白皮书

推动企业数据中心向现代化环境转型：以 Kubernetes 作为混合云基础

赞助商：IBM

Gary Chen　　Al Gillen

2019 年 9 月

IDC 观点

信息技术 (IT) 专业人员正身处一个不断变化的世界，每隔几年就会涌现出大量新技术，而这些新技术会对普通的企业计算环境产生深远的影响。近来，行业认同了将基于云的部署模型作为理想的长期部署情景的想法，但要将应用程序和数据直接从典型的本地计算环境（其中包括服务器架构、操作环境 (OS) 和虚拟化以及平台服务）迁移至以云为中心的部署环境也并非易事。

对许多组织而言，要从本地计算模型转变成以公有云为基础的平台，最可行的做法是使用混合云计算模型，让客户能够以相对无缝的方式访问本地资源和公有云资源。这种方法可以弥合传统计算架构和现代计算架构之间的差距，让企业能够从长计议，缓步实施，而不是一蹴而就，直接将重要工作负载完全迁移到公有云上。令人欣慰的是，行业已经走上了技术之路，这些技术可以让客户创造一个融合了本地和异地资源常见运营属性的混合云。

借助一个可移植的多平台云平台就可以为混合云提供关键支持。行业正在朝着这个方向发展，Linux、容器和 Kubernetes 是通用抽象层的基础。Amazon Web Services (AWS)、Google、IBM Cloud、Microsoft、Pivotal、Red Hat 和 VMware 等供应商都在开发这样一个平台，可以让客户同时使用服务器和云，还可以使用混合位置。

IBM 最近发布的下一代 Z15 企业平台系统扩展了一些功能，它通过更高的性能、新的系统压缩服务（提高了板载加密功能的吞吐量）和广泛的软件集成，增强了混合云环境的私有云部分。借助 IBM Z 系统，Red Hat OpenShift 将成为广泛使用的软件资源，而借助 IBM 的新 Cloud Paks - 一系列开发人员和部署服务，可加快部署现代开发工具、中间件、人工智能 (AI) 数据服务以及管理资源。

情况概述

在行业为简化信息技术而做出的种种努力中，每一次简化都会带来额外的复杂性，因为它在本质上具有叠加性。过渡到云计算同样如此。当前，IDC 将云计算看作一系列密切相关的技术，其中包含以下几个维度：

- **本地私有云：**其中有传统的硬件和软件，可以用作传统的企业服务器，也可以提供与公有云环境非常类似的体验（从管理层面的角度）。理想情况下，本地私有云系统的工作行为与公有云资源相似，这就意味着，客户要过渡到混合云部署情景，无需深入透彻的学习。
- **异地公有云：**这是通过超大型提供商或较小的云服务提供商提供的第三方云环境。Alibaba Cloud、AWS、Google、IBM Cloud 和 Microsoft Azure 都是常见的异地公有云。此外，公有云

服务由纯软件解决方案提供，部署在多个超大规模云提供商环境之上。Pivotal Cloud Foundry（和 Cloud Foundry Foundation 开源软件解决方案）、Red Hat OpenShift、SUSE Cloud 和 VMware Cloud 都是此类解决方案。

- **混合云：**它更像是一个用例，而不是一种存在根本上不同的技术。混合云服务可在多种云中运行，其中包括本地私有云和公有云，或者将多个公有云（即通常所指的多云）用作常见的基础架构，由通用控制面板进行管理。大多数公有云提供商都采用混合云战略，通常不会有实质性的平台多样性。相反，一些纯软件解决方案，包括 Pivotal、Red Hat、VMware 和 SUSE 的解决方案，都支持混合云和多平台混合云部署情景。

尽管混合环境的本地部分不具备同等横向扩展属性，并且开发人员感兴趣的一些公有云服务在本地都不可用（但可以在远程使用，假设延迟不足为虑），但包含私有云的混合部署情景的人气和关注度却越来越高。对许多客户来说，迁移至包含强大的私有云组件的混合云才是首选方案。

应用程序的可移植性，是困扰客户的一个由来已久的问题。但行业在实现这一目标上已经取得了重大成果，他们采取的方式如下：

- Python 和 JavaScript 等解释语言的使用有所增加。编译语言在过去之所以受到追捧，是因为针对特定平台的编译器优化可实现最大性能。如今，计算资源极大丰富，使用解释语言不会再有性能问题。可以配置现代的持续集成/持续交付 (CI/CD) 管道，以便为使用中的编译语言的不同部署场景创建多套二进制文件。
- 如今，容器已成为在不同基础架构、云和系统架构中广泛使用的抽象层。新应用程序和现有应程序，无论它们是否被设计为一套微服务，都会使用容器进行重新包装。容器现在也可包含多架构支持。
- 行业正在迅速朝着以 Kubernetes 作为下一代应用程序的基础容器管理和编排层的方向发展，有大量可移植的 Kubernetes 环境可用于私有云和公有云部署。在重要系统和云上使用一致的 Kubernetes 平台，可以构建起混合云环境，将资源分散在不同的物理位置。

并非所有容器都包含可以作为 Greenfield 微服务的软件。现实情况是，容器中有各种各样的应用程序，它们反过来又需要不同的外部服务。举例来说，某些容器包含重新托管的单体应用，而其他容器可能对有状态的应用程序进行了部分重构，这些容器需要较高的安全性，并且具有反映其起源环境的其他属性。这些应用程序需要可以满足其所有需求的系统和基础架构软件。

容器和 Kubernetes

容器化 Linux 应用程序的概念并不新鲜，最初也按照 Linux 容器 (LXC) 技术将它作为 Linux 操作系统的一部分变成了现实。但是，LXC 的采用仍十分有限，直到 Docker（公司和技术）横空出世，它能够以令人信服又简单的方式将应用程序封装到容器中。

Docker 和开放容器计划 (OCI) 容器将应用程序及其所有依赖项都封装到单个可移植的容器映像中。然后在集中的容器注册表中共享这些映像，其他开发人员可以对其进行迭代或将其推向生产环节。容器映像也会使用映像中的层概念，以便开发人员可以在现有映像之上轻松构建层。在执行容器时，它们会在自己的沙盒之内运行，因此各个容器之间相互隔离，并且每个容器都有自己完整的 OS。一些开发人员友好的工具和 API 让构建、共享和运行容器变得简单快捷，也受到了用户的欢迎。

容器化应用程序是第一步，因为我们需要一种方式来操作复杂的混合容器化服务。Google 吸收了其内部“Borg”容器编排技术的概念和经验，并在 IBM、Red Hat 等公司的积极参与下创建了开源软件 Kubernetes 项目。Kubernetes 项目的治理工作已移交给新成立的 Cloud Native Compute Foundation (CNCF)，以使该项目真正开放并可供 IT 行业使用。之后，行业将 Kubernetes 作为首选的下一代基础架构，并将它用作容器化应用程序的部署和编排平台。

容器作为一种可移植的通用应用程序封装标准，正在吸引越来越多的目光。其中一个原因就在于，容器经过标准化，是开放容器计划的一部分。OCI 定义了执行（运行时）容器的方式以及如何格式化容器映像。标准化也意味着，容器是一致的，可以相互操作并且可以在不同的容器平台和云服务之间移植。利用 Kubernetes 进行编排和管理之后，标准化还进一步扩展了堆栈。

Kubernetes 不是像 OCI 这样的成文化标准，它是一个独立治理的开源项目，行业参与度极高。因此容器不仅有标准化的格式，大量管理堆栈在不同实施中也是一致的。用于无服务器的 Istio 服务网格和 Knative 等其他组件也开始受到人们的关注，也许某一天它们也会变得无处不在，提供更为常用的堆栈。但是，尽管标准化加快了容器的采用，但最初容器引起人们极大关注，主要是因为容器为开发人员提供了开发生命周期和启用 DevOps 的部署所需的速度。

对开发人员来说，容器可以高效封装全新的云原生微服务，并且可以利用 CI/CD 将这些变化推向自动化程度日益提高的软件构建管道。开发人员友好的 API 可以更方便快捷地使用复杂的软件，从而改进了开发人员工作流。容器还支持自动化程度更高的测试系统，并为包含所有依赖项的容器提供更好的环境控制，因此还可以提高代码质量。最终的结果就是，容器可以更快地开发软件，更快地部署变化并提高开发人员生产效率。

对操作人员来说，容器和 Kubernetes 以一种高度现代化、可扩展和自动化的方式运行大型 Web 应用程序。Kubernetes 嵌入了大型 Web 公司的大量知识和经验，可以可靠地大规模运行快速变化的应用程序。它确立了蓝色/绿色升级等部署模式，针对新应用程序功能的 A/B 测试以及多个自动化扩展选项。此外，容器式部署还可利用不可变的基础架构，从而解决各种配置难题。这就意味着，容器状态定义在其映像中，它永远不会在运行时发生改变。如果需要进行更改，会拿掉原来的容器实例并启动新的映像，而不是对正在运行的映像打补丁或更改配置。容器资源库也有助于集中容器映像并维护版本。容器的便携性和灵敏性再加上现代的控制面板，让 IT 能够高效部署和管理现代应用程序。

在客户向混合云和多云迁移的过程中，容器在确保不同环境的可移植性和一致性方面发挥着重要作用。如前所述，IT 行业广泛采用 OCI 格式和 Kubernetes 控制面板，使得容器平台在其核心上具有相似性。CNCF 提供 Kubernetes 一致性测试和认证，这意味着所有 Kubernetes 的所有核心功能都必须整齐划一。这也意味着，客户可以大规模使用任何 Kubernetes 产品并可获得一定的兼容性。

开发人员可以根据自己的意愿利用完全相同的 API 使用容器和 Kubernetes，而不用考虑分布情况、云服务或底层的基础架构。这样有助于在本地和不同的公共云中形成一致的开发人员环境。此外，容器有助于在不同的系统架构之间进行抽象，因为无论底层的硬件如何，容器接口都保持不变，因此开发人员无需彻底了解系统或 OS 也可为它开发应用程序。对于管理 Kubernetes 的操作人员而言，有一些系统特定的知识以及必须安装和集成的部署工具，但在不同发行版中操作 Kubernetes 和由 Kubernetes 管理的应用程序仍在很大程度上保持一致。

多架构容器和混合云

容器让开发人员可以更轻松地抽象出底层系统的差异。但是，IBM Z 之类的企业平台可以为容器操作和应用程序带来更多优势，并且容器开发人员无需为了使用它们而学习大量新技能。

IBM Z 包含一个基于固件的虚拟机管理程序，它可与 z/VM 或 KVM 软件虚拟机管理程序一起使用，提供安全灵活的计算、内存和 I/O 配置。如今，大多数容器都在虚拟机 (VM) 中运行，而不是在裸机编排环境中运行。尽管这些技术看似相互重叠，但它们绝大多数都在不同的级别工作。虚拟机管理程序可对硬件进行虚拟化和分区，而容器则对 OS 进行虚拟化。

在 VM 中运行容器有很多优势：

- 可获得额外的隔离层，因为 VM 边界比容器边界更强大

- 将当前的大型硬件系统切分成可消费的块（它不需要在单个 OS 内核上整合大量容器，因此可以提高安全性和可靠性。）
- 可以更加灵活地混合不同的操作系统，甚至是同一操作系统的不同版本/补丁级别

IBM Z 平台中这种基于固件的独特的虚拟机管理程序有助于 Kubernetes 灵活扩展。借助虚拟机管理程序，IBM Z 能够以一种非破坏性的方式实现纵向扩展和横向扩展。例如，它补充了 Kubernetes 的垂直 Pod 自动扩缩功能，有助于调整容器的 CPU 和内存配置。

随着企业向微服务架构过渡，当今容器化的大部分内容都是可能会或可能不会重构的传统遗留应用程序。对现有应用程序进行容器化也会有一些优势，这是企业当前广为采用的方法，约有一半的容器化与遗留应用程序相关。IBM Z 及其虚拟机管理程序的灵活性和可靠性让此平台非常适合在过渡期间托管大型单体容器。

IBM 最新发布的 Z 系统还拥有下述软件功能：

- **IBM Cloud Paks：**企业级就绪、容器化的软件解决方案，该方案在 Red Hat OpenShift Container Platform 上运行，打包了 IBM 开源中间件软件
- **IBM z/OS Cloud Broker for the OpenShift Container Platform Hyper ：**为不熟悉该平台的开发者，简化 Z 系统资源的获取
- **IBM z/OS Container Extensions:** 提供执行化境，让容器化的 Linux 应用能够自然得运行于 z/OS

IBM 还提供 IBM Cloud Hyper Protect Services，它是依托 IBM Secure Service Container 技术构建的一系列公共云服务，让开发人员能够使用高度敏感的数据轻松构建应用程序。其中包括：

- **IBM Cloud Hyper Protect Crypto Services：**让客户通过客户控制的硬件安全模块保留自己的密钥以进行云数据加密
- **IBM Cloud Hyper Protect DBaaS：**提供高度安全且易于使用的企业云数据库环境，以便在公共云中实现全面的数据机密性
- **IBM Cloud Hyper Protect Virtual Servers：**对客户管理的虚拟服务器提供完全授权，以处理敏感的工作负载

尽管容器凭借不可改变的基础架构和集中式映像资源库等功能带来了许多安全优势，但它仍是堆栈中要处理的新层，而安全性仍是采用者面临的主要挑战。IBM Z 提供许多供容器使用的安全功能。IBM Z 平台的整体设计，无论是硬件还是软件，都在追求极致的安全性。IBM 密码处理器适配器可实现最高级别的 Linux 安全认证，HSM 提供 FIPS 140-2 Level 4 合规性。IBM Z 还提供一种名为 IBM Secure Service Containers 的功能，它是一种高度安全的逻辑分区，为容器化应用程序交付安全的应用程序执行环境：

- 固件中防篡改、可信的启动顺序，带有隔离内存。
- 限制管理员访问，有助于防止滥用特权凭证，提供使用中的数据保护
- 透明、自动且免费加密所有静态和动态数据
- 可靠性和动态的纵向扩展能力
- 与现有数据/应用程序共存

应用程序可移植性

过去，由于优化、配置和调优以及精选的中间件和数据管理软件产品套件，应用程序不仅会绑定到特定操作系统，还会绑定到特定服务器上的特定操作系统。x86 服务器上虚拟化的广泛使用让所有堆栈 - 操作系统、应用程序和所有相关依赖项 - 都可从一个虚拟化服务器轻松移植到另一虚拟化服务器。尽管这是朝着当今计算环境迈出的重要一步，但最终还是需要从特定的操作系统实例中抽象出应用程序。

要成功完成此抽象工作，需要从底层操作系统的特定实例中集中抽象应用程序及其依赖项。成功实现这一目标，还可降低每个工作负载部署模型一台 VM 和一个操作系统造成的开销。

使用解释语言，例如 JavaScript、Perl 和 Python 等常见的语言以及利用 Java 等字节码编译的语言，进一步增强了平台独立性。解释语言的优势在于可以近乎实时地作出解释，因此可由运行解释语言的平台做出解释，从而避免字节码排序问题 – 这是 IBM Z 系列（高位优先）和 x86 解决方案（低位优先）之间的差别之一。

通过比较，C、C++、Go 和 Haskell 等广泛使用的编译语言会在执行之前进行预编译。这就意味着，编译器可以为要部署应用程序的平台创建二进制代码。这也意味着，为 x86 Linux 平台编译的代码不会在 IBM Z 系统的 Linux 上执行，甚至也不会在 OpenShift Container Platform 等软件云平台上执行。

为解决这一问题，大多数现代的持续集成/持续交付系统都可管理多套从公共源代码编译的二进制文件，并将正确的二进制文件部署到相应的平台。Linux 的交叉编译器可以在 x86 Linux 系统上编译 Z Linux 二进制文件。此外，OCI 容器映像还具有多架构功能，其中的单个映像可包含多个系统类型的二进制文件，而无需处理多个映像。

未来展望

行业方向

信息技术往往会按捺不住对新兴技术的渴盼之情，至少在热情度和接纳度方面可以这么说。但实际情况却是，任何技术都会有支持影响，且可能会延续数十年。因此，客户应该选择那些能为可移植性、灵活性和可支持性提供最佳长期选择的技术。

因为行业已经确定 Kubernetes 用于容器控制面板，因此它成为了创新和集成的纽带，并带来了诸多优势。首先，Kubernetes 本身将从专注于一个平台的庞大社区中受益。其次，有不断增长的项目创新，这些创新并非 Kubernetes 自身的一部分，但可与 Kubernetes 集成并进行优化。

例如，适用于无服务器计算的 Istio 服务网格和 Knative 都是直接在 Kubernetes 技术的基础上构建。这些项目可以培养出一个集体性的大型社区，并成为 Kubernetes 部署的重要部分，创建更广泛、更常用的容器平台。在不同的 Kubernetes 发行版和云服务中提供更常用的组件将使用户受益，因为他们会越来越依赖混合云和多云。

IBM Z 以多种方式直接参与这一社区，包括：

- **IBM Z 上的 Red Hat OpenShift Container Platform：**IBM 对 Red Hat 产品组合觊觎已久且最想收入囊中的就是 OpenShift Kubernetes Platform。这种技术可用作混合云的基础，它能够跨越多个超大规模云环境，其中包括 Amazon Web Services、Google Cloud Platform 和 Microsoft Azure，并为 Istio 和 Knative 提供支持。它与 Red Hat Enterprise Linux 一起支持 OpenShift，Red Hat Enterprise Linux 已经在 LinuxONE 服务器上使用多年。
- **Z 上的 IBM Cloud Paks：**八月初，IBM 为 OpenShift 宣布了“Cloud Paks”概念。这些容器化产品都有单独的产品，用于支持数据和人工智能 (Cloud Pak for Data)、应用程序现代化和云原生应用程序开发支持 (Cloud Pak for Applications)、企业应用程序集成 (Cloud Pak for Integration)、业务流程和决策自动化 (Cloud Pak for Automation)，还有一个专门设计用于多云管理 (Cloud Pak for Multicloud Management)。
- **Z 上的 IBM Cloud Private：**IBM 现有的 IBM Cloud Private 技术以及通过收购 Red Hat 进入 IBM 产品组合的新技术，将继续获得 IBM 的支持和投资保护。
- **Linux 发行版：**它们可用于 IBM Z E，包括 Canonical Ubuntu、Red Hat Enterprise Linux 和 SUSE Linux Enterprise Server。此外，还有多个开源社区发行版可供客户使用，他们希望将这些发行版用于处理非关键工作负载。带有轻量级内核的发行版，例如 Alpine 和 Red Hat Enterprise Linux CoreOS 也可用于容器。

挑战/机遇

挑战：容器将开发人员的很多内容进行抽象化，让他们以同样的方式使用相同的 API，而不论底层的系统为何。但是，底层系统对如何执行、扩展和保护容器仍有重大影响。系统供应商当前面临的挑战，是如何向越来越多的将基础架构视作社区的受众传递这些差异和优势。

机遇：容器让 IBM Z 系统面向全新的现代受众：云原生开发人员。这些开发人员可以像使用容器处理任何其他基础结构一样，为 Z 系统开发容器并部署到 Z 系统。它可以将 Z 开发引入现代工作流，并参与容器部署，而不必专门针对该平台培训开发人员。但是，系统操作人员仍要完成学习曲线，因为他们会承担部署并集成容器平台的任务。

挑战：大多数开发人员都有开发 x86 环境的丰富经验，但其他架构的经验相对较少。

机遇：解释语言带来更高的可移植性；编译语言能够跨编译器支持多种编译情景和多架构容器。如今，人们使用各种非 x86 处理器类型，例如 GPU、ASIC、针对人工智能进行优化的自定义处理器、ARM 处理器以及其他技术，例如边缘的 Raspberry Pi，这就意味着，开发人员在利用各种工具帮助他们支持异质部署环境 – Kubernetes 是这些环境的共同特征。

挑战： IBM 大型机都汇总到“遗留”桶内。

机遇：IBM Z 系统可提供商业平台远不能及的可扩展性、安全性和可靠性，也是重大多云部署机遇的一个重要维度。另外，还有数百种开源工具和软件组成的不断发展的生态系统，这些工具和软件都可在 Z 上使用。尽管开发人员在很大程度上与带有容器的底层系统隔离，并且不需要特殊技能，但他们仍需要了解这些功能，才能在面对不同的需求时，知道容器中哪些类型的应用程序可以使用。承担现场可靠性

工程师角色的系统供应商和客户面临的挑战在于，如何通过容器平台来显示底层系统的价值和差异，从而使 IBM Z 能够捕获非常适合其功能的容器。

挑战：尝试迁移至异地解决方案的客户不希望在自己的数据中心内部署一个大型系统。

机遇： IBM 尚未公布 IBM Z 资源在公共云环境中的可用性。如果 IBM 提出基于消费的定价模式和多云可用性，则可以增加客户使用此平台的机会。IBM 为其全新 z 1 5 系统采用行业标准的 19 英寸框架，也让系统可以轻松适应标准的数据中心。

结语

IT 管理人员正身处一个加速创新的时代，要满足不断增加的需求，他们必须为混合云和多云基础架构制定一份可行的计划，既要满足自己特定的应用程序需求，又能为混合云环境的其他维度提供可扩展性、安全性以及同样不容忽视的可移植性。基础架构和部署软件层的标准化意味着，客户可以并且应该集中自己的资源为应用程序打造优势，包括优化用户体验，支持多个用户设备以及应用程序自身的功能/完整性。当然，它还假定优势中包含可靠性、可扩展性、安全性以及操作成本等核心属性。

IBM 为行业带来了创新的平台，该平台提供了当今开发人员想要和需要的大量软件以及部署体验和属性。这些服务大部分从底层平台中抽象而来，由 Red Hat OpenShift Container Platform 以及 IBM 平台上的一系列安全可扩展的容器部署服务提供支持，将 IBM Z 带给更多希望从 IT 投资中获得行为属性和成果的受众。

关于 IDC

International Data Corporation (IDC) 是市场情报、顾问服务以及活动的主要全球提供商，在信息技术、电信以及消费者技术市场拥有领先地位。IDC 帮助 IT 专业人员、业务主管以及投资团体在充分了解事实的基础上做出有关技术采购和业务战略的决策。在 110 多个国家/地区有 1110 多名 IDC 分析师负责针对技术和行业机遇、趋势提供全球、区域和当地专业知识。50 年来，IDC 提供的深刻战略洞察帮助无数客户实现了他们的重要业务目标。IDC 是全球领先的技术媒体、研究和活动公司 IDG 旗下的一家子公司。

全球总部

5 Speen Street
Framingham, MA 01701
USA
508.872.8200
Twitter（推特）：@IDC
idc-community.com
www.idc.com