66 | 知识串讲：用一个创业故事串起操作系统原理（五）

2019-08-28 刘超

讲述：刘超
时长 17:43 大小 16.24M

上一节我们说到，马哥的公司现在接个千万级别的项目没有任何问题，但是投资人说，要想冲一把上市，还差点劲，目前的项目虽然大，但是想象力不够丰富。

亿级项目创品牌，战略合作遵协议

马哥突然想到，西部有一个智慧城市的打单，金额几个亿，绝对标杆性质的。如果能够参与其中，应该是很有想象力的事情。

可是，甲方明确地说，“整个智慧城市的建设体系非常的大，一家公司做不下来，需要多家公司合作才能完成。你们有多家公司合作的经验和机制吗？”

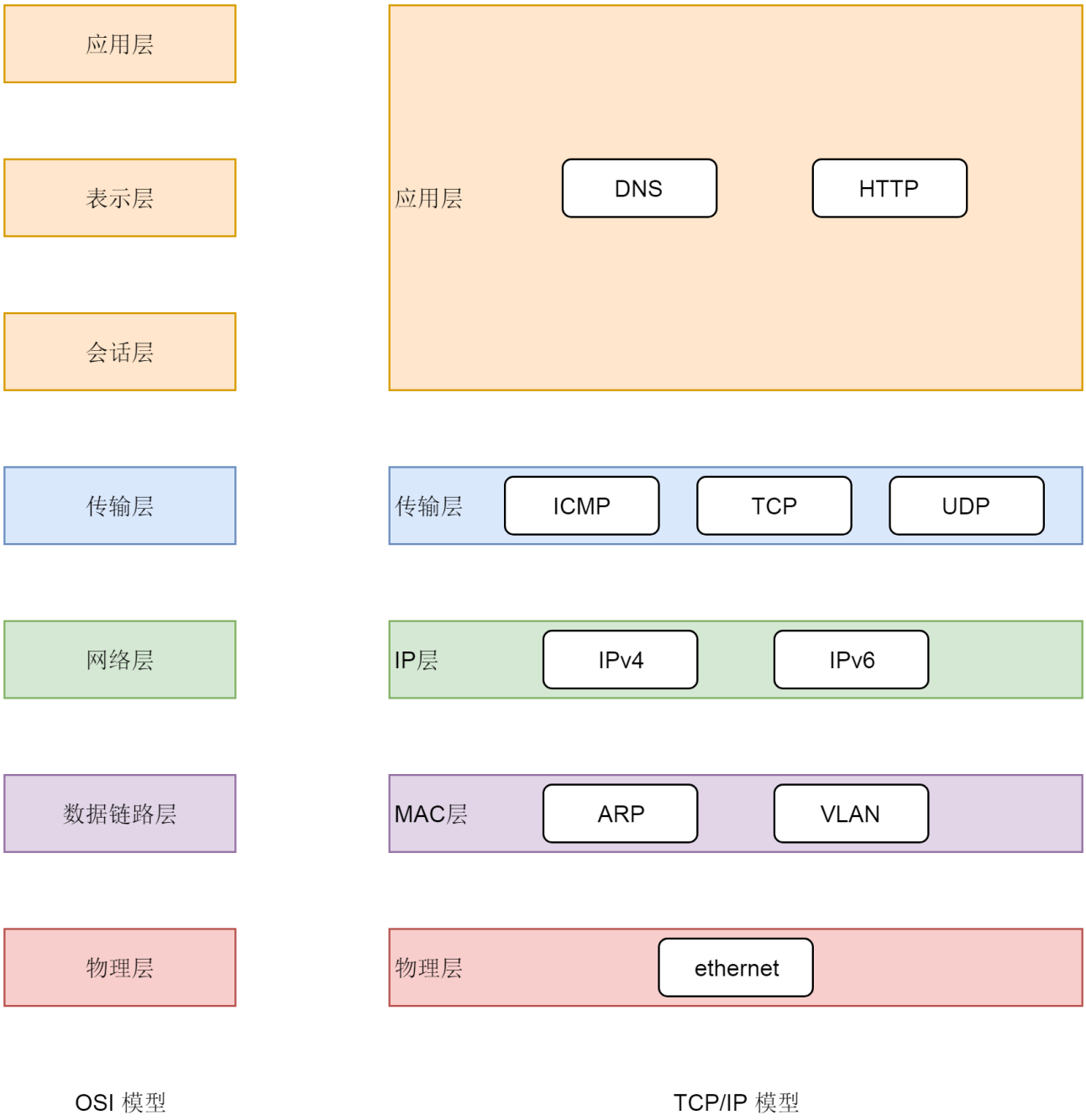
马哥咬牙说道：“当然有！”先应下来再说呗，可是这心里是真没底。原来公司都是独自接单，现在要和其他公司合作，协议怎么签，价格怎么谈呢？

马哥找到鲁肃。鲁肃说：“我给你推荐一个人吧！这个人人脉广，项目运作能力强，叫陆逊，说不定能帮上忙。”

鲁肃找来陆逊。陆逊说：“这个好办。公司间合作嘛，就是条款谈好，利益分好就行，关键是大家要遵守行规。大家都按统一的规则来，事情就好办。”

这其实就像机器与机器之间合作，一台机器将自己想要表达的内容，按照某种约定好的格式发送出去。当另外一台机器收到这些信息后，也能够按照约定好的格式解析出来，从而准确、可靠地获得发送方想要表达的内容。这种约定好的格式就是网络协议。

现在业内知名的有两种网络协议模型，一种是 OSI 的标准七层模型，一种是业界标准的 TCP/IP 模型。它们的对应关系如下图所示：



我们先从第三层网络层开始，因为这一层有我们熟悉的 IP 地址，所以这一层我们也叫 IP 层。

连接到网络上的每一个设备都至少有一个 IP 地址，用于定位这个设备。无论是近在咫尺的、你旁边同学的电脑，还是远在天边的电商网站，都可以通过 IP 地址进行定位。因此，IP 地址类似互联网上的邮寄地址，是有全局定位功能的。

就算你要访问美国的一个地址，也可以从你身边的网络出发，通过不断地打听道儿，经过多个网络，最终到达目的地址，和快递员送包裹的过程差不多。打听道儿的协议也在第三层，我们称为路由协议。将网络包从一个网络转发给另一个网络的设备，我们称为路由器。

总而言之，第三层干的事情，就是网络包从一个起始的 IP 地址，沿着路由协议指的道儿，经过多个网络，通过多次路由器转发，到达目标 IP 地址。

从第三层，我们往下看。第二层是数据链路层。有时候我们简称为二层或者 MAC 层。所谓 MAC，就是每个网卡都有的唯一的硬件地址（不绝对唯一，相对大概率唯一即可，类比 UUID）。这虽然也是一个地址，但是这个地址是没有全局定位功能的。

就像给你送外卖的小哥，不可能根据手机尾号找到你家，但是手机尾号有本地定位功能的，只不过这个定位主要靠“吼“。外卖小哥到了你的楼层就开始大喊：“尾号 xxxx 的，你外卖到了！“

MAC 地址的定位功能局限在一个网络里面，也即同一个网络号下的 IP 地址之间，可以通过 MAC 进行定位和通信。从 IP 地址获取 MAC 地址要通过 ARP 协议，是通过在本地发送广播包，也就是“吼“，获得的 MAC 地址。

由于同一个网络内的机器数量有限，通过 MAC 地址的好处就是简单。匹配上 MAC 地址就接收，匹配不上就不接收，没有什么所谓路由协议这样复杂的协议。当然坏处就是，MAC 地址的作用范围不能出本地网络，所以一旦跨网络通信，虽然 IP 地址保持不变，但是 MAC 地址每经过一个路由器就要换一次。

所以第二层干的事情，就是网络包在本地网络中的服务器之间定位及通信的机制。

我们再往下看第一层，物理层。这一层就是物理设备。例如，连着电脑的网线，我们能连上的 WiFi。

从第三层往上看，第四层是传输层，这里面有两个著名的协议，TCP 和 UDP。尤其是 TCP，更是广泛使用，在 IP 层的代码逻辑中，仅仅负责数据从一个 IP 地址发送给另一个 IP 地址，丢包、乱序、重传、拥塞，这些 IP 层都不管。处理这些问题的代码逻辑写在了传输层的 TCP 协议里面。

我们常说，TCP 是可靠传输协议，也是难为它了。因为从第一层到第三层都不可靠，网络包说丢就丢，是 TCP 这一层通过各种编号、重传等机制，让本来不可靠的网络对于更上层来讲，变得“看起来”可靠。哪有什么应用层的岁月静好，只不过是 TCP 层在负重前行。

传输层再往上就是应用层，例如，咱们在浏览器里面输入的 HTTP，Java 服务端写的 Servlet，都是这一层的。

二层到四层都是在 Linux 内核里面处理的，应用层例如浏览器、Nginx、Tomcat 都是用户态的。内核里面对于网络包的处理是不区分应用的。

从四层再往上，就需要区分网络包发给哪个应用。在传输层的 TCP 和 UDP 协议里面，都有端口的概念，不同的应用监听不同的端口。例如，服务端 Nginx 监听 80、Tomcat 监听 8080；再如客户端浏览器监听一个随机端口，FTP 客户端监听另外一个随机端口。

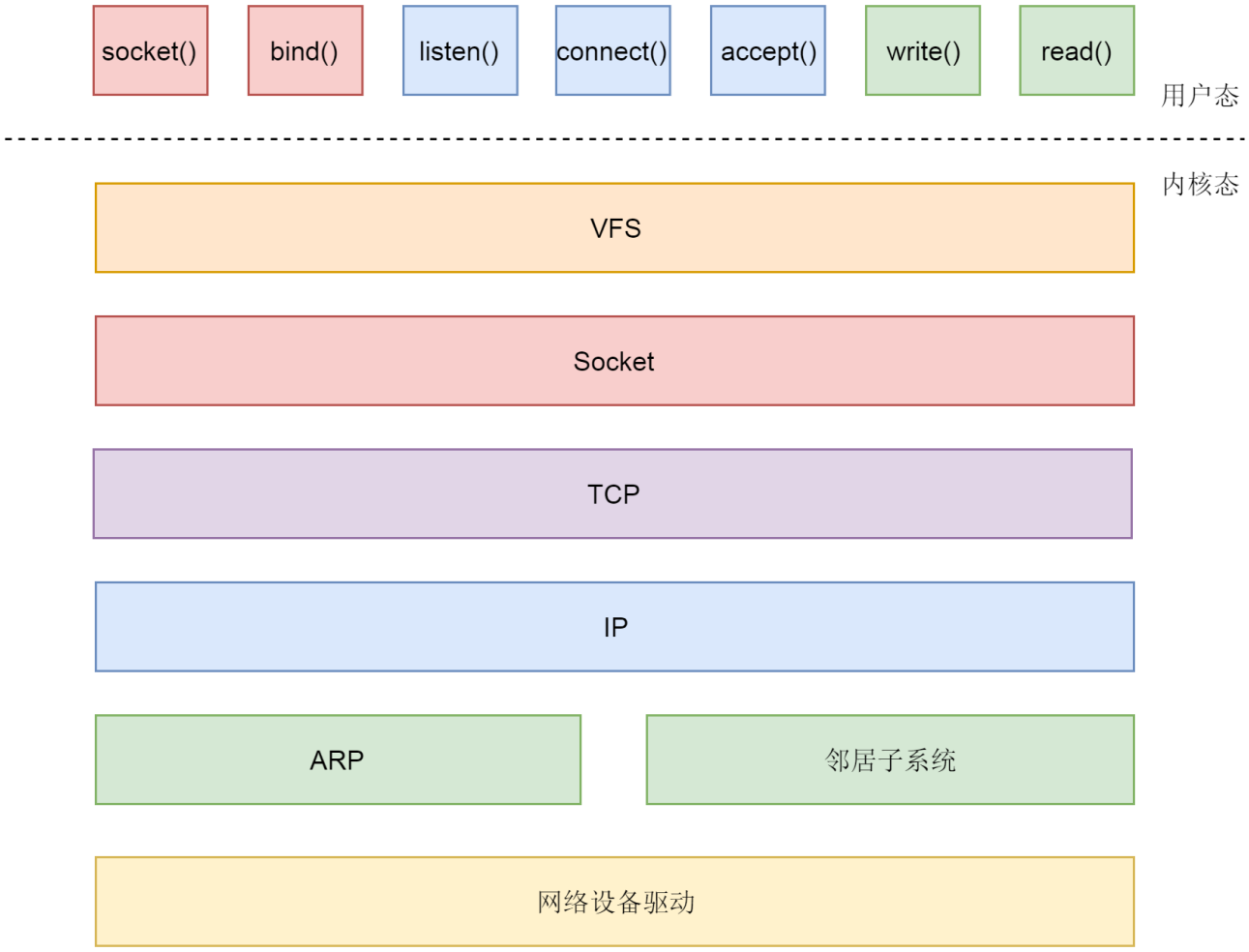
应用层和内核互通的机制，就是通过 Socket 系统调用。所以经常有人会问，Socket 属于哪一层，其实它哪一层都不属于，它属于操作系统的概念，而非网络协议分层的概念。

操作系统对于网络协议的实现模式是这样的：二到四层的处理代码在内核里面，七层的处理代码让应用自己去做。两者需要跨内核态和用户态通信，就需要一个系统调用完成这个衔接，这就是 Socket。

如果公司想要和其他公司沟通，我们将请求封装为 HTTP 协议，通过 Socket 发送到内核。内核的网络协议栈里面，在 TCP 层创建用于维护连接、序列号、重传、拥塞控制的数据结构，将 HTTP 包加上 TCP 头，发送给 IP 层，IP 层加上 IP 头，发送给 MAC 层，MAC 层加上 MAC 头，从硬件网卡发出去。

最终网络包会被转发到目标服务器，它发现 MAC 地址匹配，就将 MAC 头取下来，交给上一层。IP 层发现 IP 地址匹配，将 IP 头取下来，交给上一层。TCP 层会根据 TCP 头中的序列号等信息，发现它是一个正确的网络包，就会将网络包缓存起来，等待应用层的读取。

应用层通过 Socket 监听某个端口，因而读取的时候，内核会根据 TCP 头中的端口号，将网络包发给相应的应用。



这样一个大项目中，各个公司都按协议来，别说两家公司合作，二十家也没有问题。

于是陆逊带着马哥，到甲方那里，将自己的方案，以及和其他公司的合作模式讲述清楚。马哥成功入围。

这次参与竞标的公司可不少，马哥公司的竞争力和专业性一点都不差，最后终于拿下了智慧生态合作平台的建设部分。这下不得了，一提马哥的公司，业内无人不知，无人不晓，大家纷纷称呼他为“马总”。

公司大了不灵活，鼓励创新有妙招

慢慢地，马总发现，公司大有大的好处，自然也有大的毛病，也就是咱们常见的“大公司病”——不灵活。

这里面的不灵活，就像 Linux 服务器，越来越强大的时候，无论是计算、网络、存储，都越来越牛。例如，内存动不动就是百 G 内存，网络设备一个端口的带宽就能有几十 G 甚至上百 G。存储在数据中心至少是 PB 级别的，自然也有不灵活的毛病。

资源大小不灵活：有时候我们不需要这么大规格的机器，可能只想尝试一下某些新业务，申请个 4 核 8G 的服务器试一下，但是不可能采购这么小规格的机器。无论每个项目需要多大规格的机器，公司统一采购就限制几种，全部是上面那种大规格的。

资源申请不灵活：规格定死就定死吧，可是每次申请机器都要重新采购，周期很长。

资源复用不灵活：反正我需要的资源不多，和别人共享一台机器吧，这样不同的进程可能会产生冲突，例如 socket 的端口冲突。另外就是别人用过的机器，不知道上面做过哪些操作，有很多的历史包袱，如果重新安装则代价太大。

按说，大事情流程严禁没问题，很多小事情也要被拖累走整个流程，而且很容易出现资源冲突，每天跨部门的协调很累人，历史包袱严重，创新没有办法轻装上阵。

很多公司处理这种问题采取的策略是成立独立的子公司，独立决策，独立运营。这种办法往往会用在创新型的项目上。

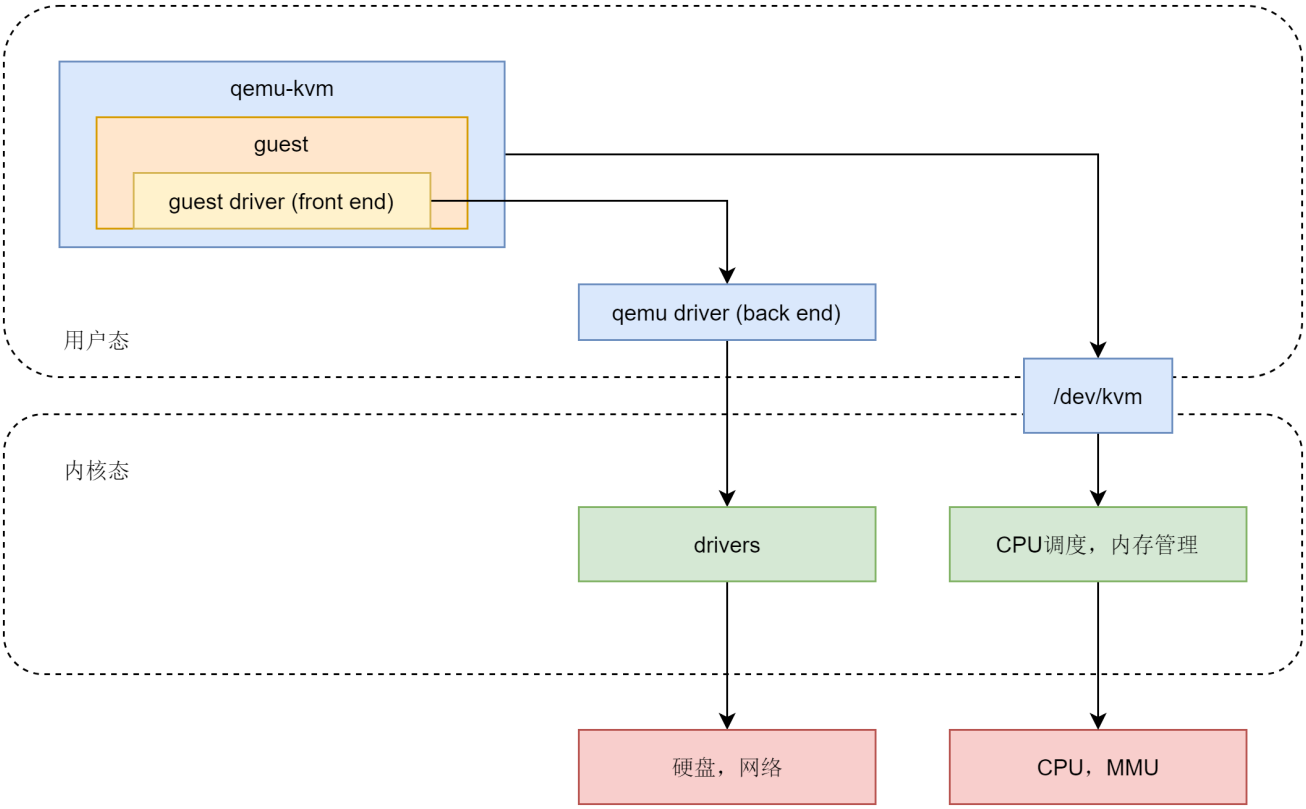
Linux 也采取了这样的手段，就是在物理机上面创建虚拟机。每个虚拟机有自己单独的操作系统、灵活的规格，一个命令就能启动起来。每次创建都是新的操作系统，很好地解决了上面不灵活的问题。

在物理机上的操作系统看来，虚拟机是一个普通的应用，他和 Excel 一样，只能运行在用户态。但是对于虚拟机里面的操作系统内核来讲，运行在内核态，应该有高的权限。

要做到这件事情，第一种方式，完全虚拟化。其实说白了，这是一种“骗人”的方式。虚拟化软件会模拟假的 CPU、内存、网络、硬盘给到虚拟机，让虚拟机里面的内核自我感觉良好，感觉他终于又像个内核了。在 Linux 上，一个叫作 qemu 的工具可以做到这一点。

qemu 向虚拟机里面的客户机操作系统模拟 CPU 和其他的硬件，骗客户机，GuestOS 认为自己和硬件直接打交道，其实是同 qemu 模拟出来的硬件打交道，qemu 会将这些指令转译给真正的硬件。由于所有的指令都要从 qemu 里面过一手，因而性能就会比较差。

第二种方式，硬件辅助虚拟化。可以使用硬件 CPU 的 Intel-VT 和 AMD-V 技术，需要 CPU 硬件开启这个标志位（一般在 BIOS 里面设置）。当确认开始了标志位之后，通过内核模块 KVM，GuestOS 的 CPU 指令将不用经过 Qemu 转译，直接运行，大大提高了速度。qemu 和 KVM 融合以后，就是 qemu-kvm。



第三种方式称为半虚拟化。对于网络或者硬盘的访问，我们让虚拟机内核加载特殊的驱动，重新定位自己的身份。虚拟机操作系统的内核知道自己不是物理机内核，没那么高的权限。他很可能要和很多虚拟机共享物理资源，所以学会了排队。虚拟机写硬盘其实写的是一个物理机上的文件，那我的写文件的缓存方式是不是可以变一下。我发送网络包，根本就不是发给真正的网络设备，而是给虚拟的设备，我可不可以直接在内存里面拷贝给它，等等等等。

网络半虚拟化方式是 virtio_net，存储是 virtio_blk。客户机需要安装这些半虚拟化驱动。客户机内核知道自己是虚拟机，所以会直接把数据发送给半虚拟化设备，然后经过特殊处理（例如排队、缓存、批量处理等性能优化方式），最终发送给真正的硬件。这在一定程度上提高了性能。

有了虚拟化的技术，公司的状态改观了不少，在主要的经营方向之外，公司还推出了很多新的创新方向，都是通过虚拟机创建子公司的方式进行的，例如跨境电商、工业互联网、社交等。一方面，能够享受大公司的支持；一方面，也可以和灵活的创业公司进行竞争。

于是，公司就变成集团公司了。

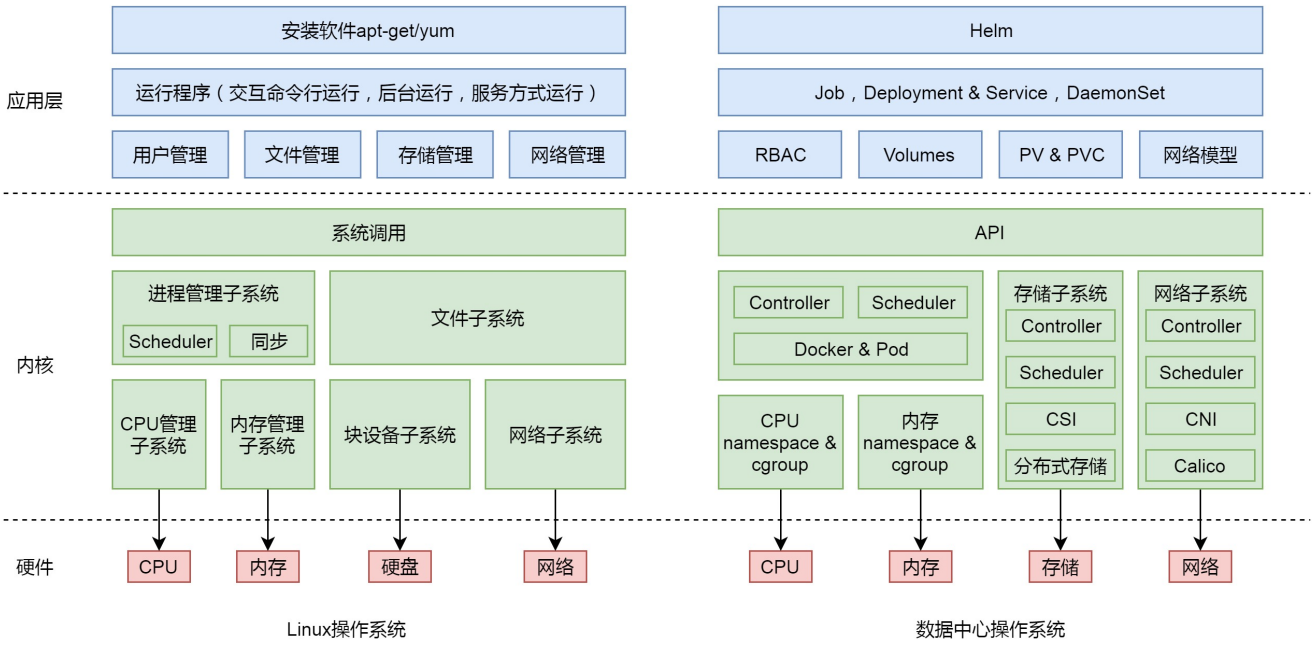
独占鳌头定格局，上市敲钟责任重

随着公司越来越大，钱赚的越来越多，马总的公司慢慢从行业的追随者，变成了领导者。这一方面，让马总觉得“会当凌绝顶，一览众山小”；另一方面，马总也觉得“高处不胜寒”。原来公司总是追着别人跑，产业格局，市场格局从来不用自己操心，只要自己的公司能赚钱就行。现在做了领头羊，马总也就慢慢成了各种政府论坛、产业论坛，甚至国际论坛的座上宾。

穷则独善其身，达则兼济天下。马总的决策可能关系到产业的发展、地方的 GDP 和就业，甚至未来的国际竞争力。因此，即便是和原来相同的事情，现在来做，方式和层次都不一样了。

就像对于单台 Linux 服务器，最重要的四种硬件资源是 CPU、内存、存储和网络。面对整个数据中心成千上万台机器，我们只要重点关注这四种硬件资源就可以了。如果运维数据中心依然像的运维一台台物理机的前辈一样，天天关心哪个程序放在了哪台机器上，使用多少内存、多少硬盘，每台机器总共有多少内存、多少硬盘，还剩多少内存和硬盘，那头就大了。

对于数据中心，我们需要一个调度器，将运维人员从指定物理机或者虚拟机的痛苦中解放出来，实现对于物理资源的统一管理，这就是 Kubernetes，也就是数据中心的操作系统。



对于 CPU 和内存这两种计算资源的管理，我们可以通过 Docker 技术完成。

容器实现封闭的环境主要要靠两种技术，一种是看起来是隔离的技术，称为 namespace。在每个 namespace 中的应用看到的，都是不同的 IP 地址、用户空间、进程 ID 等。另一种是用起来是隔离的技术，称为 cgroup，即明明整台机器有很多的 CPU、内存，但是一个应用只能用其中的一部分。

另外，容器里还有镜像。也就是说，在你焊好集装箱的那一刻，将集装箱的状态保存下来的样子。就像孙悟空说“定！”，集装箱里的状态就被“定”在了那一刻。然后，这一刻的状态会被保存成一系列文件。无论在哪里运行这个镜像，都能完整地还原当时的情况。

通过容器，我们可以将 CPU 和内存资源，从大的资源池里面隔离出来，并通过镜像技术，在数据中心里面实现计算资源的自由漂移。

没有操作系统的时候，汇编程序员需要指定程序运行的 CPU 和内存物理地址。同理，数据中心的管理员，原来也需要指定程序运行的服务器以及使用的 CPU 和内存。现在，Kubernetes 里面有一个调度器 Scheduler，你只需要告诉它，你想运行 10 个 4 核 8G 的 Java 程序，它会自动帮你选择空闲的、有足够资源的服务器，去运行这些程序。

对于存储，无论是分布式文件系统和分布式块存储，需要对接到 Kubernetes，让 Kubernetes 管理它们。如何对接呢？Kubernetes 会提供 CSI 接口。这是一个标准接口，不同的存储可以实现这个接口来对接 Kubernetes。是不是特别像设备驱动程序呀？操作系统只要定义统一的接口，不同的存储设备的驱动实现这些接口，就能被操作系统使用了。

对于网络，也是类似的机制，Kubernetes 同样是提供统一的接口 CNI。无论你用哪种方式实现网络模型，只要对接这个统一的接口，Kubernetes 就可以管理容器的网络。

到此，有了这套鼎定市场格局的策略，作为龙头企业，马总的公司终于可以顺利上市敲钟，走向人生巅峰。从此，江湖人称“马爸爸”。

好了，马同学的创业故事就讲到这里了，操作系统的原理也给你串了一遍。你是否真的记住了这些原理呢？试着将这个创业故事讲给你的朋友听吧！

上一篇　65 | 知识串讲：用一个创业故事串起操作系统原理（四）

下一篇　67 | 期末测试：这些操作系统问题，你真的掌握了吗?

精选留言 (7)

写留言

EidLeung
2019-08-28

再来一个专栏啊😊

展开

作者回复: 这是要累死人不偿命

2

罗乾林
2019-08-28

专栏里好像没对多核cpu如何管理做过多介绍。老师，有问题请教下?
多核CPU是不是每一个核都有一套对应的寄存器，如4核cpu是否PC寄存器也有4个，每个

核独立的取指，译码，执行都有一整套，发生中断后这些核会不会都跑去执行中断处理程序发生争抢

展开

1

许童童

2019-08-28

老师讲得真好，我现在就去把这个故事讲给同事听，估计要讲一个小时，希望同事不会被烦死。

Cyril

2019-08-28

老师能否详细写一点关于 smp 相关的知识，比如多 cpu 如何处理网卡过来的中断，多 cpu 如何进程调度，多 cpu 又是如何解决共享变量访问冲突的问题，对这一部分知识点一直比较模糊

安排

2019-08-28

同期待老师的其它专栏🙏

展开

leslie

2019-08-28

老师的这个串讲做的好啊-精辟；整个把知识体系梳理了一遍；老师这个串讲不是学习2-3遍的事情，是N遍的事情；这书能读的N厚，不过当把这书彻底读到薄的时候估计很多课程都串联起来了。

这就是老师讲的学习方法中的读厚，然后再读薄的修炼啊。

这堂课把两块梳理清楚了：一个是socket,一个是Docker:...

展开

高大强

2019-08-28

我们购买的阿里云服务器，是否也是虚拟的?
良心之作，学不好是自己的事，从头再来。

作者回复: 是虚拟的

2