主机的实际 CPU 不同，也不会因 CPU 不兼容而导致迁移虚拟机失败。

设置集群 IMC 策略时，如果集群中有主机或虚拟机，则必须满足下面的条件：

- 集群下主机的 CPU 功能集必须等于或高于设置的目标基准功能集。

- 集群下运行或休眠状态的虚拟机 CPU 功能集必须等于或低于目标基准功能集。如果存在不满足条件的虚拟机，需要将该虚拟机关机或迁移出该集群后设置。

5.3.2 虚拟机特性

5.3.2.1 虚拟机资源 QOS

- CPU QoS

虚拟机的 CPU QoS 用于保证虚拟机的计算资源分配，隔离虚拟机间由于业务不同而导致的计算能力相互影响，满足不同业务对虚拟机计算性能的要求，最大程度复用资源，降低成本。

创建虚拟机时，可根据虚拟机预期部署业务对 CPU 的性能要求而指定相应的 CPU QoS。不同的 CPU QoS 代表了虚拟机不同的计算能力。指定 CPU QoS 的虚拟机，系统对其 CPU 的 QoS 保障，主要体现在计算能力的最低保障和资源分配的优先级。

CPU QoS 包含如下三个参数：

- CPU 资源份额

  CPU 份额定义多个虚拟机在竞争物理 CPU 资源的时候按比例分配计算资源。

  以一个主频为 2.8GHz 的单核物理主机为例，如果上面运行有三台单 CPU 的虚拟机。三个虚拟机 A，B，C，份额分别为 1000，2000，4000。当三个虚拟机 CPU 满负载运行时，会根据三个虚拟机的份额按比例分配计算资源。份额为 1000 的虚拟机 A 的计算能力约为 400MHz 的，份额为 2000 的虚拟机 B 获得的计算能力约为 800MHz，份额为 4000 的虚拟机 C 获得的计算能力约为 1600MHz。(以上举例仅为说明 CPU 份额的概念，实际应用过程中情况会更复杂)。

  CPU 份额只在各虚拟机竞争计算资源时发挥作用，如果没有竞争情况发生，有需求的虚拟机可以独占物理 CPU 资源，例如，如果虚拟机 B 和 C 均处于空闲状态，虚拟机 A 可以获得整个物理核即 2.8GHz 的计算能力。

- CPU资源预留

CPU 预留定义了多个虚拟机竞争物理 CPU 资源的时候分配的最低计算资源。

如果虚拟机根据份额值计算出来的计算能力小于虚拟机预留值，调度算法会优先按照虚拟机预留值的能力把计算资源分配给虚拟机，对于预留值超出按份额分配的计算资源的部分，调度算法会从主机上其它虚拟机的 CPU 上按各自的份额比例扣除，因此虚拟机的计算能力会以预留值为准。

如果虚拟机根据份额值计算出来的计算能力大于虚拟机预留值，那么虚拟机的计算能力会以份额值计算为准。

以一个主频为 2.8GHz 的单核物理机为例，如果运行有三台单 CPU 的虚拟机 A、B、C，份额分别为 1000、2000、4000，预留值分别为 700MHz、0MHz、0MHz。当三个虚拟机满 CPU 负载运行时：

- 虚拟机 A 如果按照份额分配，本应得 400MHz，但由于其预留值大于 400MHz，因此最终计算能力按照预留值 700MHz 算。
- 多出的（700-400）MHz 按照 B 和 C 各自的份额比例从 B 和 C 处扣除。
- 虚拟机 B 获得的计算能力约为（800-100）MHz，虚拟机 C 获得的计算能力约为（1600-200）MHz。

CPU 预留只在各虚拟机竞争计算资源的时候才发挥作用，如果没有竞争情况发生，有需求的虚拟机可以独占物理 CPU 资源。例如，如果虚拟机 B 和 C 均处于空闲状态，虚拟机 A 可以获得整个物理核即 2.8GHz 的计算能力。

- CPU资源限额

控制虚拟机占用物理 CPU 资源的上限。以一个两 CPU 的虚拟机为例，如果设置该虚拟机 CPU 上限为 3GHz，则该虚拟机的两个虚拟 CPU 计算能力被限制为 1.5GHz。

- 内存 QoS

提供虚拟机内存智能复用功能，依赖内存预留比。通过内存气泡等内存复用技术将物理内存虚拟出更多的虚拟内存供虚拟机使用，每个虚拟机都能完全使用分配的虚拟内存。该功能可最大

程度的复用内存资源，提高资源利用率，且保证虚拟机运行时至少可以获取到预留大小的内存，保证业务的可靠运行。

系统管理员可根据用户实际需求设置虚拟机内存预留。内存复用的主要原则是：优先使用物理内存。

内存 QoS 包含如下三个参数：

- 内存资源份额

  内存份额定义多个虚拟机竞争内存资源的时候按比例分配内存资源。

  在虚拟机申请内存资源，或主机释放空闲内存（虚拟机迁移或关闭）时，会根据虚拟机的内存份额情况按比例分配。

  不同于 CPU 资源可实时调度，内存资源的调度是平缓的过程，内存份额策略在虚拟机运行过程中会不断进行微调，使虚拟机的内存获取量逐渐趋于比例。

  以 6G 内存规格的主机为例，假设其上运行有三台 4G 内存规格的虚拟机，内存份额分别为 20480、20480、40960，那么其内存分配比例为 1:1:2。当三个虚拟机内部均逐步加压，策略会根据三个虚拟机的份额按比例分配调整内存资源，最终三个虚拟机获得的内存量稳定为 1.5G、1.5G、3G。

  内存份额只在各虚拟机竞争内存资源时发挥作用，如果没有竞争情况发生，有需求的虚拟机可以最大限度地获得内存资源。例如，如果虚拟机 B 和 C 没有内存压力且未达到预留值，虚拟机 A 内存需求压力增大后，可以从空闲内存、虚拟机 B 和 C 中获取内存资源，直到虚拟机 A 达到上限或空闲内存用尽且虚拟机 B 和 C 达到预留值。以上面的例子，当份额为 40960 的虚拟机没有内存压力（内存资源预留为 1G），那么份额为 20480 的两个虚拟机理论上可以各获得最大 2.5G 的内存。

- 内存资源预留

内存预留定义多个虚拟机竞争内存资源的时候分配的内存下限，能够确保虚拟机在实际使用过程中一定可使用的内存资源。

预留的内存被会虚拟机独占。即，一旦内存被某个虚拟机预留，即使虚拟机实际内存使用量不超过预留量，其它虚拟机也无法抢占该虚拟机的空闲内存资源。

- 内存资源限额

控制虚拟机占用物理内存资源的上限。在开启多个虚拟机时，虚拟机之间会相互竞争内存资源，为了使虚拟机的内存得到充分利用，尽量减少空闲内存，用户可以在创建虚拟机时设置虚拟机配置文件中的内存上限参数，使服务器分配给该虚拟机的内存大小不超过内存上限值。

- 网络 QoS

网络 QoS 策略提供带宽配置控制能力，QoS 功能不支持同一主机上虚拟机之间的流量限制。包含如下方面：

- 基于端口组成员接口发送方向与接收方向的带宽控制
- 基于端口组的每个成员接口提供流量整形、带宽优先级的控制能力。

6 云计算发展趋势

有人说过，云计算是土壤，人工智能是盛开的花朵，而大数据则是花朵生长需要的水分和肥料。花朵和肥料不断地在变化，所以土壤也需要随着它们一起发展，本章就介绍和云计算相关的其它领域以及在其它领域刺激下云计算的发展趋势。

6.1 与云计算相关的其它领域

- 物联网（IoT）

物联网是新一代信息技术的重要组成部分，其英文名称是“The Internet of things”。顾名思义，“物联网就是物物相连的互联网”。这有两层意思：第一，物联网的核心和基础仍然是互联网，是在互联网基础上的延伸和扩展的网络；第二，其用户端延伸和扩展到了任何物品与物品之间，进行信息交换和通信。因此，物联网的定义是通过射频识别（RFID）、红外感应器、全球定位系统、激光扫描器等信息传感设备，按约定的协议，把任何物品与互联网相连接，进行信息交换和通信，以实现对物品的智能化识别、定位、跟踪、监控和管理的一种网络。

物联网使用的主要技术有：

RFID：电子标签属于智能卡的一类，物联网概念是 1999 年 MIT Auto-ID 中心主任 Ashton 教授提出来的，RFID 技术在物联网中主要起“使能”（Enable）作用；

传感器技术：借助于各种传感器，探测和集成包括温度、湿度、压力、速度等物质现象的网络，也是温总理“感知中国”提法的主要依据之一；

嵌入式系统技术：嵌入式系统是硬件 和软件紧密结合的专用计算机系统。“嵌入式”反映了这些系统通常是更大系统中的一个组成部分。

- 云计算、大数据与人工智能

人工智能（后面简称 AI）一直都被认为是可以改变世界的，甚至是可以终结人类的，从云计算到大数据，再到 AI，各种创新且具有革命性意义的技术推动着 ICT 产业乃至整个社会迈进数字化、智能化时代。

大数据最初归属云计算领域，是云计算的一个应用。没有云计算，大数据无法真正的实现。2011 年，大数据出现在 Gartner 的新型技术成熟度曲线中第一阶段的技术触发期；2013 年，当云计算进入泡沫幻灭期之后，大数据才步入了期望膨胀期；2014 年，大数据迅速进入了泡沫幻灭期，并开始与云计算齐头并进。今天，我们已经很难将大数据与云计算割裂开来，大数据需要云计算的支撑，云计算为大数据提供不可或缺的平台。但值得注意的是，大数据也成就了云计算，没有了大数据的云计算将会变得无的放矢。

在 IT 领域最早期的时候，数据量并不大，原来才有多少数据？现在大家都去看电子书，上网看新闻了，在我们 80 后小时候，信息量没有那么大，也就看看书、看看报，一个星期的报纸加起来才有多少字？如果你不在一个大城市，一个普通的学校的图书馆加起来也没几个书架，是后来随着信息化的到来，信息才会越来越多。

大数据中的数据一般分为三种：

结构化的数据：即有固定格式和有限长度的数据。例如填的表格就是结构化的数据，国籍：中华人民共和国，民族：汉，性别：男，这都叫结构化数据。

非结构化的数据：现在非结构化的数据越来越多，就是不定长、无固定格式的数据，例如网页，有时候非常长，有时候几句话就没了；例如语音，视频都是非结构化的数据。

半结构化数据：是一些 XML 或者 HTML 的格式的，不从事技术的可能不了解，但也没有关系。

其实数据本身不是有用的，必须要经过一定的处理。例如你每天跑步带个手环收集的也是数据，网上这么多网页也是数据，我们称为 Data。数据本身没有什么用处，但数据里面包含一个很重要的东西，叫做信息（Information）。

数据十分杂乱，经过梳理和清洗，才能够称为信息。信息会包含很多规律，我们需要从信息中将规律总结出来，称为知识（Knowledge），而知识改变命运。信息是很多的，但有人看到了信息相当于白看，但有人就从信息中看到了电商的未来，有人看到了直播的未来，所以人家就牛了。如果你没有从信息中提取出知识，天天看朋友圈也只能在互联网滚滚大潮中做个看客。

有了知识，然后利用这些知识去应用于实战，有的人会做得非常好，这个东西叫做智慧（Intelligence）。有知识并不一定有智慧，例如好多学者很有知识，已经发生的事情可以从各个角度分析得头头是道，但一到实干就歇菜，并不能转化成为智慧。而很多的创业家之所以伟大，就是通过获得的知识应用于实践，最后做了很大的生意。

所以数据的应用分这四个步骤：数据、信息、知识、智慧。

图 6-1 数据应用四步骤

智慧是很多商家都想要的。你看我收集了这么多的数据，能不能基于这些数据来帮我做下一步的决策，改善我的产品。例如让用户看视频的时候旁边弹出广告，正好是他想买的东西；再如让用户听音乐时，另外推荐一些他非常想听的其它音乐。

用户在我的应用或者网站上随便点点鼠标，输入文字对我来说都是数据，我就是要将其中某些东西提取出来、指导实践、形成智慧，让用户陷入到我的应用里面不可自拔，上了我的网就不想离开，手不停地点、不停地买。

很多人说双十一我都想断网了，我老婆在上面不断地买买买，买了 A 又推荐 B，老婆大人说，“哎呀，B 也是我喜欢的啊，老公我要买”。你说这个程序怎么这么牛，这么有智慧，比我还了解我老婆，这是怎么做到的呢?

第一个步骤叫数据的收集；第二个步骤是数据的传输；第三个步骤是数据的存储；第四个步骤是数据的处理和分析；第五个步骤是对于数据的检索和挖掘。

当数据量很小时，很少的几台机器就能解决。慢慢的，当数据量越来越大，最牛的服务器都解决不了问题时，怎么办呢？这时就要聚合多台机器的力量，大家齐心协力一起把这个事搞定，众人拾柴火焰高。

对于数据的收集，就 IoT 来讲，外面部署这成千上万的检测设备，将大量的温度、湿度、监控、电力等数据统统收集上来；就互联网网页的搜索引擎来讲，需要将整个互联网所有的网页都下载下来。这显然一台机器做不到，需要多台机器组成网络爬虫系统，每台机器下载一部分，同时工作，才能在有限的时间内，将海量的网页下载完毕。

对于数据的传输，一个内存里面的队列肯定会被大量的数据挤爆掉，于是就产生了基于硬盘的分布式队列，这样队列可以多台机器同时传输，随你数据量多大，只要我的队列足够多，管道足够粗，就能够撑得住。

对于数据的存储，一台机器的文件系统肯定是放不下的，所以需要一个很大的分布式文件系统来做这件事情，把多台机器的硬盘打成一块大的文件系统。

对于数据的分析，可能需要对大量的数据做分解、统计、汇总，一台机器肯定搞不定，处理到猴年马月也分析不完。于是就有分布式计算的方法，将大量的数据分成小份，每台机器处理一小份，多台机器并行处理，很快就能算完。例如著名的 Terasort 对 1 个 TB 的数据排序，相当于 1000G，如果单机处理，怎么也要几个小时，但并行处理 209 秒就完成了。

说到这里，大家想起云计算了吧。当想要干这些活时，需要很多的机器一块做，真的是想什么时候要就什么时候要，想要多少就要多少。

例如大数据分析公司的财务情况，可能一周分析一次，如果要把这一百台机器或者一千台机器都在那放着，一周用一次非常浪费。那能不能需要计算的时候，把这一千台机器拿出来；不算的时候，让这一千台机器去干别的事情?

谁能做这个事儿呢？只有云计算，可以为大数据的运算提供资源层的灵活性。而云计算也会部署大数据放到它的 PaaS 平台上，作为一个非常非常重要的通用应用。因为大数据平台能够使得多台机器一起干一个事儿，这个东西不是一般人能开发出来的，也不是一般人玩得转的，怎么也得雇个几十上百号人才能把这个玩起来。

所以说就像数据库一样，其实还是需要有一帮专业的人来玩这个东西。现在公有云上基本上都会有大数据的解决方案了，一个小公司需要大数据平台的时候，不需要采购一千台机器，只要到公有云上一点，这一千台机器都出来了，并且上面已经部署好了的大数据平台，只要把数据放进去算就可以了。

虽说有了大数据，人的欲望却不能够满足。虽说在大数据平台里面有搜索引擎这个东西，想要什么东西一搜就出来了。但也存在这样的情况：我想要的东西不会搜，表达不出来，搜索出来的又不是我想要的。

例如音乐软件推荐了一首歌，这首歌我没听过，当然不知道名字，也没法搜。但是软件推荐给我，我的确喜欢，这就是搜索做不到的事情。当人们使用这种应用时，会发现机器知道我想要什么，而不是说当我想要时，去机器里面搜索。这个机器真像我的朋友一样懂我，这就有点人工智能的意思了。

人们很早就在想这个事情了。最早的时候，人们想象，要是有一堵墙，墙后面是个机器，我给它说话，它就给我回应。如果我感觉不出它那边是人还是机器，那它就真的是一个人工智能的东西了。

怎么才能做到这一点呢？人们就想：我首先要告诉计算机人类的推理的能力。你看人重要的是什么？人和动物的区别在什么？就是能推理。要是把我这个推理的能力告诉机器，让机器根据你的提问，推理出相应的回答，这样多好?

其实目前人们慢慢地让机器能够做到一些推理了，例如证明数学公式。这是一个非常让人惊喜的一个过程，机器竟然能够证明数学公式。但慢慢又发现其实这个结果也没有那么令人惊喜。因为大家发现了一个问题：数学公式非常严谨，推理过程也非常严谨，而且数学公式很容易拿机器来进行表达，程序也相对容易表达。

然而人类的语言就没这么简单了。比如今天晚上，你和你女朋友约会，你女朋友说：如果你早来，我没来；你等着，如果我早来；你没来，你等着！这个机器就比较难理解了，但人都懂。所以你和女朋友约会，是不敢迟到的。

因此，仅仅告诉机器严格的推理是不够的，还要告诉机器一些知识。但告诉机器知识这个事情，一般人可能就做不来了。可能专家可以，比如语言领域的专家或者财经领域的专家。

语言领域和财经领域知识能不能表示成像数学公式一样稍微严格点呢？例如语言专家可能会总结出主谓宾定状补这些语法规则，主语后面一定是谓语，谓语后面一定是宾语，将这些总结出来，并严格表达出来不久行了吗？

后来发现这个不行，太难总结了，语言表达千变万化。就拿主谓宾的例子，很多时候在口语里面就省略了谓语，别人问：你谁啊？我回答：我张三。但你不能规定在语音语义识别时，要求对着机器说标准的书面语，这样还是不够智能，就像罗永浩在一次演讲中说的那样，每次对着手机，用书面语说：请帮我呼叫某某某，这是一件很尴尬的事情。

人工智能这个阶段叫做专家系统。专家系统不易成功，一方面是知识比较难总结，另一方面总结出来的知识难以教给计算机。因为你自己还迷迷糊糊，觉得似乎有规律，就是说不出来，又怎么能够通过编程教给计算机呢？

于是人们想到：机器是和人完全不一样的物种，干脆让机器自己学习好了。

机器怎么学习呢？既然机器的统计能力这么强，基于统计学习，一定能从大量的数字中发现一定的规律。

当然，真正基于统计的学习算法比这个简单的统计复杂得多。

然而统计学习比较容易理解简单的相关性：例如一个词和另一个词总是一起出现，两个词应该有关系；而无法表达复杂的相关性。并且统计方法的公式往往非常复杂，为了简化计算，常常做出各种独立性的假设，来降低公式的计算难度，然而在现实生活中，具有独立性的事件是相对较少的。

于是人类开始从机器的世界，反思人类的世界是怎么工作的。

人类的脑子里面不是存储着大量的规则，也不是记录着大量的统计数据，而是通过神经元的触发实现的，每个神经元有从其它神经元的输入，当接收到输入时，会产生一个输出来刺激其它神经元。于是大量的神经元相互反应，最终形成各种输出的结果

人工智能可以做的事情非常多，例如可以鉴别垃圾邮件、鉴别黄色暴力文字和图片等。这也是经历了三个阶段的。

第一个阶段依赖于关键词黑白名单和过滤技术，包含哪些词就是黄色或者暴力的文字。随着这个网络语言越来越多，词也不断地变化，不断地更新这个词库就有点顾不过来。

第二个阶段时，基于一些新的算法，比如说贝叶斯过滤等，你不用管贝叶斯算法是什么，但是这个名字你应该听过，这个一个基于概率的算法。

第三个阶段就是基于大数据和人工智能，进行更加精准的用户画像和文本理解和图像理解。

由于人工智能算法多是依赖于大量的数据的，这些数据往往需要面向某个特定的领域（例如电商，邮箱）进行长期的积累，如果没有数据，就算有人工智能算法也白搭，所以人工智能程序很少像前面的 IaaS 和 PaaS 一样，将人工智能程序给某个客户安装一套，让客户去用。因为给某个客户单独安装一套，客户没有相关的数据做训练，结果往往是很差的。

但云计算厂商往往是积累了大量数据的，于是就在云计算厂商里面安装一套，暴露一个服务接口，比如您想鉴别一个文本是不是涉及黄色和暴力，直接用这个在线服务就可以了。这种形势的服务，在云计算里面称为软件即服务，SaaS （Software AS A Service）

于是人工智能程序作为 SaaS 平台进入了云计算。

终于云计算的三兄弟凑齐了，分别是 IaaS、PaaS 和 SaaS。所以一般在一个云计算平台上，云、大数据、人工智能都找得到。一个大数据公司，积累了大量的数据，会使用一些人工智能的算法提供一些服务；一个人工智能公司，也不可能没有大数据平台支撑。

- 云计算、物联网和大数据

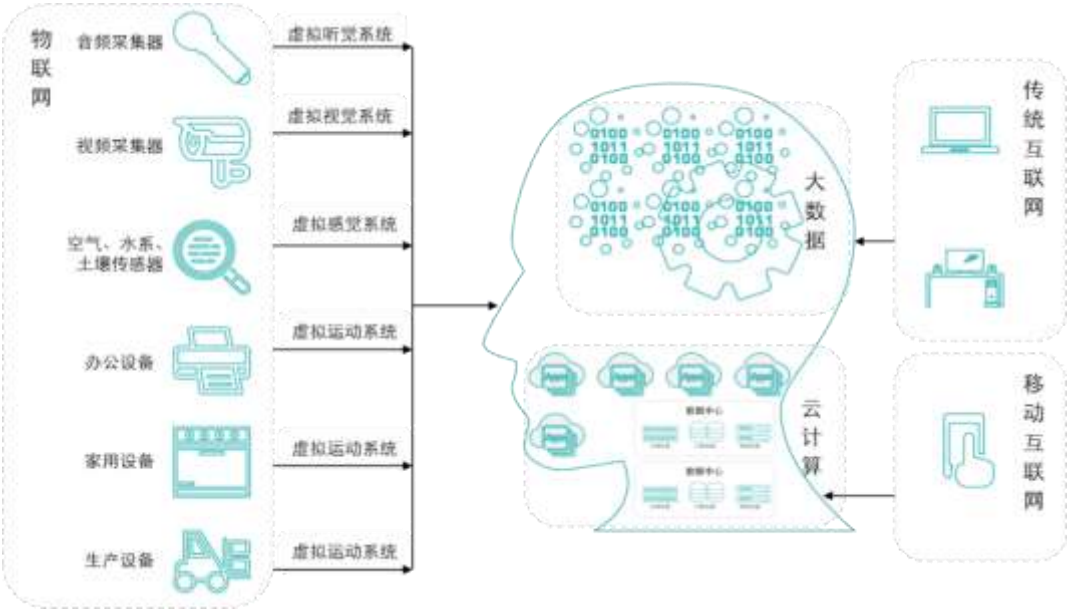


图 6-2 云计算、物联网和大数据之间的关系

前面说过，云计算是土壤，大数据是肥料，那么物联网就是肥料的制作原料。

物联网是大数据的重要来源，物联网对应了互联网的感觉和运动审计系统，大数据大数据代表了互联网的信息层，是互联网智慧和意识产生的基础，大数据在进行数据挖掘和分析时，恰好需要的是云计算快速弹性伸缩等特性，因此，云计算和物联网推动了大数据的发展，大数据也促进了云计算和物联网的进步。

- 5G 和云计算

云计算是计算和互联网发展的产物，移动互联网正在进入 5G 时代。华为轮值主席胡厚崑在伦敦举行的 2018 全球移动宽带论坛上表示：“5G 技术将赋予信息通信技术力量，并引发技术和商业的诸多变革。”

5G 将使网络连接成为一个平台。所有的东西都会在网上，并在默认情况下保持在线状态。到 2025 年，预计将有 400 亿智能设备实现 5G。胡厚崑认为，在未来，不上网要比上网更困难。

胡厚崑指出 5G 可能带来的五个革命性变化：连接平台化、永远在线、全云化、重新定义终端以及连续性。他还表示世界正在拥抱云技术。这将让每个人随时随地获取可用信息。像 Cloud X 这样的新商业模式将开始出现，在这种模式中，设备基于云在推动。很快就会有更多的 Cloud X 应用，比如 Cloud PC 工作站、Cloud 游戏和 Cloud VR/AR，像微软，育碧和 EA 这样的公司已经在研究这些了。

6.2 实现云计算的技术介绍

6.2.1 容器

容器（Container）是一种轻量级的虚拟化技术，所谓的轻量级虚拟化，就是使用了一种操作系统虚拟化技术，这种技术允许一个操作系统上用户空间被分割成几个独立的单元在内核中运行，彼此互不干扰，这样一个独立的空间，就被称之为一个容器。

使用容器技术后，应用程序可以在几乎任何地方一相同的方式运行。开发人员可以在自己的电脑上常见并测试好容器，无须任何修改就能够在其它的虚拟机或者物理机上运行。

容器也是一种虚拟化技术，所以不可避免要和虚拟机进行对比。

容器由两部分组成：

- 应用程序
- 应用程序运行环境，比如应用程序需要的库等。

容器是一个软件的轻量级独立可执行软件包，包含运行它所需的一切：代码，运行时，系统工具，系统库，设置。不管环境如何，集装箱化软件都可以运行相同的 Linux 和 Windows 应用程序。容器将软件与其周围环境隔离开来，例如开发环境和测试环境之间的差异，并有助于减少

在同一基础架构上运行不同软件的团队之间的冲突。虚拟机除了部署应用容器中包含的这些外，还需要包含操作系统，这是二者最大的区别。

那么容器究竟解决了什么问题呢?

当代软件的架构非常复杂，软件环境配置就成了现在软件开发最大的麻烦，开发者常常会说：“它在我的机器可以跑了”，言下之意就是这个软件在别的电脑上不一定能跑得起来，最大的原因是软件所在的操作系统的设置及各种库和组件的安装，只有他们全部正确，软件才能运行起来。比如一个软件是用 JAVA 开发的，计算必须有配套的引擎、依赖和环境变量的配置。那么能不能软件能不能带环境安装嗯，也就是说，安装软件的时候，把原始环境也一模一样的复制过来，这样不就解决问题了。

也有人说，那我用虚拟机模板，不也一样可以解决这个问题吗?

前面我们讲了，虚拟机是带操作系统的，所以会有以下几个缺陷:

1、资源占用太多

虚拟机会独占一部分内存和硬盘空间。它运行的时候，其它程序就不能使用这些资源了。哪怕虚拟机里面的应用程序，真正使用的内存只有 1MB，虚拟机依然需要几百 MB 的内存才能运行。

2、冗余步骤多

虚拟机是完整的操作系统，一些系统级别的操作步骤，往往无法跳过，比如用户登录。

3、启动起来较慢

启动操作系统需要多久，启动虚拟机就需要多久，所以可能要等几分钟，应用程序才能真正运行。

对比容器，上面的这是缺陷恰恰是容器的优势。

1、启动快

容器里面的应用，直接就是底层系统的一个进程，而不是虚拟机内部的进程。所以，启动容器相当于启动本机的一个进程，而不是启动一个操作系统，速度就快很多。

2、体积小

容器只要包含用到的组件即可，而虚拟机是整个操作系统的打包，所以容器文件比虚拟机文件要小很多。

3、资源占用少

容器只占用需要的资源，不占用那些没有用到的资源；虚拟机由于是完整的操作系统，不可避免要占用所有资源。另外，多个容器可以共享资源，虚拟机都是独享资源。

提到容器时，我们会最常听到的就是 Docker，它和容器时什么关系呢?

Docker 属于 Linux 容器的一种封装，提供简单易用的容器使用接口。它是目前最流行的 Linux 容器解决方案。Docker 将应用程序与该程序的依赖，打包在一个文件里面。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。总体来说，Docker 的接口相当简单，用户可以方便地创建和使用容器，把自己的应用放入容器。容器还可以进行版本管理、复制、分享、修改，就像管理普通的代码一样。

Docker 的主要用途，目前有三类。

- 提供一次性的环境。比如，本地测试他人的软件、持续集成的时候提供单元测试和构建的环境。
- 提供弹性的云服务。因为 Docker 容器可以随开随关，很适合动态扩容和缩容。
- 组建微服务架构。通过多个容器，一台机器可以跑多个服务，因此在本机就可以模拟出微服务架构。

Docker 的核心组件包括:

- Docker 客户端 – Docker Client

- Docker 守护进程 - Docker daemon
- Docker 镜像 - Image
- Registry
- Docker 容器 - Container

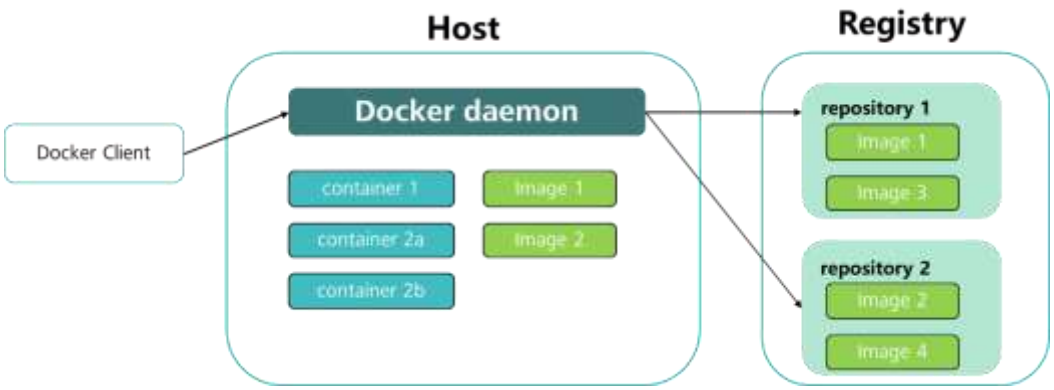


图 6-3 Docker 架构

Docker 采用的是 Client/Server 架构。客户端向服务器发送请求，服务器负责构建、运行和分发容器。客户端和服务器可以运行在同一个 Host 上，客户端也可以通过 socket 或 REST API 与远程的服务器通信。

Docker 客户端：最常用的 Docker 客户端是命令行。通过命令行我们可以在主机上构建和运行容器。

Docker 守护进程：Docker 服务器组件指的是 Docker daemon，它运行在 Linux 的操作系统上，负责创建、运行、监控容器，同时也构建和存储镜像。

Docker 镜像：它是 Docker 的关键组件，用来创建 Docker 容器。Docker 镜像类似虚拟机模板，只可读。如果需要修改，需要先转换为容器，修改完以后再转为镜像。

Docker 容器：Docker 容器是 Docker 镜像运行的实例。

Registry：存放 Docker 镜像的仓库，Registry 可分为私有和公有两种。

6.2.2 OpenStack

6.2.2.1 什么是 OpenStack

OpenStack 是目前最为流行的开源云操作系统框架。自 2010 年 6 月首次发布以来，经过数以千计的开发者和数以万计的使用者的共同努力，OpenStack 不断成长，日渐成熟。目前，OpenStack 的功能强大而丰富，已经在私有云、公有云、NFV 等多个领域得到了日益广泛的生产应用。与此同时，OpenStack 已经受到了 IT 业界几乎所有主流厂商的关注与支持，并催生出大量提供相关产品和服务的创业企业，在事实上成为了开源云计算领域的主流标准。时至今日，围绕 OpenStack 已经形成了一个繁荣而影响深远的生态系统，OpenStack 已经是云计算时代一个无法回避的关键话题。可以说，不了解 OpenStack，就无法理解当今云计算技术的发展，也无法把握云计算产业的脉搏。因此，本章将对 OpenStack 的相关要点进行概括介绍。

在 OpenStack 官网，将 OpenStack 定义成一个云操作系统。关于云操作系统，我们可以借助 PC 电脑操作系统来类比，进行理解。

操作系统，是计算机系统领域里一个至关重要的概念。有了操作系统，我们才能将计算机系统中的各类软硬件整合起来，形成一个能够完成各类处理任务的完整系统，为用户提供服务。这个描述较为抽象，但结合到日常生活与工作中的实例，就清楚易懂多了。无论是服务器和个人电脑上的 Linux、Windows，还是手机上的 Android、IOS， 都是操作系统的常见实例。 无论是服务器、个人电脑还是手机上的操作系统，本质上，其核心功能都可以概括为五个方面，即资源接入与抽象、资源分配与调度、应用生命周期管理、系统管理维护和人机交互支持。换言之，只有具备了以上这五个方面的主要功能，一个操作系统才能够实现各类软硬件的整合，让系统具备为用户提供服务的能力。

具体而言：（1）资源接入与抽象，是指将各类硬件设备，如 CPU、内存、本地硬盘、网卡等， 接入到系统中，并将其抽象为操作系统可以识别的逻辑资源，以此作为操作系统对各类硬件资源实施管理的基础；（2）资源分配与调度，是指利用操作系统的资源管理能力，将上述不同的硬件资源，按照需求的类型和数量，分配给不同的系统软件或应用软件，供其使用；（3）应用

生命周期管理，是指协助用户实现各类应用软件在操作系统上的安装、升级、启动、停止、卸载等管理操作；（4）系统管理维护，是指协助系统管理员实现对系统自身的各类配置、监控、升级等管理操作；（5）人机交互支持，指提供必要的人机界面，支持系统管理员和用户对系统实施各类操作。

与之对应，一个完整的云操作系统，同样应该能够具备上述五个方面的主要功能。其核心区别只是在于，云操作系统需要管理的，是一个由大量软硬件组成的分布式的云计算系统，而一个普通操作系统需要管理的，则是一台服务器、一台个人电脑，或者一部手机。

针对云操作系统，上述五项主要功能的内容应该是：（1）资源接入与抽象，是指将各类服务器、存储、网络设备等硬件资源，通过虚拟化的或者可软件定义的方式，接入到云计算系统中，并将其抽象为云操作系统可以识别的计算、存储、网络等资源池，以此作为云操作系统对各类 硬件资源实施管理的基础；（2）资源分配与调度，是指利用云操作系统的资源管理能力，将前述的不同资源，按照不同的云租户对于资源类型与数量的不同需求，将资源分配给各个租户，以及不同租户的不同应用；（3）应用生命周期管理，是指协助租户实现各类云应用在云操作系统上的安装、启动、停止、卸载等管理操作；（4）系统管理维护，是指协助系统管理员实现对于云计算系统的各类管理与运维操作；（5）人机交互支持，指提供必要的人机界面，支持系统管理员和普通租户对系统实施各类操作。

由上述介绍可以看出，虽然云操作系统比我们日常接触的操作系统复杂很多，但其最为关键的五项主要功能，其实是可以一一对应的。通过这种对应，我们可以更为直观地理解云操作系统这个概念。而 OpenStack，则是实现云操作系统的关键组件，或者说，是构建一个完整的云操作系统的框架。

要构建一个完整的云操作系统，需要对大量软件组件进行有机整合，让它们协同工作，共同提供系统管理员和租户所需的功能与服务。而 OpenStack 本身，尚且不能独立具备一个完整云操作系统所需的全部能力。举例而言：在上面提到的云操作系统的五项主要功能中，OpenStack 不能独立实现资源接入与抽象，而需要和底层的虚拟化软件、软件定义存储、软件

定义网络等软件相配合；OpenStack 不能独立提供完善的应用生命周期管理能力，而需要在上层集成各类管理软件平台；OpenStack 自身不具备完整的系统管理维护能力，在投入生产实用时，还需要集成各类管理软件与维护工具；OpenStack 自身提供的人机界面，其功能也还不够丰富强大，等等。

由此不难看出，想在 OpenStack 基础上构建一个完整的云操作系统，需要将 OpenStack 与其它一些软件组件进行集成，以实现 OpenStack 自身并不提供的能力。因此，OpenStack 自身的准确定位， 是一个云操作系统框架。基于这个框架，可以集成不同的各类组件，实现满足不同场景需要的云操作系统，并在此基础上，最终构建完整的云计算系统。

开源，是 OpenStack 的一个重要属性。应该说，不理解开源，就不能真正理解 OpenStack 的发展历程与未来趋势。与简单地在网络上公开源代码不同，OpenStack 社区遵循的，是一种更为深入、更为彻底的开源理念。在 OpenStack 社区中，对于每一个组件，每一个特性，乃至每一行代码，其需求提出、场景分析、方案设计、代码提交、测试执行、代码合入的整个流程，都总体遵循开放原则，对公众可见，并且在最大程度上保证了社区贡献者的监督与参与。 正是这种监督与参与的机制，保证了 OpenStack 社区总体上处于一种开放与均衡的状态，避免了少数人或者少数公司、组织的绝对控制，由此保障了社区生态的健康与繁荣。同时，OpenStack 遵循了对商业最为友好的 Apache 2.0 许可，也保障了企业参与社区的商业利益，从而推动了 OpenStack 的产品落地与商业成功。通过以上介绍，可以看出，OpenStack 是一个以开源方式开发与发布的，用于构建不同场景下的云操作系统的框架性软件。深入理解这个本 质，对于深入学习和掌握 OpenStack，有着非常关键的意义。

6.2.2.2 OpenStack 与云计算系统的关系

基于前面的介绍，不难看出，OpenStack 与云计算系统之间，既紧密联系，又相互区别。

OpenStack 是构建云操作系统的框架。使用云操作系统，集成并管理各类硬件设备，并承载各类上层应用与服务，才能最终形成一个完整的云计算系统。由此可见，OpenStack 是云计

算系统的核心软件组件，是构建云计算系统的基础框架，但 OpenStack 和云计算系统并不能直接等同。

6.2.2.3 OpenStack 与计算虚拟化的关系

计算虚拟化，是很多读者非常熟悉的概念。其对应的软件实现，就是平常所说的 Hypervisor，如开源的 KVM、Xen，以及 VMware 的 vSphere、华为的 FusionCompute、微软的 Hyper-V 等。OpenStack 与计算虚拟化之间的关系，是目前仍然被频繁混淆的一个问题。理解这二者之间的联系与区别，也是理解 OpenStack 的关键之一。OpenStack 是一个云操作系统的框架。为构建完整的云操作系统，特别是为实现资源接入与抽象的功能，OpenStack 需要与虚拟化软件实施集成，从而实现对服务器的计算资源的池化。应该指出的是，在资源池化的过程中，物理资源虚拟化的功能，仍然由虚拟化软件完成。举例而言，在使用 KVM 作为 OpenStack 的虚拟化软件时，仍然由 KVM 完成将一台物理服务器虚拟为多台虚拟机的功能，而 OpenStack 负责记录与维护资源池的状态。例如，系统中一共有多少台服务器，每台服务器的资源共有多少，其中已经向用户分配了多少，还有多少资源空闲。在此基础上，OpenStack 负责根据用户的要求，向 KVM 下发各类控制命令，执行相应的虚拟机生命周期管理操作，如虚拟机的创建、删除、启动、关机等。由此可见，两相对比，OpenStack 更像是系统的控制中枢，是云操作系统的“大脑”；计算虚拟化软件则更像是系统的执行机构，是云操作系统的“肢体”。二者分工合作，共同完成对云计算系统中的计算资源池的管理，但绝不能认为 OpenStack 等同于计算虚拟化软件。

6.2.2.4 OpenStack 的设计思想

OpenStack 之所以能够取得快速的发展，除了有云计算技术和产业快速发展的大背景之外，其自身设计思想的独到之处，也起到了有力的促进作用。OpenStack 的设计思想，在总体上可以被概 括为“开放、灵活、可扩展”。本节将对此展开扼要分析。

1. 开放

OpenStack 的开放，根源于其开源模式本身。前已述及，OpenStack 的开源，不仅体现在简单的源代码开放，更体现在其设计、开发、测试、发布的全流程中。这种开源模式，总体上可以保证 OpenStack 不被个别人或个别企业所控制，在技术上不会走向封闭架构、封闭体系，从而始终呈现出良好的开放性。无论是北向的 API 标准开放，还是南向的各类软件、硬件自由接入，都是 OpenStack 开放性的充分体现。与此同时，OpenStack 也秉持了开源社区中“不重复发明轮子”的一贯理念，在设计中持续引入并充分复用各相关技术领域中的优秀开源软件，从而提升了设计与开发效率，并为软件质量提供了基本保证。

2. 灵活

OpenStack 的灵活，首先体现在其大量使用插件化、可配置的方式进行设计。最为突出的体现，就在于 OpenStack 采用插件化的方式实现不同类型计算、存储、网络资源的接入，由此实现 OpenStack 对于不同类型资源的灵活接入与管理，用一套架构实现了对于不同厂商、不同类型设备的资源池化，例如，在计算领域，可以以插件化的形式接入 KVM、Xen、vCenter、FusionCompute 等不同的 Hypervisor；在存储领域，可以以插件化的形式实现对不同厂商的存储设备，以及 Ceph、FusionStorage、vSAN 等不同的软件定义存储的管理；在网络领域，可以实现对不同的网络硬件设备，OVS、Liunx-bridge、 HAProxy 等开源网络组件，以及多种 SDN 控制器的接入。并且，这些接入都是通过可配置的方式加以选择。当在不同的资源之间进行选择时，OpenStack 自身并不需要重新打包发布，只需通过配置项选择不同的接入插件即可，非常方便。

在此基础上，OpenStack 的灵活还体现在不依赖于任何特定的商用软硬件。换言之，任何商用软硬件产品在 OpenStack 中一定是可选、可替换的，从而严格保证用户可以使用完全开源、开放的方案来构建基于 OpenStack 的云计算系统，而完全不必担心被锁定在某些特定厂商的产品之上。

3. 可扩展

OpenStack 的架构高度可扩展。具体而言，其扩展性体现在功能和系统规模两个方面。从功能视角看，OpenStack 由多个相互解耦的项目组成。不同的项目分别完成云计算系统中的不同功能，如身份认证与授权服务、计算服务、块存储服务、网络服务、镜像服务、对象存储服务等。对于一个特定场景下的云计算系统，系统设计人员可以根据实际需要决定使用 OpenStack 中的若干个项目，也可以在系统上线后，根据需求继续引入新的 OpenStack 项目。OpenStack 的一些项目自身也具有功能可扩展性。系统设计人员可以在这些项目中引入新的功能模块，在不影响项目既有功能使用的前提下，对其功能进行扩展。从系统规模视角看，OpenStack 总体上遵循了无中心、无状态的架构设计思想。其主要项目，均可实现规模水平扩展，以应对不同规模的云计算系统建设需求。在系统建成后，可根据应用负载规模的实际增长，通过增加系统管理节点和资源节点的方式，逐渐扩展系统规模。这种架构可以有效避免高额的初始建设投资，也降低了系统初始规划的难度，为云计算系统的建设者和运营者提供了充分的扩展空间。

6.2.2.5 OpenStack 架构与组成

在 2010 年 OpenStack 社区首次发布其第一个发行版——Austin 时，OpenStack 仅包含两个项目 Nova 和 Swift，仅能实现非常简单和基础的功能。时至今日，OpenStack 已经日渐成熟和强大，其组成项目也已经大大增多，仅包含在 Mitaka 版本 release notes 中的服务项目就多达 29 个。各个项目各司其责，分工合作，共同形成了一个架构灵活、功能丰富、扩展性强的云操作系统框架。

为便于读者快速了解 OpenStack 的概貌，但又不致淹没在众多的信息当中，本节将优先选择 OpenStack 中最为关键和有代表性的部分项目，进行扼要介绍，以便帮助读者更为直观地了解 OpenStack。

1. Keystone：身份认证与授权服务

将计算、存储、网络等各种资源，以及基于上述资源构建的各类 IaaS、PaaS、SaaS 层服务，在不同的用户间共享，让众多用户安全地访问和使用同一个云计算系统，是一个云操作系统

的基本能力。而实现这个能力的基础，就是一个安全可靠的身份认证与授权服务。而 Keystone 就是 OpenStack 的身份认证与授权服务项目。Keystone 负责对用户进行身份认证，并向被认定为合法的用户发放令牌（token）。用户持 Keystone 发放的令牌访问 OpenStack 的其它项目， 以使用其提供的服务。而各个组件中内嵌的令牌校验和权限控制机制，将与 Keystone 配合实现对用户身份的识别和权限级别的控制，保证只有恰当的用户能够对恰当的资源实施恰当的操作，以 此保证对不同用户资源的隔离与保护。

2. Nova：计算服务

向用户按需提供不同规格的虚拟机，是任何一个云操作系统最为基础的功能。而 Nova 就是 OpenStack 中负责提供此类计算服务的项目。Nova 的核心功能，是将大量部署了计算虚拟化软件 （即 Hypervisor） 的物理服务器统一纳入管理之下，组成一个具有完整资源视图的逻辑资源池。在此基础上，Nova 通过接收不同用户发起的请求，对资源池中的资源进行生命周期管理操作。其中最为核心的，就是虚拟机的创建、删除、启动、停止等操作。通过执行客户发起的虚拟机创建操作，Nova 将逻辑资源池中的 CPU、内存、本地存储、IO 设备等资源，组装成不同规格的虚拟机，再安装上不同类型的操作系统，最终提供给用户进行使用，由此满足用户对于计算资源的需求。

除了虚拟机资源管理服务能力之外，Nova 还通 过与 Ironic 项目配合，共同为用户提供裸机资源管理服务能力。具体而言，Nova 可以接收用户发起的裸机资源申请，然后调用 Ironic 项目的对应功能，实现对裸机的自动化选择、分配与操作系统安装部署，从而使得用户可以获得与虚拟机资源使用体验相当的物理机资源使用体验。

3. Ironic：裸机管理

Ironic 通过与 Nova 相配合，共同为用户提供裸机服务能力。

在实际工作时，Ironic 直接负责对物理服务器的管理操作。一方面，在物理服务器被纳入到资源池之中时，Ironic 负责记录物理服务器的硬件规格信息，并向 Nova 上报；另一方面，在用户发起裸机管理操作时，Ironic 负责根据 Nova 的指令，对相应的物理服务器执行具体的管理操

作动作。例如，当用户发起一个创建裸机操作时，Ironic 需要根据 Nova 调度的结果，对选定的物理服务器执行硬件初始化配置、操作系统安装等一系列具体操作，以完成裸机创建动作。

4. Glance：镜像服务

通常而言，在虚拟机被创建之后，都需要为其安装一个操作系统，以便用户使用。为此，云计算系统中往往需要预置若干不同种类、不同版本的操作系统镜像，以便用户选用。此外，在一些应用场景下，为进一步方便用户，镜像中还需要预装一些常用的应用软件，这将进一步增加镜像的种类与数量。为此，云操作系统必须具备镜像管理服务能力。Glance 就是 OpenStack 中的镜像服务项目。

Glance 主要负责对系统中提供的各类镜像的元数据进行管理，并提供镜像的创建、删除、查询、上传、下载等能力。但在正常的生产环境下，Glance 本身并不直接负责镜像文件的存储，而是仅负责保管镜像文件的元数据，本质上是一个管理前端。Glance 需要与真正的对象存储后端对接，才能共同提供完整的镜像管理与存储服务能力。

5. Swift：对象存储服务

对象存储服务，是云计算领域中一种常见的数据存储服务，通常用于存储单文件数据量较大、访问不甚频繁、对数据访问延迟要求不高、对数据存储成本较为敏感的场景。Swift 就是 OpenStack 中用于提供对象存储服务的项目。

与 OpenStack 中大部分只实现控制功能、并不直接承载用户业务的项目不同，Swift 本身实现了完整的对象存储系统功能，甚至可以独立于 OpenStack，被单独作为一个对象存储系统加以应用。

此外，在 OpenStack 系统中，Swift 也可以被用做 Glance 项目的后端存储，负责存储镜像文件。

6. Cinder：块存储服务

在典型的、基于 KVM 虚拟化技术的 OpenStack 部署方案下，Nova 创建的虚拟机默认使用各个计算节点的本地文件系统作为数据存储。这种数据存储的生命周期与虚拟机本身的生命周期相同，即当虚拟机被删除时，数据存储也随之被删除。如果用户希望获得生命周期独立于虚拟机自身的、能够持久存在的块存储介质，则需要使用 Cinder 提供的块存储服务，也称为卷服务。

Cinder 负责将不同的后端存储设备或软件定义存储集群提供的存储能力，统一抽象为块存储资源池，然后根据不同需求划分为大小各异的卷，分配给用户使用。

用户在使用 Cinder 提供的卷时，需要使用 Nova 提供的能力，将卷挂载在指定的虚拟机上。 此时，用户可以在虚拟机操作系统内看到该卷对应的块设备，并加以访问。

7. Neutron：网络服务

网络服务，是任意云操作系统 IaaS 层能力的关键组成部分。只有基于稳定、易用、高性能的云上虚拟网络，用户才能将云计算系统提供的各类资源和服务能力连接成真正满足需求的应用系统，以解决自身的实际业务需求。

Neutron 是 OpenStack 中的网络服务项目。Neutron 及其自身孵化出来的一系列子项目，共同为用户提供了从 Layer 2 到 Layer 7 不同层次的多种网络服务功能，包括 Layer 2 组网、Layer 3 组网、内网 DHCP 管理、Internet 浮动 IP 管理、内外网防火墙、负载均衡、VPN 等。整体而言，Neutron 的 Layer 2、Layer 3 服务能力已经较为成熟。时至今日，Neutron 已经取代了早期的 novanetwork，成为了 OpenStack 中 Layer 2、Layer3 的主流虚拟网络服务实现方式。与之对应，Neutron 的 Layer 4 至 Layer 7 服务能力仍在迅速发展中，目前已具备初步应用能力。

需要说明的是，OpenStack 的 DNS 即服务能力，并未包含在 Neutron 项目的功能范围当中，而是由另一个单独的项目 Designate 负责实现。

8. Heat：资源编排服务

云计算的核心价值之一，即在于 IT 资源与服务管理和使用的自动化。换言之，在引入云计算技术之后，大量在传统 IT 领域中需要依靠管理人员或用户通过手工操作实现的复杂管理任务，应当可以通过调用云操作系统提供的 API，以程序化的方式自动完成，从而显著提高 IT 系统管理的效率。

在上述提及的 IT 领域复杂管理操作中，用户业务应用系统的生命周期管理操作，即应用系统的安装、配置、扩容、撤除等，可谓是具有代表性的一类。这类操作的复杂与耗时耗力，与当前不断凸现的业务快速上线、弹性部署诉求，已经表现出明显的不适应性。Heat 项目的出现，就是为了在 OpenStack 中提供自动化的应用系统生命周期管理能力。具体而言，Heat 能够解析用户提交的，描述应用系统对资源类型、数量、连接关系要求的定义模板，并根据模板要求，调用 Nova、Cinder、Neutron 等项目提供的 API，自动实现应用系统的部署工作。这一过程高度自动化，高度程序化。同样的模板，可以在相同或不同的基于 OpenStack 的云计算系统上重复使用，从而大大提升了应用系统的部署效率。在此基础上，Heat 还可以与 OpenStack Ceilometer 项目的 Aodh 子项目相配合，共同实现对于应用系统的自动伸缩能力。这更进一步简化了部分采用无状态、可水平扩展架构的应用系统的管理，具有典型的云计算服务特征。

9. Ceilometer：监控与计量

在云计算系统中，各类资源均以服务化的形式向用户提供，用户也需要按照所使用资源的类型和数量缴费。这种基本业务形态，就要求云操作系统必须能够提供资源使用量的监控与计量能力。这正是 OpenStack 引入 Ceilometer 项目的根本动机。

Ceilometer 项目的核心功能，是以轮询的方式，收集不同用户所使用的资源类型与数量信息，以此作为计费的依据。

在此基础上，Ceilometer 可以利用收集的信息，通过 Aodh 子项目发送告警信号，触发 Heat 项目执行弹性伸缩功能。

需要说明的是，Ceilometer 项目自身并不提供计费能力。系统设计者需要将其与适当的计费模块相对接，才能实现完整的用户计费功能。目前，OpenStack 社区已经创建了 CloudKitty

项目，作为 OpenStack 社区原生的计费组件。但该项目当前尚处于较为初期的阶段，难以直接商用。

10. Horizon：图形界面

Horizon 项目是 OpenStack 社区提供的图形化人机界面。经过社区长期的开发完善，Horizon 界面简洁美观，功能丰富易用，可以满足云计算系统管理员和普通用户的基本需求，适于作为基于 OpenStack 的云计算系统的基本管理界面使用。

此外，Horizon 的架构高度插件化，灵活而易于扩展，也便于有定制化需求的系统设计人员针对具体场景进行增量开发。

注：6.2.2 小节节选自《云计算架构技术与实践》。

6.3 其它新兴技术简介

6.3.1 雾计算

雾计算（Fog Computing），在该模式中数据、（数据）处理和应用程序集中在网络边缘的设备中，而不是几乎全部保存在云中，是云计算（Cloud Computing）的延伸概念，由思科（Cisco）提出的。这个因“云”而“雾”的命名源自“雾是更贴近地面的云”这一名句。

雾计算，这个名字由美国纽约哥伦比亚大学的斯特尔佛教授起的，他当时的目的是利用“雾”来阻挡黑客入侵。后来思科首次正是提出，赋予雾计算新含义。雾计算是一种面向物联网的分布式计算基础设施，可将计算能力和数据分析应用扩展至网络“边缘”，它使客户能够在本地分析和管理数据，从而通过联接获得即时的见解。

雾计算和云计算一样，十分形象。云在天空飘浮，高高在上，遥不可及，刻意抽象；而雾却现实可及，贴近地面，就在你我身边。雾计算并非由性能强大的服务器组成，而是由性能较弱、更为分散的各类功能计算机组成，渗入工厂、汽车、电器、街灯及人们物质生活中的各类用品。

6.3.2 边缘计算

云计算从进入我们视线到今，已经发展了十几年，在商业上取得了很大的成功，边缘计算则是云计算继续发展的产物，目前还处于概念阶段。

引用 Wikipedia 对 Edge Computing 的定义，边缘计算指的是：与将数据传到远程的云端进行处理相反，边缘计算则是在靠近数据源头的网络边缘提供计算和存储资源。

通俗的说，边缘计算是去中心化或分布式的云计算，原始数据不传回云端，而是在本地完成分析。看好边缘计算的人认为计算能力正在从云端向边缘移动，因此边缘计算会成为下一个像云计算这样成功的技术爆发点。另一方面，边缘计算是驱动物联网的关键技术，因此边缘计算的推动者往往是从事物联网的人。

边缘是一个很笼统的概念，它是指接近数据源的计算基础设施，不同的边缘计算提供商往往有不同的边缘。比如美国电信公司 AT&T 的边缘就是离客户几英里的蜂窝网络基站；对于世界最大的 CDN 厂商阿卡麦，边缘则是指遍布全球的 CDN 设备；对于机场的监控设备，边缘就是覆盖整个机场无死角的高清摄像头。

根据 Fog Computing vs. Edge Computing: What’ s the Difference?（https://www.automationworld.com/fog-computing-vs-edge-computing-whats-difference）的观点，雾计算与边缘计算都是将智能和计算能力推向靠近数据来源的位置，因此他们常常被混用。他们的区别也正是计算能力（大脑）所处的位置：

- 雾计算将计算能力推向局域网，在雾节点或物联网的网关处完成数据处理。
- 边缘计算将边缘网关的智能，处理能力和通信能力直接推送到诸如可编程自动化控制器的设备。

6.3.3 微服务

微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间相互协调、相互配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用

轻量级的通信机制互相沟通（通常是基于 HTTP 的 RESTful API），每个服务都围绕着具体业务进行构建，并且能够被独立地部署到生产环境、类生产环境等。另外，应尽量避免统一的、集中式的服务管理机制，对于具体的服务，应根据业务上下文，选择合适的语言、工具对其进行构建。 微服务架构模式，核心是将复杂应用划分成小颗粒度、轻量化的自治服务，并围绕微服务开展服务的开发和服务的治理，这是实现云化软件的一种架构模式，也是近两年最热门的架构模式。

实践中，微服务架构模式中的微服务具有如下几个主要特点：

1、小

微服务架构通过对于特定的业务领域进行分析和业务建模，将复杂的业务逻辑剥离成为小而专一、耦合度低并且高度自治的一组服务，每个服务都是很小的应用。 虽然强调每个服务小，但是每个微服务本身还是完整的应用，这与我们通常说的组件、插件、共享库是有区别的。微服务的规模也没有严格的定义，通常一个微服务的开发团队在 6~8 人左右，一个微服务的架构重构可以在 2 周时间内完成这样的描述来确定一个大致的规模。这样的开发团队能够完成的开发规模就是一个微服务规模的上限。

2、独

独，指的是微服务的独立性。这里的独立性主要是针对一个微服务应用的交付过程而言的，也就是开发、测试以及部署升级的独立。在微服务架构中，每个服务都是一个独立的业务单元。这个业务单元在部署形态上，是独立的业务进程。对某个微服务进行改变，不会影响其它的服务。对于每个微服务，都有独立的代码库。某个微服务的代码修改，不会影响其它的微服务。对于每个微服务，都有独立的测试验证机制，不必担心破坏完整功能而开展大范围的回归测试（这往往是现有大集成全覆盖测试研发模式中消耗很大，但测试结果却不让人放心的地方）。

3、轻

微服务强调服务自治，因此服务之间的交互，必须采用消息通信的方式予以开展。从效率的角度，应当选择轻量级的通信机制。在软件实现的实践上，RESTful API 的方式被广泛采用。

这种通信机制的优点是语言无关、平台无关，并且十分便于制定通信协议，保证接口的前向兼容性。

在从传统软件架构向微服务化架构演进的过程中，业界实践部分也保留了 RPC 的通信机制，但要求基于 RPC 进行通信协议的制定，以保证接口的前向兼容性，实际是为了支撑服务间的独立和服务的松耦合态。

4、松

松是指微服务间松耦合，每个微服务可独立部署，互相之间没有部署先后顺序的依赖，微服务的接口前向兼容，单独微服务的上线，对于其他服务而言不产生关联，可以独立地灰度发布和灰度升级。

实现微服务间松耦合还有一点需要注意，就是一个微服务完成且最好仅完成一件事，业务逻辑的独立是微服务间解耦的关键。

6.3.4 无服务器

无服务器和 Functions-as-a-Service（FaaS）是云计算最新的热点趋势。以下关于无服务器和 FaaS 的介绍来自于费良宏（亚马逊 AWS 首席云计算技术顾问）的观点。

如同许多新的概念一样，Serverless 目前还没有一个普遍公认的权威的定义。最新的一个定义是这样描述的：“无服务器架构是基于互联网的系统，其中应用开发不使用常规的服务进程。相反，它们仅依赖于第三方服务（例如 AWS Lambda 服务），客户端逻辑和服务托管远程过程调用的组合。”

最开始，“无服务器”架构试图帮助开发者摆脱运行后端应用程序所需的服务器设备的设置和管理工作。这项技术的目标并不是为了实现真正意义上的“无服务器”，而是指由第三方云计算供应商负责后端基础结构的维护，以服务的方式为开发者提供所需功能，例如数据库、消息，以及身份验证等。简单地说，这个架构就是要让开发人员关注代码的运行而不需要管理任何的基础设施。程序代码被部署在诸如 AWS Lambda 这样的平台之上，通过事件驱动的方法去触发对

函数的调用。很明显，这是一种完全针对程序员的架构技术。其技术特点包括了事件驱动的调用方式，以及有一定限制的程序运行方式，例如 AWS Lambda 的函数的运行时间默认为 3 秒到 5 分钟。从这种架构技术出现的两年多时间来看，这个技术已经有了非常广泛的应用，例如移动应用的后端和物联网应用等。简而言之，无服务器架构的出现不是为了取代传统的应用。然而，从具有高度灵活性的使用模式及事件驱动的特点出发，开发人员/架构师应该重视这个新的计算范例，它可以帮助我们达到减少部署、提高扩展性并减少代码后面的基础设施的维护负担。

微服务（MicroService）是软件架构领域另一个热门的话题。如果说微服务是以专注于单一责任与功能的小型功能模块为基础，利用模块化的方式组合出复杂的大型应用程序，那么我们还可以进一步认为 Serverless 架构可以提供一种更加“代码碎片化”的软件架构模式，我们称之为 Function as a Services（FaaS）。而所谓的“函数”（Function）提供的是相比微服务更加细小的程序单元。例如，可以通过微服务代表为某个客户执行所有 CRUD 操作所需的代码，而 FaaS 中的“函数”可以代表客户所要执行的每个操作：创建、读取、更新，以及删除。当触发“创建账户”事件后，将通过 AWS Lambda 函数的方式执行相应的“函数”。从这一层意思来说，我们可以简单地将 Serverless 架构与 FaaS 概念等同起来。

7 结语

云计算以及和云计算相关的技术还有很多，比如网络、存储等，在这里我们重点介绍了虚拟化以及和虚拟化相关的知识。其他一些成熟的技术，比如容器和 OpenStack，我们会在最新版本的 HCIP 认证课程中重点介绍。另外的一些新兴的技术，比如雾计算、边缘技术和无服务器等，请大家收集相关的材料进行学习，华为官网也会不定期更新一些相关的文档供大家参考。

为了更好的和大家进行沟通，我们在华为企业互动社区设立了 HCIA-Cloud Computing 官方论坛区域，具体网址为：

https://forum.huawei.com/enterprise/zh/thread-513763-1-1.html

大家有任何的和云计算相关的建议、意见以及技术方面的问题都可以在这里留言。

此外，您还可以通过以下获取更多学习资源：

1、华为认证官网：https://support.huawei.com/learning/zh/newindex.html

2、华为 e 学云在线学习平台：https://ilearningx.huawei.com/portal/#/portal/EBG/26

3、华为 ICT 学院官网：https://www.huaweiacad.com/webapps/hw-toboo-BBLEARN/portal/exec.do?t=1561101910908

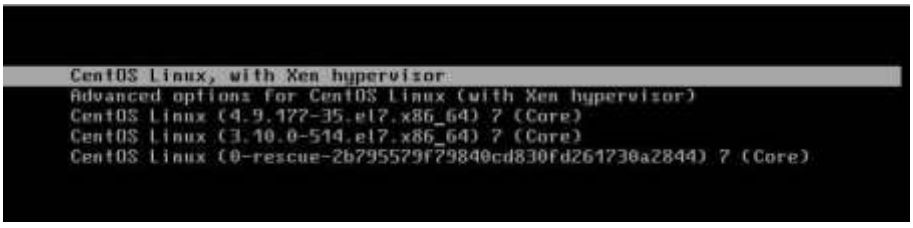
4、微信公众号：

- 华为认证
- 华为 HCIE 精英沙龙
- 华为 ICT 学院

8 附录

8.1 验证 1：

Xen 和 KVM 由于架构不一样，都需要运行在 Linux 内核，所以，如果一个 Linux 既安装了 KVM，也安装了 Xen 的时候，开机的时候需要选择使用哪个内核，因此在同一个 Linux 中，Xen 和 KVM 无法同时运行，开机时需要选择运行哪个内核，如下图。



根据上面的结论，如果不是使用 Xen 的 Linux 就可以使用 KVM，首先我们看看如果使用了 Xen 的情况，使用命令 lscpu 查看 Hypervisor vendor，具体如下（注意加粗部分）：

```
[root@localhost ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 79
Model name:            Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
Stepping:              1
CPU MHz:               2095.164
BogoMIPS:              4190.32
Hypervisor vendor:     Xen
Virtualization type:   none
L1d cache:             32K
```

```
L1i cache:           32K
L2 cache:            256K
L3 cache:            20480K
NUMA node0 CPU(s):    0-31
Flags:fpu de tsc msr pae mce cx8 apic sep mca cmov pat clflush acpi mmx fxsr
sse sse2 ht syscall nx lm constant_tsc arch_perfmon rep_good nopl nonstop_tsc
pni pclmulqdq est ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c
rdrand hypervisor lahf_lm abm 3dnowprefetch epb fsgsbase bmi1 hle avx2 bmi2
erms rtm rdseed adx xsaveopt cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local
dtherm arat pln pts
```

然后我们在 CNA 上使用命令同样的命令查看一下输出，具体如下：

```
CNA:~ # lscpu
Architecture:         x86_64
CPU op-mode(s):       32-bit, 64-bit
Byte Order:           Little Endian
CPU(s):              32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):           2
NUMA node(s):         2
Vendor ID:           GenuineIntel
CPU family:          6
Model:               63
Model name:          Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
Stepping:            2
CPU MHz:             2600.609
BogoMIPS:            5193.14
Virtualization:       VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            20480K
NUMA node0 CPU(s):    0-7,16-23
NUMA node1 CPU(s):    8-15,24-31
```

然后再检查一下是否加载了 KVM 模块，具体如下：

```
CNA:~ # lsmod | grep kvm
kvm_intel           176445  6
kvm                598042  1 kvm_intel
irqbypass           13503  5 kvm,vfio_pci
```

由此可证明，在 FusionCompute 6.3 中 CNA 上使用的 KVM 的虚拟化架构。

8.2 验证 2：

在 CNA 中使用命令 qemu-kvm -version，通过输出来验证是否使用的 qemu 模块，具体如下：

```
CNA:~ # qemu-kvm -version
2019-06-06T10:52:17.263814+08:00|info|qemu[15147]|[15147]|scan_dir[163]|: no hotpatch direcToRy, will not reload hotpatch
2019-06-06T10:52:17.264967+08:00|info|qemu[15147]|[15147]|main[3342]|: qemu pid is 15147, options parsing start!
QEMU emulaToR version 2.8.1.1(25.165)
Copyright (c) 2003-2016 Fabrice Bellard and the QEMU Project developers
```

通过该命令的输出，我们可以看到 QEMU 的版本为 2.8.1.1，证明了 CNA 中也使用了 QEMU 模块，并进一步证明 CNA 使用的 KVM 架构。

在 Linux 中，使用命令 systemctl 可以查看到正在使用的所有服务，因此，我们在 CNA 中使用此命令看一下没有相关的服务，具体如下：

```
CNA:~ # systemctl | grep libvirt
libvirt-guests.service
loaded active exited   Suspend/Resume Running libvirt Guests
libvirtd.service
loaded active running  Virtualization daemon
```

通过命令的输出，我们可以看到 libvirtd.service 正处在 running 状态，一次，可以证明在 CNA 中同样也是用到 libvirt。

8.3 验证 3：

登录任一台 CNA 主机，输入命令 ovs-vsctl show 后输出如下：

```
CNA:/etc/libvirt/qemu # ovs-vsctl show
248b2f99-7edb-4ac4-a654-c9054def32bb
    Bridge "br1"
        Port "tap00000002.0"
            tag: 0
            Interface "tap00000002.0"
                options: {brd_ratelimit="0"}
        Port "patch-1"
            Interface "patch-1"
                type: patch
```

```
            options: {peer="patch-bond13"}
        Port "br1"
            Interface "br1"
                type: internal
        Port "tap00000001.0"
            tag: 0
            Interface "tap00000001.0"
                options: {brd_ratelimit="0"}
    Bridge "br-bond13"
        Port "Mgnt-0"
            Interface "Mgnt-0"
                type: internal
        Port "patch-bond13"
            Interface "patch-bond13"
                type: patch
                options: {peer="patch-1"}
        Port "br-bond13"
            Interface "br-bond13"
                type: internal
        Port "bond13"
            Interface "bond13"
ovs_version: "2.7.3"
```

前面我们已经讲过，tap 设备和虚拟机网卡有关，黑体部分是我们需要关注的。可以看出，该网卡有 tag 和 options 两个属性，tag 是设置端口的 vlan 标签，options 是设置 QoS、类型等属性，如果没有端口组，我们需要一个一个端口的设置这些属性，如果有了端口组，我们只需要对端口组设置就可以了，然后在创建虚拟机时网卡指定端口组，该虚拟机网卡就具备了这些属性。

8.4 验证 4：

我们前面讲过，由于网桥不支持流量统计等功能，所以现在虚拟化厂家都采用虚拟交换机的方式，所以 VRM 要想统计端口的流量，首先 DVS 要有流量信息，我们使用命令 vs-ofctl dump-ports br1 来看一下 DVS 所捕获的信息，具体如下：

```
CNA:/etc/libvirt/qemu # ovs-ofctl dump-ports br1
OFPST_PORT reply (xid=0x2): 4 ports
  port  2: rx pkts=2681611, bytes=676430572, drop=0, errs=0, frame=0, over=0,
crc=0
           tx pkts=3750074, bytes=1031065371, drop=0, errs=0, coll=0
```

```
  port  1: rx pkts=4521055, bytes=1094061951, drop=0, errs=0, frame=0, over=0,
crc=0
          tx pkts=2681620, bytes=676432674, drop=0, errs=0, coll=0
  port  3: rx pkts=9, bytes=2102, drop=0, errs=0, frame=0, over=0, crc=0
          tx pkts=35, bytes=2202, drop=1053200, errs=0, coll=0
  port LOCAL: rx pkts=1437344, bytes=84784444, drop=0, errs=0, frame=0, over=0,
crc=0
          tx pkts=0, bytes=0, drop=0, errs=0, coll=0
```

然后我们对比 VRM 上所体现出来的流量信息，如下图：

网卡	所属端口组	所属分布式交换机 ...	接收包数(packet) ...	接收字节数(KB)	发送包数(packet) ...	发送字节数(KB)
Network Adapter 0	managePortgroup	ManagementDVS	3751825	1007390.72	2682880	660930.56

我们可以发现这个网卡的报吞吐数等数量个 port2 的信息差不多，因此，我们可以得出结论：

VRM 是通过采集了 DVS 统计的端口流量来收集到每个端口的状况的。