Journal of Computer Applications
计算机应用,2017,37(4):915 -923
ISSN 1001-9081
CODEN JYIIDU
2017-04-10
http://www.joca.cn

文章编号:1001-9081(2017)04-0915-09
DOI:10.11772/j.issn.1001-9081.2017.04.0915

云计算虚拟化技术的发展与趋势

武志学[1,2*]

(1.成都五舟汉云科技有限公司,成都 611731; 2.成都信息工程大学 信息安全工程学院,成都 610225)

(* 通信作者电子邮箱 zhixue.wu@gmail.com)

摘 要:云计算是一种融合了多项计算机技术的以数据和处理能力为中心的密集型计算模式,其中以虚拟化、分布式数据存储、分布式并发编程模型、大规模数据管理和分布式资源管理技术最为关键。经过十多年的发展,云计算技术已经从发展培育期步入快速成长期,越来越多的企业已经开始使用云计算服务。与此同时,云计算的核心技术也在发生着巨大的变化,新一代的技术正在改进甚至取代前一代技术。容器虚拟化技术以其轻便、灵活和快速部署等特性对传统的基于虚拟机的虚拟化技术带来了颠覆性的挑战,正在改变着基础设施即服务(IaaS)平台和平台即服务(PaaS)平台的架构和实现。对容器虚拟化技术进行深入介绍,并通过分析和比较阐述容器虚拟化技术和虚拟机虚拟化技术各自的优势、适应场景和亟待解决的问题,然后对云计算虚拟化技术的下一步研究方向和发展趋势进行展望。

关键词:云计算;虚拟化;Docker;容器;虚拟机

中图分类号:TP391 **文献标志码**:A

Advances on virtualization technology of cloud computing

WU Zhixue[1,2*]

(1. *Chengdu Wuzhou Handge Technology Limited*, *Chengdu Sichuan* 611731, *China*;
2. *School of Information Security Engineering*, *Chengdu University of Information Technology*, *Chengdu Sichuan* 610225, *China*)

Abstract: Cloud computing is a new computing model focused on the capability of data and its processing. It integrates a number of information and communication technologies, including virtualization, distributed data storage, distributed parallel programming model, big data management and distributed resource management. After more than a decade of development, cloud computing has entered a rapid growth period, more and more enterprises have adapted to cloud computing services. At the same time, the key technologies of cloud computing have advanced as well. The new generation technologies are enhancing and even replacing the existing technologies. Container is a new type virtualization technology. With its lightweight, elastic and fast advantages, container challenges the traditional virtual machine technology, and brings changes to the architecture and implementation of both the Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). Container virtualization technology was described in detail, the advantages and disadvantages, the suitable use cases of container and virtual machine technology were compared and analyzed, then the future research directions and development trends of cloud computing virtualization technology were prospected.

Key words: cloud computing; virtualization; Docker; container; virtual machine

0 引言

云计算是一种融合了多项计算机技术的以数据和处理能力为中心的密集型计算模式。它的发展是虚拟化、分布式系统、分布式并发编程模式、面向对象的体系架构、软件即服务和信息安全等各项技术共同发展的结果;同时,托管服务、后向收费、按需交付等商业模式的出现也加速了云计算市场的突飞猛进。

经过十多年的发展,云计算技术已经趋于成熟,云计算平台产品也得到了企业的认同和广泛使用。与此同时,云计算的核心技术的发展并没有停止。特别是在近两年里,基于容器技术的 Docker 系统[1]、基于纠删码技术[2]的分布式存储系统和基于内存计算的 Spark 系统[3]的出现颠覆了原有的虚拟化技术、基于多副本的分布式数据存储技术,以及基于 MapReduce 的分布式并发编程模型。这些新技术不仅进一步提高了资源利用率,提高了云平台的计算速度,并且还为企业提供了更多的大数据应用模型,包含批处理、实时数据处理、流式数据处理、随机数据查询和数据挖掘等。

容器虚拟化技术以其轻便、灵活和快速部署等特性给传统的基于虚拟机的虚拟化技术带来了颠覆性的挑战。容器技术正在改变着基础设施即服务(Infrastructure as a Service, IaaS)平台和平台即服务(Platform as a Service, PaaS)平台的架构和实现。本文首先对容器虚拟化技术进行深入介绍;然后对容器虚拟化技术和虚拟机虚拟化技术进行分析和比较,阐述各自的优势、适应场景和亟待解决的问题;最后对云计算虚拟化技术的下一步研究方向和发展趋势进行展望。

收稿日期:2016-10-18;修回日期:2017-01-06。

作者简介:武志学(1960—),男,山西河津人,教授,博士,主要研究方向:云计算、流式数据处理、数据挖掘。

1 虚拟化技术

云计算模式最关键的突破就是资源使用方式的改变[4]。通过虚拟化的方式，可以在几分钟之内，虚拟出一个独立的、随需配置的虚拟机供用户使用。虚拟化技术给资源使用和调度带来了极大的方便，系统可以根据应用的实际负载情况及时进行资源调度，从而可以保证既不会因为资源得不到充分利用造成系统资源的浪费，又能够保证应用和服务不会因为资源缺乏而带来性能的下降。

虚拟化技术是指计算元件在虚拟的基础上而不是真实的基础上运行，通过软件的方法重新定义划分信息技术(Information Technology，IT)资源，实现IT资源的动态分配、灵活调度和跨域共享，从而提高IT资源的利用率，使IT资源真正成为计算基础设施，可以满足各种应用的灵活多变的需求。

受益于虚拟化技术的发展，计算机整体资源的使用效率和用户工作的时间价值都得到了巨大的提升，同时也相应减少了交付服务所做的重复性工作。通过虚拟化技术，云计算把计算、存储、应用和服务都变成了可以动态配置和扩展的资源，从而才能够实现在逻辑上以单一整体的服务形式呈现给用户。所以，虚拟化技术是云计算中最关键、最核心的技术原动力[5]。

1.1 服务器虚拟化

服务器虚拟化是指通过虚拟化技术将一台计算机虚拟为多台逻辑计算机。服务器的虚拟化是通过在硬件和操作系统之间引入虚拟化层实现硬件与操作系统的解耦而实现的，如图1所示。虚拟化层的主要功能就是实现在一台物理服务器上同时运行多个操作系统实例。通过动态分区，虚拟化层使这些操作系统实例可以共享物理服务器资源，使每个虚拟机得到一套独立的模拟出的硬件设备，包含CPU、内存、存储、主板、显卡、网卡等硬件资源。然后，再在其上安装自己的操作系统，称为客户(Guest)操作系统。最终用户的应用程序，运行在Guest操作系统中。

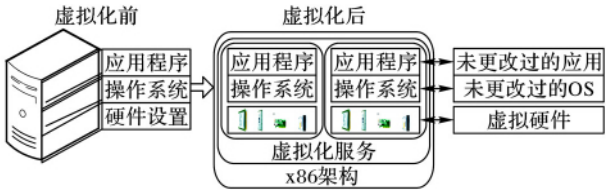
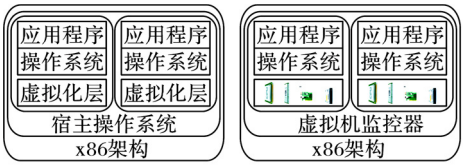


图1 服务器虚拟化

Fig. 1 Server virtualization

服务器虚拟化有两种常见的架构：寄居架构(Hosted Architecture)和裸金属架构("Bare Metal" Architecture)。寄居架构将虚拟化层运行在操作系统之上，当作一个应用来运行[6]。寄居架构依赖于主机操作系统对设备的支持和物理资源的管理，如图2(a)所示；裸金属架构直接将虚拟化层运行在x86的硬件系统上，再在其上安装操作系统和应用，如图2(b)所示。因为裸金属架构可以直接访问硬件资源，而不需要通过操作系统来实现对硬件访问，因此具有更高的效率。VMware Server是寄居架构虚拟化产品的代表；而Xen[7]、XenServer[8]、VMware ESX Server[8]和KVM[9]都是基于裸金属架构的虚拟化产品。



(a) 寄居架构虚拟化 (b) 裸金属架构虚拟化

图2 服务器虚拟化架构

Fig. 2 Server virtualization architecture

1.2 Docker容器技术

通过解除操作系统与物理主机之间的紧耦合，虚拟机虚拟化技术使操作系统的部署更为轻松便捷，工作负载的移动性显著增强。通过虚拟化的方式，可以很快虚拟出一个小的、独立的、随需随用CPU内核供用户使用。但是，当用户仅仅需要使用一小部分资源去运行一个很简单的应用时，虚拟出一整台计算机来完成软件发布不但会浪费相当的系统资源，并且启动虚拟机运行也需要几分钟的时间。因此，需要一种比虚拟机更小的资源分配粒度来满足这类需求。

为了能够比虚拟机模式以更快、更少资源的方式发布软件，就需要对资源进行比虚拟机模式更高级别的抽象，使得服务可以通过更细的粒度对资源进行分配和控制。为此，Linux内核添加了新的技术，这便是众所周知的控制组(Control Groups，cgroups)[10]。通过这一技术来对服务运行时环境进行隔离，这种被隔离起来的运行时环境就被称为容器[11]。

通过容器可以为应用程序提供一个隔离的运行空间，包括完整用户环境空间；一个容器内的变动不会影响其他容器的运行环境。所以，可以使用容器虚拟化技术将应用组件以及依赖打包为一个标准、独立、轻量的环境，来部署分布式应用，从而满足上述需要比虚拟机更小粒度来控制资源的需求。

容器技术使用了一系列的系统级别的机制，包括利用Linux namespaces来进行空间隔离，通过文件系统的挂载点来决定容器可以访问文件的权限，通过cgroups来控制每个容器可以利用多少资源。此外，多个容器之间可以共享同一个操作系统的内核，这样当同一个系统库被多个容器使用时，内存的使用效率会得到很大的提升。

Docker是一个可以简化和标准化不同环境中应用部署的容器平台，目前已经有很多的分布式容器管理相关的生态圈软件。近几年以来，随着Docker的出现，容器技术对云计算发展产生了巨大的影响。

1.2.1 Docker

Docker诞生于2013年，最初是dotCloud公司内部的一个业余项目[12]。项目后来加入Linux基金会以后成为了一个遵从Apache 2.0协议的开源项目。Docker自开源后受到广泛的关注和讨论，dotCloud公司已经改名为Docker Inc。Redhat已经在其RHEL6.5中集中支持Docker；Google也在其PaaS产品中广泛使用了Docker技术。

Docker项目的目标是实现轻量级的操作系统虚拟化解决方案，基于Google公司推出的Go语言实现。Docker以Linux容器(LXC)技术为基础，它的主要功能是通过实现对LXC的进一步封装，使得对容器的操作变得更为简便，并且让用户不

再需要关心容器的管理[13]。用户使用 Docker 平台上的容器就像操作一个轻量级的虚拟机一样简单。正是因为使得操作 LXC 变得简单和方便使用,Docker 带来了容器虚拟化进入云计算产品的热潮。

为了让 Docker 朝着容器封装、运行的标准化更进一步,2016 年发布的 Docker 1.10 推出了自己的 libcontainer,它集成了 Linux 内核中的很多特性,作为一个独特、稳定且不受制于 Linux 的 library[14]。

与提供硬件虚拟化机制的虚拟机不同,Docker 通过命名空间(namespaces)和控制组(cgroups)两个核心技术提供操作系统层级的虚拟化机制。Docker 利用 namespaces 进行权限的隔离控制;利用 cgroups 进行资源的限制隔离。

Docker 利用 namespaces 技术[13]来提供隔离的工作空间,称之为容器(Container)。当运行一个容器时,Docker 为该容器创建了一个命名空间集合。这样就给容器提供了一个隔离层,每一个应用在它们自己的命名空间中运行而且不会访问到命名空间之外。

Docker 利用 cgroups 技术对共享资源进行隔离、限制、审计等,确保 Docker 容器只使用其必需的资源,并在必要情况下设置其所能使用的资源上限。另外,cgroups 还能够确保单一容器不至于占用太多资源而导致整体系统陷入瘫痪。

图 3 显示了容器虚拟化的整体架构,可以看到容器和宿主机的关系,容器隔离独立的应用并使用已经被 Docker 抽象化的操作系统资源。在图 3 的右侧视图中,可以看到容器是用层(layer)来建立的,多个容器共享基础层以减少资源的使用。

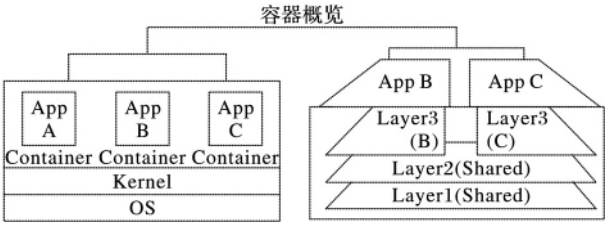


图 3　容器虚拟化架构

Fig. 3　Container virtualization architecture

1.2.2　Docker 的工作原理

Docker 的实现机制包含大量活动组件,其底层的核心技术包括 Linux 上的命名空间(Namespaces)、控制组(Control Groups)、Union 文件系统(Union File Systems)和容器格式(Container Format)。

传统的虚拟机是通过在宿主主机中运行 Hypervisor 来模拟一整套完整的硬件环境提供给虚拟机的操作系统使用。虚拟机系统看到的环境是可限制的,也是彼此隔离的。这种实现方法可以对资源进行最完整的封装,但同时又意味着对系统资源的浪费。例如,在宿主机和虚拟机系统都为 Linux 系统的情况下,虚拟机中运行的应用其实完全可以利用宿主机系统中的运行环境。

操作系统包括内核、文件系统、网络、PID(Process ID)、UID(User Identification)、IPC(Inter Process Communication)、内存、硬盘、CPU 等资源,所有这些资源都是应用进程直接共享的。因此,要实现虚拟化,首先需要实现对内存、CPU、网络 IO、硬盘 IO、存储空间等的限制,此外还需要实现对文件系统、网络、PID、UID、IPC 等之间的相互隔离。实现前者相对来讲要容易一些,而实现后者则需要宿主机系统的深入支持。

近年来,随着 Linux 系统对名字空间功能实现的不断完善,基本上已经可以实现上面的所有需求,进程可以在彼此隔离的命名空间中运行。尽管多个进程都在共享同一个内核和部分运行时环境(例如一些系统命令和系统库),但是进程相互之间是不可见的,都感觉系统中只有自己单独存在。这种机制就是容器(Container)技术。

1) 命名空间。

命名空间是 Linux 内核一个强大的特性。Docker 使用命名空间来实现容器之间的隔离。每个容器都有自己单独的命名空间,运行在其中的应用都像是在独立的操作系统中运行一样。命名空间为容器提供对应的底层 Linux 系统视图,即限制容器的查看与访问范畴。为了给每个运行的容器提供一个独立的运行环境,Docker 会创建一组命名空间来供特定的容器使用。

Docker 会在内核中使用多种不同类型的命名空间,实现对不同系统资源的隔离,包含:

pid 命名空间:用来进行进程隔离 PID,使两个不同命名空间下的进程可以有同一个 PID。

net 命名空间:用来管理网络接口 NET(Networking),主要提供对网络资源的隔离,包括网络设备、IPv4 和 IPv6 协议栈、IP 路由表、防火墙、/proc/net 目录、/sys/class/net 目录、端口(socket)等。

ipc 命名空间:用来管理进程间通信资源 IPC,负责对运行在每套容器内的进程进行 IPC 资源隔离。

mnt 命名空间:用来管理挂载点 MNT(Mount),使不同 mount 命名空间内的进程拥有彼此不同的文件系统结构。

uts 命名空间:用来隔离内核和版本标识 UTS(Unix Timesharing System),允许容器拥有不同于其他容器以及主机系统的主机名称与网络信息服务(Network Information Service, NIS)域名。

user 命名空间:用来对每套容器内的用户进行隔离,允许各容器拥有不同的 uid(UserID)与 gid(Group ID)视图区间,使得某一进程的 uid 与 gid 在用户命名空间之内与之外即有所不同,从而使得该进程能够在不影响容器内 root 权限的情况下,撤销同一用户在容器外的权限。

Docker 将这些命名空间结合起来完成隔离并创建容器。每个容器都有自己单独的名字空间,保证了容器之间彼此互不影响。运行在名字空间中的应用都像是在独立的操作系统中运行一样。

2) 控制组。

控制组(cgroups)是 Linux 内核提供的一种机制,主要用来限制、记录、隔离进程组所使用的物理资源(包括:CPU、内存、磁盘 IO 等)。cgroups 为容器实现虚拟化提供了基本保证,是构建 Docker 虚拟化管理工具的基础。只有能够对分配到容器的资源进行控制,才能够避免当多个容器同时运行时对系统资源的竞争。也就是说,cgroups 能够确保 Docker 容器只能使用分配给它的资源,并在必要情况下设置其所能使用的资源上限。另外,cgroups 还能够确保不会因为单一容器占用太多资源而导致整体系统陷入瘫痪。

3) Union 文件系统。

Union 文件系统(UnionFS)是一种分层、轻量级并且高性能的文件系统,它支持对文件系统的修改作为一次提交来一层层的叠加,同时可以将不同目录挂载到同一个虚拟文件系统下[16]。Union 文件系统是 Docker 镜像的基础。镜像可以通过分层来进行继承,基于基础镜像,通过分层叠加制作各种具体的应用镜像。所以,使用 Union 文件系统,不同 Docker 容器可以共享一些基础的文件系统层,同时再加上自己独有的改动层,从而大大提高了存储的效率。Docker 镜像层次如图4所示。

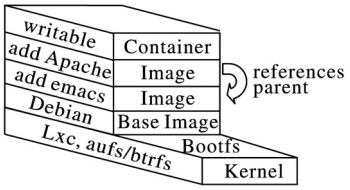
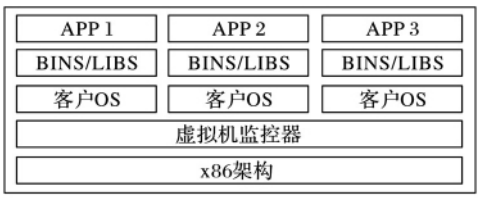


图4 Docker 镜像层次

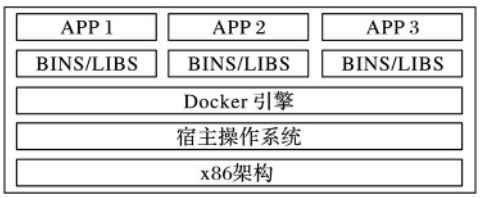
Fig. 4 Docker image layer

2 Docker 容器与虚拟机对比

容器在外观上与虚拟机非常相似,二者皆拥有专有处理空间,能够作为 root 执行命令,提供专有网络接口与 IP 地址,允许定制化路由及 iptable 规则,且可启动文件系统等。但是,容器与虚拟机之间存在着本质的差别:容器是在操作系统层面上实现虚拟化,每个容器拥有自己独立的运行空间,但是不单独拥有操作系统,而是直接复用本地主机的操作系统;而虚拟机则是在硬件层面实现虚拟化,每个虚拟机拥有自己独立的操作系统。架构对比如图5所示。



(a) 虚拟机架构



(b) Docker架构

图5 虚拟机架构与 Docker 架构对比

Fig. 5 VM architecture vs. Docker architecture

图5(a)显示在服务器虚拟化技术下,每个虚拟机不但包括应用本身和必须的二进制文件与代码库,并且还需要包括整个客户操作系统。一般来讲,应用代码只有几十 MB 大小。但是,客户操作系统有时会多达几十 GB 大小。

图5(b)显示了 Docker 对“用户空间”进行打包的方式。与虚拟机不同,容器唯一需要独立构建的只有二进制文件与代码库,而全部操作系统层级的架构都可实现跨容器共享。由此可见,Docker 拥有极为出色的轻量化特性,同时还享受与虚拟机一样的资源隔离与分配的好处。

2.1 Docker 的优势

Docker 是一种新兴的虚拟化方式,与传统的虚拟化方式有着本质的差别,更具有众多的优势。首先,Docker 容器的启动速度要比虚拟机方式快很多,可以在秒级实现;而虚拟机的启动时间一般会需要几分钟时间。其次,Docker 对系统资源的利用率也要比虚拟机高很多,一台主机上同时运行 Docker 容器的数量可以高达数千个;而一台主机上仅能够运行几十个虚拟机。容器运行时基本不消耗额外的系统资源,只需要运行其中的应用,所以使得系统的开销很小,应用的性能很高。如果用户需要运行 10 个不同的应用,采用传统虚拟机方式,一般来讲就需要启动 10 个虚拟机;而采用 Docker 容器只需要启动 10 个隔离的应用即可。

具体说来,Docker 在如下几个方面具有较大的优势:

1) 简化部署。使用 Docker 技术,开发者可以使用一个标准的镜像来构建一套开发容器。开发完成之后,运维人员就可以直接把这个容器部署到运行环境而不需要重新安装。不论需要把服务部署到哪里,容器都可以通过一行命令就完成部署,从而在根本上简化了部署应用的工作。

2) 快速可用。Docker 容器很轻快。容器技术是对操作系统的资源进行再次抽象,而并非对整个物理机资源进行虚拟化。通过这种方式,打包在容器内的服务可以在 1/20 s 的时间内快速启动,而启动一台虚拟机一般可能需要一分钟的时间。

3) 更高效的虚拟化。Docker 容器是内核级的虚拟化,运行时不需要额外的 hypervisor 支持。因此,Docker 容器可以实现更高的性能和效率,可以非常接近裸机的性能。

4) 微服务化。容器允许对计算资源进行比虚拟机更小粒度的细分。如果相对于服务运行所需要的资源来讲,一个小型的虚拟机所提供的资源过于庞大,或者对于用户的系统而言,一次性地扩展出一台虚拟机会所需要工作量很多,那么容器可以很好地解决这类问题。

5) 更轻松的迁移和扩展。Docker 容器可以在各类平台上运行,包括物理机、虚拟机、公有云、私有云、个人电脑、服务器等。这种广泛的兼容性可以让用户很方便地把一个应用程序从一个平台直接迁移到另外一个平台,不用担心平台锁定问题。

6) 更简单的管理。使用 Docker,可以通过微小的修改替代以往大量的更新工作。这些修改都可用增量的方式被分发和更新,从而实现代码更新自动化,大大提高管理的效率。

2.2 Docker 的劣势

与任何技术一样,Docker 也不会是一个完美无缺的系统。相对于虚拟机来讲,Docker 还存在以下几个劣势需要作进一步改进:

1) 资源隔离问题。

Docker 是利用 cgroup 实现资源限制的,但是只能限制每个容器资源消耗的最大值,而不能隔绝其他程序占用自己的资源。也就是说,也许一个容器不能保证任何时候都能够享受分配给它的资源。

比如说,系统给某个容器分配了 4 核 CPU,但是由于系统负载在某个阶段过高,该容器也许只能使用 3 核 CPU。对于有些实时性要求不高的应用,短时间的资源短缺可以接受。

而对于一些实时在线处理系统需要保证比较高并且一致的吞吐量,如果不能保证分配给它的系统资源的固定可用性,那么就不能满足其实时性保证。

2) 安全性问题。

安全性问题是容器技术的一个主要弱点,其资源隔离性没有虚拟机那么彻底,也是阻碍容器技术在实际应用中普及的主要原因。

近年来,Docker 也在不断地通过技术来提高容器的安全性。在早期 Docker 版本中,只有具有 root 权限的用户才能够运行容器,这给容器使用带来了潜在的安全问题。为了解决这个问题,Docker1. 10 引入了用户命名空间功能。但是,Docker 目前还不能分辨具体执行指令的用户,只要一个用户拥有执行 Docker 的权限,就可以对 Docker 的容器进行所有的操作,即使该容器不是由该用户创建的,存在一定的安全风险。

总之,尽管当前的 Docker 版本已经在安全性方面有了很大的提高,但是相对于虚拟机来讲,安全性问题仍然是 Docker 容器技术的一个主要弱点。不克服安全性方面的问题,容器技术就不能适用于实际行业应用。

3) 容器管理需要强化。

因为其轻量快速的特点,容器多用于快速应用部署。每个容器往往只运行一个单一的系统软件或者应用程序,所以,相对于虚拟机可以部署多个系统或应用的场景来讲,容器管理平台需要管理的容器的数量会远远多于虚拟机管理平台需要管理虚拟机的数量。如何管理在一个物理服务器上运行的几千个容器,而不是几十台虚拟机,具有一定的技术挑战。

4) 兼容性问题。

Docker 目前还在版本的快速更新中,细节功能调整比较大,而一些核心模块依赖于高版本内核,存在版本兼容问题。

5) Windows 容器还不成熟。

Docker 容器平台是基于 Linux 内核技术开发的,只能适用于基于 Linux 系统开发的应用和服务,还不能支持 Windows 应用。

尽管微软已经开始研发基于 Windows 操作系统的容器技术,但是与达到实用阶段还有一定的距离。因此,在 Windows 系统方面,虚拟机仍然是解决应用快速部署的唯一选择。

6) 容器编排引擎还不成熟。

分布式应用部署不仅仅是简单地启动一个个装有软件的容器。应用组件之间需要通过网络协议进行通信,因此在使用容器部署分布式应用时,需要设置容器之间的网络接口。网络设置一方面要保证容器之间能够完成所需要的通信任务,另一方面还必须确保容器之间只能够进行允许范围内的通信,以便保证应用的安全性。

分布式应用各个组件的依赖性并不只限于网络连接方面,各个组件之间还存在有服务依赖关系。比如说,一个 Web 应用系统中的 Web 服务器需要连接数据库系统才能正常运行,所以,在启动 Web 服务器之前,必须首先启动数据库系统。因此,在使用容器部署分布式应用时,必须保证装有各种应用组件的容器有序启动。这也需要容器编排引擎来保证。

目前,市场上比较活跃的编排系统有 Google 的 Kubernetes、Mesosphere 的 Marathon 和 Docker 的 Swarm 等。Swarm 为 Docker 容器提供集群功能,可以把多个容器引擎组成一个虚拟容器引擎; Kubernetes 使 Docker 容器在 Google 的云平台上运行,可以把容器编排成一个集群,并且可以根据用户的需求管理容器的负载; Marathon 是 Apache Mesos 的容器编排引擎,主要用来编排和管理长期运行的应用。这些容器编排引擎各有优势,但是现在还没有一个完全成熟能够达到企业应用级。

3　Docker 容器对云计算发展的影响

每一项新技术的出现都可能会对现有局面带来颠覆性的影响。近两年多来,以容器技术为核心的 Docker 引领了一场影响了整个 IT 界的技术革新,不仅在彻底改变着应用开发和发布的方式,并且改变着云计算领域的运行规则。从 IaaS 到 PaaS,乃至大数据平台,Docker 已经给云计算技术实践带来了颠覆性的改变,并深深影响着云计算技术的发展前景。

3. 1　容器即服务

基于虚拟机虚拟化技术的 IaaS 云服务已经进入成熟阶段,人们可以从多个提供商的公有云中方便地获取所需的计算资源和存储资源,基本达到了云计算的“按需获取、按量计费、弹性扩展”的目标。从 IT 管理角度来看,虚拟机提供了一流的资源隔离、安全性和稳定性。但是,旨在提供完整的操作系统的虚拟机虚拟化技术,对资源的分配粒度过大,无法提供最优的资源利用解决方案对外提供服务: 一方面,每个虚拟机都需要包含完整的操作系统,而不能与其他虚拟机共享,因此浪费了大量的系统资源; 另一方面,即使用户只需要运行一个很简单的应用,也需要使用一个完整的虚拟机。

Docker 容器通过使用 cgroups 技术大大降低了控制系统资源的粒度,从而大幅度地提高了对系统资源的利用率。一台物理主机上只能运行几十台虚拟机,但是可以同时运行数千个 Docker 容器。Docker 容器从根本上克服了虚拟机技术的资源浪费问题。其次,Docker 容器的运行不需要额外的 Hypervisor 支持,它是内核级的虚拟化,因此可以实现更高的性能和效率,非常接近裸机的性能。此外,Docker 容器的启动比虚拟机要快得多,可以在秒级完成,而传统的虚拟机方式需要几分钟时间才能启动。

现阶段的云计算是以虚拟机为核心,IaaS 以虚拟机为提供计算资源的基本单元,PaaS 以虚拟机为部署应用的基础设施。随着 Docker 容器技术的成熟,以容器为核心的云计算时代已经开始,容器即服务(Container as a Service, CaaS) 把第一代云计算的 IaaS 层与 PaaS 层合二为一,成为了新一代的云计算架构[17]。基于 CaaS 的云计算架构赋予了企业强大的工业化生产通用软件的能力,再根据消费者和用户的个性化需求,快速组装通用软件形成个性化的解决方案,这就是下一代的云计算商业模式[18]。

在第一个云计算时代,企业主要享受了基于硬件的虚拟化为企业带来的益处,企业可以像使用水、电、煤气等公共服务一样使用 IaaS 服务。而 Docker 的出现,使得 IaaS 服务提供商能够以更细的粒度为企业提供计算资源。这不但可以进一步提高资源利用率,还可以缩短资源启动时间,从而为进一步降低公有云服务的成本提供了可能。

现在的公有云提供商,比如 Amazon AWS,除了提供虚拟

机和存储服务以外，还提供许多通用计算服务，如负载均衡、数据缓存、消息队列和防火墙服务。现在公有云服务提供商完全可以将这些应用迁移到容器中，不但可以降低资源开销，还可以提供更好的可移植性。

混合云是企业现在使用最多的一种云服务模式。使用混合云模式，企业可以根据实际需求，把应用运行在企业的数据中心或者公有云上。因为 Docker 容器相比传统的虚拟机更轻量，所以可以进行动态的设置和迁移。所以，无论是从资源的利用率方面，还是从应用的迁移方便性而言，Docker 容器都比虚拟机更适合部署在混合云中。

CaaS 模式为企业带来的另一个优势就是 CaaS 使企业可以方便地、动态地在不同公有云平台之间迁移服务，而不需要担心平台锁定问题。企业可以很方便地跨 IaaS 平台实现容器动态调度和移动。就像 IaaS 的用户不需要关心其虚拟机使用的哪种虚拟化技术一样，CaaS 的用户也不需要关心他们的容器到底是运行在哪个服务商的云平台之上，是 Amazon 的 AWS 上、Microsoft 的 Azure 上，还是阿里云上。客户只需说明自己期望的应用部署的地理位置，以及他们想要的运行容器，CaaS 服务提供商将为客户提供自动化的编排程序帮助客户进行资源调配，选择最合适的云平台，为用户提供最优质的服务。

尽管 CaaS 这种基于 IaaS 之上提供 Docker 容器的商业模式刚出现不久，但 Docker 已经在云计算行业产生了巨大的影响力。Amazon 的 AWS、Microsoft 的 Azure 和 Google 的云平台都已经开始提供 CaaS 服务。随着 Docker 容器技术和云平台技术的不断发展，今后，CaaS 会在更多的企业，特别是企业的实际生产环境中发挥作用。CaaS 会像 IaaS 一样取得成功，并且会比 IaaS 使用得更广泛。

3.2 基于容器的 PaaS 服务

相对于 IaaS 来讲，Docker 容器技术对于传统 PaaS 更具颠覆意义。近年来 IaaS 技术取得了长足的发展，已经进入了成熟阶段，但是 PaaS 的发展很不理想。现在的 PaaS 平台，一类是提供一个共享的运行平台，包括共享的数据库系统、文件系统、Web 服务器等，托管企业的应用。但是，企业必须基于 PaaS 平台提供的新的存储系统和数据库系统重新实现自己的应用，而不能把原有的应用迁移到 PaaS 平台。此外，因为不同企业的应用共享存储系统和数据库系统，如何保证用户应用和数据的安全性具有相当的挑战，安全性成为用户使用 PaaS 平台的一个巨大障碍。GAE（Google App Engine）和 Microsoft 的 Azure 和 VMware 基于开源平台 Cloud Foundry 提供的 PaaS 服务都属于这类 PaaS 平台。

另外一类 PaaS 平台只在云端提供一些常用的标准软件系统实例供用户使用，比如数据库系统、消息队列、负载均衡系统和数据缓存系统等。对于企业来讲，并不能将 IT 系统整体都迁移到云端，其日常的开发、测试、部署、运维等，除了服务器位于云端之外，并没有得到显著的改善。Amazon 的 AWS 提供的 PaaS 服务是这类 PaaS 的典型代表。

总之，云计算在 PaaS 层面还没有达到为用户提供按需获取服务、按负载进行自动扩展的能力，企业还不能无需改变地把自己已有的应用迁移到云端，更无法动态地在企业数据中心和公有云平台之间进行应用调度和迁移。

以位于操作系统之上的轻量级虚拟化方案和类似于软件版本管理的镜像管理模式等技术为核心，Docker 的出现为 PaaS 平台的实现提供了一条完全不同的路径。不同于现有 PaaS 平台为用户提供一个具有新的存储系统和数据库系统的共享平台，基于容器的 PaaS 平台把用户的每个应用部署到一个单独的容器之中，用户既不需要学习新的存储系统和数据库系统的使用，也不需要重新开发应用，还不需要担心与其他用户共享运行环境带来的安全问题。

总之，容器可以很好地解决当前 PaaS 平台遇到的问题，从而使云服务的提供者有能力提供真正达到云计算标准的 PaaS 服务。从这一层面来讲，Docker 的出现无疑对现阶段云计算的服务能力从 IaaS 向 PaaS 层级的提升有着巨大的推动作用[18]。

当前 PaaS 服务的另一个问题就是平台之间的不兼容问题。不同 PaaS 平台都提供了自己的存储系统和数据库系统，用户的应用必须使用这些 PaaS 平台特有的系统才能正常运行。这就意味着运行在 GAE 平台上的应用无法迁移到 Azure 平台上，反之亦然。

造成 PaaS 平台不兼容的主要原因有两个：

第一个原因是企业应用从开发、管理和运维上都有各种个性化的需求，无法通过提供一个规范、一致的环境来满足各类应用的需求。个性化需求和统一环境之间这种难以克服的矛盾影响了市场对 PaaS 的认可和接受。

造成 PaaS 平台不兼容的另一个原因是因为每一个 PaaS 服务提供商都在为应用提供各自的服务和 API，从而形成了平台特有的运行环境，这就造成了应用很难在 PaaS 服务平台之间进行移植。一些组织曾经在 PaaS 的迁移方面作了积极的努力，希望能够降低企业向 PaaS 平台迁移应用的难度，甚至希望能够实现跨 PaaS 平台的迁移。但是由于没能得到类似 GAE 和 Azure 等主要 PaaS 服务提供商的支持，这些工作想成为事实上的行业标准很困难。

Docker 的出现为 PaaS 平台的实现提供了一条全新的途径，也使为开发者提供更简洁的服务成为了可能。目前，开源 PaaS 平台 Cloud Foundry 已经开始支持并集成 Docker 容器。以 Red Hat 为首开发的 OpenShift 开源 PaaS 平台就是基于 Docker 容器和 Kubernetes 编排引擎开发的。

基于 Docker 容器，开发人员不再需要花费大量精力来处理各种开发、测试和生产环境之间的差异，而直接将应用从开发环境迁移到 PaaS 平台的运行环境，不必担心各种依赖和配置问题。另外，开发者也不再需要为了能够在 PaaS 平台上运行自己的应用而学习额外的编程方式、新的 API，他们的应用不需进行调整就可运行在 PaaS 平台的 Docker 容器中。

使用 Docker 容器，开发者可用微服务的方式来实现他们的应用，将整个应用解耦为较小的功能组件，而容器将组件更进一步从底层的硬件中分离出来，独立运行在一个容器中。通过使用微服务，应用程序可以得到更快的创建，更易于维护，还可以得到更高的质量。所以，Docker 容器技术将会使 PaaS 平台更容易管理，更快速地提供服务。

初期的 Docker 其实并没有容器编排功能，不具备部署企业应用的能力。随着 Docker 的快速成长，围绕 Docker 的生态系统开始完善，多个容器编排引擎已经出现。由于 Docker 能

够运行在具有 Linux 内核的虚拟机中,所以它应该可用在大部分 IaaS 平台上。2015 年 6 月,Docker 公司与 Linux 基金会推出的开放容器计划(Open Container Initiative, OCI) ,包括 Oracle、Microsoft、EMC、IBM、Cisco 和 VMware 等在内的一大批国际著名软件厂商的加入,使 Docker 生态圈开始迅速膨胀。目前,几家著名的云服务提供商都已经宣布支持 Docker 及其生态系统。所以,Docker 容器很有希望成为下一代通用型 PaaS 平台的核心技术。

总之,Docker 的出现已经对还不成熟的 PaaS 市场产生了巨大影响,颠覆了基于原有技术的 PaaS 平台的演进,但同时却加速了新一代基于 Docker 容器技术的 PaaS 平台的发展。尽管目前 Docker 容器及其生态圈还不成熟,但是 Docker 容器通过容器虚拟化的方式提供了一个解决应用环境依赖和可移植性问题的完美方案,为 PaaS 平台的实验提供了一条全新的路径。

4　容器的发展趋势展望

Docker 的出现给传统的基于虚拟机的云计算发展带来了强有力的冲击,那么,是不是虚拟机技术就过时了,要让容器彻底取代了? 容器技术本身能不能以及如何解决目前仍然存在的问题? 基于容器技术的云计算的下一步将朝着哪些方面发展,将会给云计算市场带来哪些根本的变化? 本章将针对这些问题来对容器的发展趋势作一些预测和展望。

4.1　容器和虚拟化技术将会共存

容器和虚拟机的主要差别来源于容器提供了一种虚拟化操作系统的方法使得多个应用可以相互独立地运行在一个操作系统实例上; 而虚拟机则是提供了一种虚拟化硬件的方法使得多个操作系统可以相互独立地运行在一个物理服务器上。尽管容器技术比虚拟机更轻便快速,并具有更好的可迁移性,但是由于容器本身的不完善性以及一些应用场合的特性,容器并不能够完全代替虚拟机成为唯一的虚拟化技术。

首先,虚拟机的安全性要比容器强[19]。其主要原因是因为虚拟机之间是通过硬件实现隔离的,而容器之间共享操作系统和应用库程序,因此容器可能受到的攻击面要大于虚拟机的。第一,容器需要防止宿主操作系统不会被攻破。如果攻击者能够获取访问宿主操作系统,那么就会有机会攻击运行在其上面的容器。第二,用来隔离容器的命名空间设施也是一个潜在的攻击面。攻击者可以通过运行在一个容器的应用去攻击运行在同一台宿主机上的其他容器。况且目前来讲,还不是进程的所有属性都通过命名空间进行了隔离,有些属性还是在容器间共享。第三,操作系统暴露给容器的使用界面要远远大于虚拟机监控程序(Hypervisor) 暴露给虚拟机的界面。任何系统调用方面的缺陷都有可能被攻击者利用来攻击操作系统。

尽管容器的安全性已经得到了一定的改善,但是现在还不能完全保证上述安全问题都得到了解决。对于安全性要求高的用户场景来讲,虚拟机仍然是更好的选择。所以,公有云服务提供商还不能放弃提供虚拟机服务,从而导致所提供服务的适应面变窄。

其次,容器的整个生态链还不完善,用来对容器进行监控、管理和维护的工具还不成熟,而虚拟机经过二十多年的发展已经有了成熟的管理和控制工具生态圈。不管是容器还是虚拟机本身都不能够保证应用可以作为产品的正常运行,都还需要各种管理工具来管理应用的性能、安全和资源。对于一个应用来讲,消耗资源少、启动快和迁移性好是重要的,但这还不够,一个成熟的、经过企业级应用验证的应用架构更为重要。容器目前的生态圈还不成熟,特别是还缺少经过企业级应用验证的应用架构,容器编排引擎也还在不断改进中,因此还不具备替代虚拟机的条件。

第三,容器更适合于需要运行同一个应用的多个实例,或者许多相近应用的场景,但是不太适合于多租户场景。使用容器运行一个应用(如 MySQL) 的多个实例,因为相应的容器之间共享同样的操作系统和代码,从而可以节省大量资源; 同样的,使用容器运行许多相近的应用也可以节省大量的系统资源。

但是,容器并不太适合于多租户场景: 一方面是因为不同租户之间是用的操作系统的版本可能不同,应用之间共享的代码也会有差别,因此节省的系统资源量有限; 更主要的原因是因为容器的安全性弱点,可能导致攻击者使用容器非法入侵其他容器。

总之,因为容器和虚拟机具有各自的优势和弱点,并且具有不同的功能,我们应该把容器虚拟化和虚拟机看成是两个互相补充的技术,而不是相互替代的技术。一个数据中心应该同时支持这两个技术以满足不同应用场景的需求。

如何选择是使用容器还是虚拟机取决于应用的场景。如果需要在有限的资源上运行同一个应用的许多实例,那么容器是最好的选择。比如说,在一个教育云平台上,需要为几百个学生每人部署一套实验环境,那么容器就是必然的选择。

如果应用需要同时使用多种不同的操作系统,或者版本差别较大的操作系统,并且应用的安全性也很重要,那么就应该使用虚拟机来运行这些应用。比如说,需要为某个政府部门部署一套进行数据挖掘的大数据平台。因为大数据平台需要安装 Hadoop、HBase、Hive、Kafka、MySql、Elastic Search 和 SQL Server 等多个系统,这些应用还需要运行在不同的操作系统之上,并且这些应用启动以后会长期运行,而且政府机构对安全性要求都比较高,所以这种场景就需要选择使用虚拟机。

4.2　在虚拟机内运行容器将会成为趋势

容器和虚拟机技术各有优势和弱点,并且各自都有自己的使用场景。如何把容器和虚拟机技术结合,相互取长补短,利用一个技术的优势克服另一个技术的弱点成为了一个热门研究课题。

基本的思路是把容器运行在虚拟机的内部,利用虚拟机的硬件隔离来提高容器的安全性,使用虚拟机丰富成熟的监控和管理工具完善容器的运行环境; 同时,通过简化虚拟机客户操作系统本身来尽量保持容器的轻量和快速的优势。传统虚拟化技术的领军公司 Citrix、VMware 和 Microsoft 都在研究在各自的虚拟化产品里运行容器。

虚拟机技术提供了一个灵活安全且方便使用的运行平台,可以高效部署和管理各种应用; 而容器则提供了一种方便地包装、分发和部署应用的方法。这两种技术处于应用生命周期的不同阶段,如果能够把它们结合起来发挥各自的优势,那么就可以更好地同时满足应用开发者和应用架构管理者的需求。

2015年,Citrix为XenServer虚拟机技术引进了对Docker容器技术的支持,使XenServer可以掌握哪些虚拟机里运行了Docker容器,并把容器的相关信息展示给系统管理员[20]。这样就可以使系统管理员比较方便地使用同一套工具监控、分析和管理容器;同时开发人员还可以使用他们熟悉的工具进行容器的部署和管理。Citrix还计划在XenServer的管理工具中引入容器编排引擎Kubernetes、Swarm和Mesos以便可以快速完成基于容器的应用部署。

VMware的VIC(vSphere Integrated Containers)项目的目的是为使用VMware产品公司的开发团队提供一套产品级容器方案[21]。通过使用vSphere的现有能力,企业IT管理者可以在同一套系统架构上同时运行容器和虚拟机。在虚拟机里运行容器,开发人员可以利用vSphere的安全、网络、存储、资源管理等能力来运行容器应用。VIC包括如下三个核心组件:容器仓库、容器引擎和容器管理。容器仓库用来安全地存储容器的镜像;容器引擎提供了启动容器的远程API;容器管理给应用开发团队提供了管理容器镜像、仓库和宿主机,以及运行容器的界面。

Microsoft与Docker已经宣布合伙把Windows服务器的生态系统引入到Docker社区[22]。Windows为容器提供了两个运行模式:Windows Server容器模式和Hyper-V容器模式。Windows Server容器模式类似于Docker在Linux的模式,容器共享宿主操作系统;Hyper-V容器模式是在轻量虚拟机上运行容器,每个容器拥有自己的操作系统内核。

如上所述,在虚拟机里运行容器将会提高容器的安全性,并且可以利用各种成熟的虚拟机监控和管理工具来为容器服务,但是这种方法是不是会影响容器的性能?无可置疑,在虚拟机内运行容器的性能会低于直接在物理机上运行容器的性能,但是两者之间的差距到底有多大?VMware的测试报告[23]指出两点:第一点,把一个应用运行在基于vSphere虚拟机内的容器上与运行在直接部署在物理机上的容器上相比,两者之间的性能差别不到5%;第二点,把一个应用运行在vSphere虚拟机上与运行在基于vSphere虚拟机的容器上相比,两者之间的性能几乎没有任何差别。所以,用不到5%的性能损失换取安全性能的提高和方便的监控和管理工具是完全值得的。

在虚拟机内运行容器是当前研究的一个热点,特别是传统的虚拟化公司希望借助于自己多年来在虚拟化领域的技术沉淀来迎接容器虚拟化带来的挑战。从当前的初步结果来看,这种方式是一个很有潜力的快速解决容器弱点的途径。相信在传统虚拟化公司的带领下,容器和虚拟机技术会更加融合,在虚拟机内运行容器将会成为一种趋势。

4.3 以容器为中心的云计算时代即将开始

Docker容器技术的出现给云计算行业带来了颠覆性的改变。过去的几年中,在以Docker为代表的容器技术在不断发展的同时,云计算行业也在研究如何迎接容器技术给云计算平台带来的挑战和机会。IaaS平台开发商在研究如何在平台上引进对容器的支持,PaaS平台开发商也在研究如何利用容器技术来改进平台。随着容器技术不断走向成熟,以容器为核心的云计算时代即将开始。通过把第一代云计算的IaaS层与PaaS层的合二为一,CaaS将会成为新一代的云计算架构。

Amazon AWS已经提供容器管理服务ECS(Amazon EC2 Container Service)。ECS提供高度可扩展的高性能容器管理服务,目前它支持Docker容器从而使用户可以在租用的Amazon EC2实例群集上通过容器轻松地运行应用程序。Amazon ECS的用户不需要关心安装、运维、扩展和管理集群基础设施,只需简单地调用API便可以启动和停止Docker应用程序,查询集群的整体状态。同时,用户还可以使用Amazon EC2的其他功能,包括安全组、Elastic Load Balancing和EBS卷等。Amazon ECS还可以根据应用的资源需求和可用性要求在集群中设置和安排容器。

Microsoft Azure也引进了对容器的支持,Azure容器服务提供了一个优化的容器托管服务。用户可以方便地创建、设置和管理专门用来运行容器的虚拟机集群。用户可以设置容器集群的大小,并且可以选择流行的容器编排引擎Kubernetes、Marathon或者Swarm来部署自己的应用。

Google容器引擎是一个功能强大的用来运行Docker容器的集群管理和编排系统。容器引擎负责使用编排和部署用户的容器到集群中,并且负责根据用户定义的需求对容器进行管理。Google容器引擎建立在Kubernetes开源编排系统之上。

开源云计算系统OpenStack从2014年就开始努力添加对容器的支持,并且在Liberty版本中提供了Swarm和Kubernetes编排引擎功能。OpenStack的最终目标是想提供一套兼容的API来统一管理物理机、虚拟机和容器。OpenStack的Magnum项目的任务就是实现在OpenStack中提供多租户CaaS服务。尽管OpenStack当前版本对容器的支持还不理想,相信在后面的版本中会不断进行完善。

Docker不仅掀起了IaaS服务提供商对容器支持的热潮,同时也极大地推动了PaaS平台的发展。PaaS的目标是提供一个让开发者可以快速开发、部署和全周期管理应用的环境,而容器技术的优势就是使应用的开发和管理变得简单方便。早期的PaaS平台支持许多上传应用及其包装和依赖的方法,随着开发者开始使用Docker容器作为包装和转移应用的方法,支持Docker容器已经成为PaaS平台的一种新局势。三大PaaS公司的产品,Pivotal的Cloud Foundry、Red Had的OpenShift和Apprenda,都开始支持容器。

Red Hat在2015年颁布了OpenShift Enterprise 3,一个基于Docker容器、Kubernetes容器引擎和Red Hat Enterprise Linux 7的容器应用平台。

2016年3月Apprenda也宣布在其PaaS平台上支持Kubernetes。Apprenda过去以支持传统的.NET和Java应用为主,通过支持Kubernetes,Apprenda希望成为一个可以管理传统应用、微应用和容器应用的通用平台。

Cloud Foundry从一开始就使用了容器的概念,并引入了容器管理平台Warden,但是该容器格式与Docker容器不一样。随着Docker容器得到了广泛采用,Cloud Foundry也计划支持Docker。为此,Cloud Foundry重新开发了其容器管理平台,改名为Garden。

总之,Docker容器的出现彻底打乱了原来云计算发展的进程,支持容器成为了IaaS和PaaS平台的必要功能,以容器为核心的云计算时代即将开始。

5 结语

Docker 的出现使得以容器为单位的云平台成为可能。相比传统系统虚拟化技术，Docker 可以让更多数量的应用程序在同一硬件上运行; 可以让开发人员更容易快速构建可随时运行的容器化应用程序; 可以大大简化管理和部署应用程序的任务。任何后端的服务程序，都可以封装在 Docker 容器中进行销售、分发和部署，后端开发者能像 Mobile App 开发者那样去做自己的产品。Docker 的分布式特点令其具有大好的发展前景。

但是，传统的虚拟技术并不会被容器取代。容器技术和虚拟机并非简单的取舍关系。举例来说，如果我们需要在多台服务器上运行多款应用程序，并且需要多种操作系统，那么最理想的方案就是使用虚拟机。在另一方面，如果需要运行同一应用程序的多套副本，那么容器则拥有更多成本优势。另外，尽管容器允许开发者将应用程序拆分成多个功能组件，但这种分散化趋势意味着用户需要管理更多功能部件，即带来更为复杂的控制与协调任务。还有，安全也是 Docker 容器需要解决的一大难题。由于各容器共享同一套内核，因此不同容器之间的屏障相对薄弱。与只需要调用主机虚拟机管理程序的虚拟机方案不同，Docker 容器需要向主机内核发出系统调用，而这会带来更庞大的攻击面。

尽管容器技术目前还不够完善，其生态链还不成熟，但已经有很多厂商已经开始使用。同时一批围绕 Docker 建立起来的初创企业已经形成，学习 Docker 风气正盛。可以预见在不远的将来，CaaS 和基于 Docker 的 PaaS 会获得更多云计算服务提供商的支持，越来越多的企业会使用基 Docker 提供的云服务，以容器为核心的云计算时代即将到来。

参考文献(References)

[1] 杨保华，戴王剑，曹亚仑. Docker 技术入门与实战[M]. 北京: 机械工业出版社, 2015: 1 - 15. (YANG B H, DAI W J, CAO Y L. Docker Technology Introduction and Actual Combat[M]. Beijing: China Machine Press, 2015: 1 - 15.)

[2] PLANK J S. Erasure codes for storage systems: a brief primer[J]. ; login: the magazine of USENIX & SAGE, 2013, 38(6): 44 - 50.

[3] BELL J. Apache spark[EB/OL]. [2016-03-10]. http://onlinelibrary.wiley.com/doi/10.1002/9781119183464.ch11/summary.

[4] 从容器和 Kubernetes 技术看现代云计算的发展轨迹[EB/OL]. [2016-10-16]. http://dockone.io/article/140. (The development track of modern cloud computing from container and Kubernetes Technology[EB/OL]. [2016-10-16]. http://dockone.io/article/140.)

[5] 武志学. 云计算导论: 概念 架构与应用[M]. 北京: 人民邮电出版社, 2016: 43 - 52. (WU Z X. Introduction to Cloud Computing: Concepts Frameworks and Applications[M]. Beijing: Posts and Telecom Press, 2016: 43 - 52.)

[6] IBM 虚拟化与云计算小组. 虚拟化与云计算[M]. 北京: 电子工业出版社, 2010. (IBM virtualization and cloud computing group. Virtualization and Cloud Computing[M]. Beijing: Publishing House of Electronics Industry, 2010.)

[7] PRATT I, FRASER K, HAND S, et al. Xen 3.0 and the art of virtualization[J]. Proceedings of the Ottawa Linux Symposium, 2005, 36(5): 164 - 177.

[8] 马博峰. VMware、Citrix 和 Microsoft 虚拟化技术详解与应用实践[M]. 北京: 机械工业出版社, 2013. (MA B F. VMware, Citrix and Microsoft Virtualization Technology Explanation and Application Practice[M]. Beijing: China Machine Press, 2013.)

[9] KIVITY A, KAMAY Y, LAOR D, et al. KVM: the Linux virtual machine monitor[EB/OL]. [2016-03-10]. https://www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf.

[10] ROSEN R. Linux Kernel Networking: Implementation and Theory[M]. Berkeley: Apress, 2014: 405 - 482.

[11] 汪恺，张功萱，周秀敏. 基于容器虚拟化技术研究[J]. 计算机技术与发展, 2015, 25(8): 138 - 141. (WANG K, ZHANG G X, ZHOU X M. Research on virtualization technology based on container[J]. Computer Technology and Development, 2015, 25(8): 138 - 141.)

[12] What is docker [EB/OL]. [2016-10-15]. https://www.docker.com.

[13] MERKEL D. Docker: lightweight Linux containers for consistent development and deployment[J]. Linux Journal, 2014, 2014(239): Article No. 2.

[14] Docker v1.10 正式版 发布[Z/OL]. [2016-03-10]. https://www.oschina.net/news/70489/docker-1-10-0-final. (Docher v1.10 official version [Z/OL]. [2016-03-10]. https://www.oschina.net/news/70489/docker-1-10-0-final.)

[15] BIEDERMAN E W. Multiple instances of the global Linux namespaces[EB/OL]. [2016-03-10]. https://www.landley.net/kdocs/ols/2006/ols2006v1-pages-101 - 112.pdf.

[16] WRIGHT C P, ZADOK E. Kernel korner — unionfs: bringing filesystems together[J]. Linux Journal, 2004, 2004(128): 24 - 27.

[17] PIRAGHAJ S F, DASTJERDI A V, CALHEIROS R N, et al. Efficient virtual machine sizing for hosting containers as a service (SERVICES 2015)[C]// Proceedings of the 2015 IEEE World Congress on Services. Piscataway, NJ: IEEE, 2015: 31 - 38.

[18] 王亚玲，李春阳，崔蔚，等. 基于 Docker 的 PaaS 平台建设[J]: 计算机系统应用, 2016, 25(3): 72 - 77. (WANG Y L, LI C Y, CUI W, et al. Construction of PaaS platform based on Docker[J]. Computer Application Systems, 2016, 25(3): 72 - 77.)

[19] FELTER W, FERREIRA A, RAJAMONY R, et al. An updated performance comparison of virtual machines and Linux containers[C]// Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software. Washington, DC: IEEE Computer Society, 2015: 171 - 172.

[20] Docker support in XenServer, the ultimate guide[Z/OL]. [2016-12-16]. https://xen-orchestra.com/blog/docker-support-in-xen-server-the-ultimate-guide/.

[21] vSphere integrated containers — technology walkthrough[Z/OL]. [2016-12-16]. https://blogs.vmware.com/vsphere/2015/10/vsphere-integrated-containers-technology-walkthrough.html.

[22] Docker containers coming to a Microsoft or Linux server near you [Z/OL]. [2014-10-15]. https://msopentech.com/blog/2014/10/15/docker-containers-coming-microsoft-linux-server-near/.

[23] Docker containers performance in VMware vSphere [Z/OL]. [2016-12-22]. http://blogs.vmware.com/performance/2014/10/docker-containers-performance-vmware-vsphere.html.

WU Zhixue, born in 1960, Ph. D., professor. His research interests include cloud computing, streaming data processing, data mining.