Docker 容器及其使用案例调查

Bellishree [P1]，Deepamala 博士。N [2]

[1] 班加罗尔 RVCE CSE 系

2 班加罗尔 RVCE CSE 系副教授

摘要--*Docker 是一个用于开发、运输和运行应用程序的开放平台。它有助于快速交付软件，并为开发和运行应用程序提供便利。*

它将应用程序与基础架构分离开来。Docker 方法促进了代码在任何地方的快速运输、测试和部署，从而减少了代码开发与在生产中部署之间的延迟。它为开发人员和系统管理员带来了更多好处，允许开发人员编写代码，而不必担心代码最终将在哪个系统上运行。此外，它还能减少系统数量，为操作人员提供灵活性。本文讨论了 Docker 容器的基础知识以及该领域正在开展的相关工作。此外，本文还包括关键用例及其使用中的优势和挑战。

关键词 - Docker；容器；虚拟机；Docker 映像；Docker 用例

I. 引言

多年前，在 Docker 容器出现之前，沃尔玛、塔吉特、大通银行等依赖技术的大公司传统上使用服务器，并增加了许多服务器，导致过度分配，这是为了处理用户不断增加的请求。过度分配服务器的 弊端是成本极高，如果不能很好地扩展，服务器就会崩溃和烧毁，企业就会死掉。几年后，一家名为 VMware 的公司提出了 "虚拟化 "的概念，允许在同一台主机上运行多个操作系统，这实质上意味着在同一台服务器和同一基础架构上隔离运行任何应用程序，因此，这似乎是在同一台计算机上运行一台完全独立的计算机，改变了许多行业的游戏规则。

尽管虚拟化概念具有特权，但其成本非常高昂。首先，在基础架构上运行的每个客户操作系统都有多个内核。

e-ISSN: 2395-0056
p-ISSN: 2395-0072

在基础设施中添加系统时，必须分配资源。此外，虚拟化之所以昂贵，是因为即使没有物理硬件，也会有虚拟硬件占用资源、客户操作系统空间以及该操作系统所需的内存分配[6]。

容器化是一种操作系统（OS）虚拟化技术，它将应用程序运行在隔离的用户空间，并使用相同的共享操作系统。Docker 作为一个容器化平台，将所有应用程序及其依赖关系以 Docker 容器的形式组合在一起，以确保应用程序在任何环境下都能无缝运行。Docker 提供了比虚拟机更密集的服务器整合，并能在实例间共享额外的可用内存。[9] 其优势在于：它有利于快速部署；它速度快、重量轻，因为它不会为每个虚拟机启动单独的操作系统，因此启动/停止速度快；由于在映像间共享通用层，它需要的磁盘空间更少；由于映像分层，新版本应用程序的增量部署更小，因此比虚拟机更快；它在容器间共享内核，因此内存用量更少。它具有可移植性，因为 Docker 文件可用于即时配置任何机器。没有管理程序的 Docker 具有更大的优势，因为它不需要单独的内核，仍然可以利用与主机操作系统相同的资源，并利用命名空间和控制组来更有效地利用这些资源。

II. 文献概览

题为 **"OpenStack 和 Docker：为交互式社交媒体应用构建高性能 IaaS 平台 "**的论文**[1]**介绍了 Nova-Docker 插件，该插件可快速高效地调配计算资源，并可作为管理程序运行，从而帮助管理应用程序用户的增长。该插件使用 OpenStack IaaS 构建，可控制用于云计算的数据中心。OpenStack 标准架构包含三个重要角色：Nova，即

计算和存储资源由 Cinder 管理。整个网络资源由 Neutron 跨多个数据中心管理。NUBOMEDIA 是另一种通过云实现（PaaS）互动社交媒体的方法。采用的主要技术有 Kurento Media Server (KMS)，它通过 WebRTC 媒体服务器提供交互式通信。OpenBaton 使用 Docker 容器管理媒体服务器功能的生命周期。为了托管消耗媒体服务器功能的应用程序，启用了 OpenShift Origin。与基于内核的虚拟机相比，开发人员和管理员对 Docker 容器更感兴趣，主要是因为它启动速度快、可直接访问容器，而且可以在任何支持基于 Linux 操作系统的硬件上运行。Docker 容器是轻量级的，能最大限度地减少使用所需资源部署所需的带宽[8]。

题为**《Docker 作为边缘计算平台的评估》[2] 的**论文介绍了如何克服高延迟、网络瓶颈和网络拥塞等问题。我们可以通过将集中式模式转变为分散式模式来实现这一目标，边缘计算将能够缩短应用程序的响应时间，从而获得更好的用户体验。与基于虚拟机的边缘计算相比，基于容器技术的 Docker 平台具有更多优势。本文主要评估了 EC 的基本要求：1）部署和终止，主要描述了提供简单方法来管理、安装和配置服务以部署低端设备的平台。2) 资源和服务管理，允许用户在资源超出限制的情况下使用服务。3) 容错，为用户提供高可用性和可靠性。4）缓存，让用户体验更好的性能，Docker 镜像可以在边缘缓存。在 Hadoop Streaming 上应用的 Docker 概念减少了设置时间和配置错误。总体而言，Docker 还存在需要改进的地方，但它提供了弹性和良好的性能。

题为 **"Docker 容器的模型驱动管理 "**的论文**[3]**关注的是 Docker 容器的管理，主要是用户发现底层系统问题的地方，该论文描述了 Docker 容器建模如何帮助实现 Docker 容器的可持续部署和管理。与 Azure、Amazon 等云计算相比，Docker 系统确实有更多优势；但 Docker 容器在同步方面也存在缺点。

e-ISSN: 2395-0056
p-ISSN: 2395-0072

部署的容器和设计的容器之间的关系。本文提供了一种模型驱动的方法，不仅可以管理容器架构的设计，还可以在目标系统中表示已部署的容器。采用这种模型驱动方法的动机是，目前的 docker 容器缺乏验证和资源管理。本文所描述的架构概览介绍了 Docker 模型、连接器和执行环境这三个组成部分。Docker 模型提供了容器的抽象。连接器提供了 Docker 模型和执行环境之间的链接，它通过生成与执行环境变化相关的特定工件，提供了高效运行 Docker 模型的工具。本文介绍了一种用于验证和同步的模型驱动方法，其未来工作将研究托管容器架构中执行的原子变化。

题为 **"物联网边缘基于 Docker 容器的分析"[4] 的**论文涉及使用传感器的互联网连接设备，这些设备最终会产生大量数据。这就成为计算和处理数据的一项具有挑战性的任务。作为一种变通方法，本文主要介绍了使用 Docker--轻量级虚拟化机制简化物联网端的应用部署。研究主要描述了在英国建立的视频监控馈送的用例，以便使用深度学习分析和检测即将发生的事件。上述用例的实现主要涉及 Raspberry Pi 3 等组件，它被用作网关并从 CCTV 获取视频馈送。Docker Swarm 用于协调监控中获取的帧，多个深度学习框架用于分析馈送，MQTT 客户端用于通知云端服务器帧中出现的线程。该实施方案还树立了一个基准，以实现高效处理高分辨率帧。这项研究表明，深度学习可与基于单机的 Docker 容器结合使用，与裸机部署相比，CPU 处理的开销可以忽略不计。

题为 **"面向高能效异构 Docker 容器的工作负载感知资源管理 "**的论文**[5]**提出了工作负载感知高能效容器（WEEC）中介系统，以应对在多个云服务器中运行容器应用所消耗的能源。该系统展示了如何有效降低能耗。基本上是这样、

e-ISSN: 2395-0056
p-ISSN: 2395-0072

该系统根据多个服务器机架的容器资源利用模式对输入请求进行分类。WEEC 中介系统主要由 4 个子组件组成 1) 每应用能耗表管理器（Power Consumption Per Application (ppA) Table Manager），负责量化容器中异构服务器消耗的初始能效。2) 指数加权移动平均（EWMA）预测未来输入请求需求的工作量。3) 动态合理调整（DRS）有助于管理活动服务器的数量。

4) 请求分配管理器负责将输入请求分配给 Docker 容器中的每个指定服务器。这项研究通过使用基准应用程序和名为 Yocto-Watt 的功率测量设备来测量多个异构 Docker 容器的功率和性能，从而设定了一个基准。

III. 关键的 docker 使用案例 [10]

A. ***简化配置*** - 这是一个基本用例；Docker 配置可在多种环境中多次使用。虚拟机在运行任何平台及其配置时都有优势，Docker 也有同样的优势，但没有任何虚拟机开销，而且可以将配置和环境放入代码中，然后部署代码。基础设施要求与应用环境无关。说到现实生活中的用例，它可以让企业跳过环境设置和配置程序等重复性工作，直接投入开发，从而加快项目设置。

B. ***代码流水线管理*** - 先前的使用案例对代码流水线管理产生了重大影响。在开发人员环境中编写的代码在经过不同阶段（每个阶段的平台/环境都不同）并接近生产阶段时，可能会出现一些细微的差异；然而，Docker 为从开发到生产的所有阶段提供了一致的环境，从而简化了开发和部署管道。此外，Docker 映像的稳定特性和启动的便捷性也有助于实现上述管道管理。

C. ***Docker 提高开发效率*** - Docker 有助于在开发人员环境中实现两个目标--首先是让开发人员接近生产，这可以通过 Docker 来实现，因为它在远程工作时没有开销或开销很小。其次是为交互式使用提供一个活跃的开发环境；Docker 通过共享卷使应用代码可以从容器的主机操作系统访问容器，从而实现了这一点。这样做的好处包括：开发人员可以在自己选择的平台上修改源代码，还可以观察源代码。

D. ***多租户*** - Docker 有助于防止重大应用程序重写，因此可用于多租户应用程序。多租户应用程序的代码库要复杂得多、僵化得多、难以管理得多。重新设计应用程序会耗费大量时间和金钱。使用 Docker 可以轻松创建封闭环境，为每个租户运行多个应用程序实例，并使这一过程变得经济实惠。Docker 提供的应用程序接口易于使用，能够以编程方式启动容器。

E. ***调试功能***--Docker 提供了各种与容器概念娴熟配合的工具。仅举几例，其中之一就是检查点容器及其版本的能力，以区分两个容器，从而在必要时随时修复应用程序。

F. ***更好的灾难恢复***--Docker 映像（别名快照）可以在某个时间点进行备份，并检索以解决任何紧急问题。有了 Docker，文件可以复制到新的、不同的硬件上。Docker 映像允许在同一软件的两个不同版本之间切换。

G. ***改进 DevOps 的采用***--DevOps 社区从 Docker 基础上开发了标准化部署。CI/CD 决定了 Docker 与 DevOps 之间的关联。Docker 保持了测试环境和生产环境的一致性。Docker 系统化了配置界面，简化了机器设置。使用 Docker 可以改进公司的 DevOps。

H. ***快速部署***--在使用虚拟机之前，创建新的硬件资源需要耗费数天的时间，而虚拟化技术将这一过程缩短到了几分钟。另一方面，Docker 通过支持为该过程创建一个容器，无需启动操作系统，从而将时间进一步缩短到几秒钟。它为创建和销毁资源提供了便利，无需担心再次启动的成本。由于降低了创建新实例的成本，资源分配方式变得更加动态。

IV. 船坞使用

A. 何时使用?

i. ***探索最新技术：*** Docker 提供了一个一次性的隔离环境，即使不花很多时间进行安装和配置，也能开始使用新工具。有几个项目维护着带有先前安装和配置的应用程序的 docker 镜像。

ii. ***基本用例：*** Docker Hub 是云上的注册服务，可用于下载其他社区构建的 Docker 镜像。

iii. ***应用程序隔离：*** 将每个应用程序组件放在单独的容器中进行管理，可以避免在不同服务器上运行各种应用程序时产生依赖关系。

iv. ***开发团队：*** Docker 为在不同设置下工作的开发人员提供了本地开发环境与生产环境之间的紧密匹配。

B. 何时不使用?

Docker 并不总是最佳解决方案，下面介绍的几种情况[10]。

i. ***复杂的应用程序：*** 与基本应用不同，使用预先获取的 docker 文件或从 Docker Hub 获取镜像对于复杂应用来说是不够的，因为在不同服务器上编辑、构建和管理多个容器之间的请求和响应非常耗时。

e-ISSN: 2395-0056
p-ISSN: 2395-0072

ii. ***性能关键型应用：***与虚拟机相比，Docker 在性能方面更胜一筹，因为容器共享主机内核并模拟完整的操作系统。为了获得服务器的最佳性能，可以避免使用 docker，因为在本地操作系统上运行的进程会比容器中的进程更快。

iii. ***升级麻烦：***Docker 作为一种新兴技术，仍处于开发阶段，因此需要频繁更新才能体验到新功能。

iv. ***安全是一个关键因素：***当涉及到更复杂的应用程序时，docker 容器化的方法会涉及一些漏洞，如内核级威胁、docker 容器的不一致更新和打补丁、未经验证的 docker 映像、不安全的通信和不受限制的网络流量。

v. ***多操作系统：***由于 docker 容器共享主机的操作系统，因此需要使用虚拟机在不同的操作系统上测试或运行相同的应用程序。

V. 案例研究

将容器集成到工作流中：使用 Makeflow、Work Queue 和 Docker 的案例研究[11]--集群、云和网格等分布式系统被集成用于执行大型科学应用，这解释了工作流系统作为一种广泛使用的抽象概念的主题。不过，工作流在涉及多个环境以及操作系统、数据和某些资源的差异时会遇到障碍。为了解决这个问题，本案例研究讨论了轻量级容器是如何通过利用 Make Flow 和 Work Queue（工作队列）提供一个定义明确的操作系统而产生的。具体来说，本案例讨论了使用 4 种策略将 Docker 与 Make Flow 和 Work Queue 集成的问题；基础架构--基本架构将容器放置在目录中并作为占位符使用，这样可以确保在执行应用程序时不会产生开销。封装脚本（Wrapper-Script）--这解释了可帮助封装本地 docker 的脚本，该脚本可提取所需的映像，并将每个任务分配给一个容器。这样，每个任务都将彼此隔离。Worker-in-container- 这提供了一种辅助方法，将整个工作流系统放置在一个容器中。

在这种方法中，每个环境只创建一个容器，这样可以避免每个任务的启动和关闭成本。共享容器-- 在这种方法中，每个环境只创建一个容器，这就实现了在同一容器上运行任务而不是删除容器的概念。这就避免了创建多个容器的开销。在工作流系统中创建容器技术后，可以大大降低成本，而不会对分布式环境中的工作流产生任何影响。

IBM Control Desk 现有解决方案：使用 Docker 的容器化案例研究 [13] - IBM Control Desk

IBM Control Desk 案例研究与本文讨论的主题相关。IBM Control Desk 软件提供 IT 服务管理，可简化用户支持和基础设施。本案例研究纯粹是使用 Docker 容器在云上构建控制台，并将其作为边缘平台提供 SaaS。这主要遵循协议进行构建和部署。首先，在管理工作站模式下构建控制台，并将其部署在 DB2 节点上。在 JMS 服务器下为控制台构建一个 docker 镜像。容器之间建立了直接通信网络。每个 Docker 镜像处理一个容器。IBM HTTP 服务器被配置为路由流量通道。通过实施这些，IBM 在应用层面上没有受到任何影响，并朝着云原生架构的方向发展。

Spotify：容器化案例研究 - Spotify 是一款使用微服务架构的数字音乐服务，可提供音乐流媒体、通过全球社交媒体凭据登录等一系列功能。该应用程序的终极音乐流功能让 6000 万用户乐此不疲。Spotify 通过对请求捆绑进 行 容器化来实现微服务，这样单个请求就能通过其界面提供所有信息，从而帮助用户无需重复请求来匹配特定信息。Spotify 自 2014 年以来一直在使用这项技术，并成为战略可扩展性的关键支柱。

VI. 结 论

本文对 Docker 容器的使用案例进行了回顾，调查了各种方法，如将 Docker 作为边缘计算、为交互式社交媒体应用程序构建 IaaS 平台，以及将 Docker 容器与其他应用程序集成。

利用物联网和资源管理提高能源效率的 Docker 容器。本文还总结了 Docker 容器的演变，并延伸到其使用案例和使用领域。

VII. 参考文献

[1] Alin Calinciuc 、 Cristian Constantin Spoiala 、 Corneliu Octavian Turcu、Constantin Filote，"OpenStack 和 Docker：为交互式社交媒体应用构建高性能 IaaS 平台"，2016 年 5 月 19-21 日。

[2] Bukhary Ikhwan Ismail 、 Ehsan Mostajeran Goortani、Mohd Bazli Ab Karim、Wong Ming Tat、Sharipah Setapa、Jing、Yuan Luke、Ong Hong Hoe，"作为边缘计算平台的 Docker 评估"，2015 年高级计算实验室。

[3] Fawaz Paraiso, St´ephanie Challita, Yahya Al-Dhuraibi, Philippe Merle, "Model-DrivenManagement of Docker Containers"., University of Lille & Inria Lille - Nord Europe 2016.

[4] Pankaj Mendki，"基于 Docker 容器的物联网边缘分析"，高级首席工程师，2018 年研发成员。

[5] Dong-Ki Kang、Gyu-Beom Choi、Seong-Hwan Kim、Il-Sun Hwang 和 Chan-Hyun Youn，"Workload-aware Resource Management for Energy Efficient Heterogeneous Docker Containers"，电子工程学院。

[6] 容 器 与 虚 拟 机 ： http://searchservervirtualization.techtarget.com/answer/Containers-vs-VMs-Whats-the-difference.

[7] 了解 架构 架 构 https://docs.docker.com/engine/understanding-docker/ 。

[8] M.Raho, A. Spyridakis, M. Paolino, D. Raho, "KVM, Xen and Docker: a performance analysis for ARM based NFV and Cloud computing," IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)、pp.1-8, November 2015.

[9] R.R. Yadav、E. T. G. Sousa 和 G. R. A. Callou，《Docker 容器与基于虚拟机的虚拟化》：IEMIS 2018论文集》。

[10] 部 署 开 源 Docker 或 高 性 能 系 统 企 业 版 ， https://www.flux7.com/tech/ container-technology/docker/

[11] Chao Zheng 和 Douglas，《将容器集成到工作流中》：使用 Makeflow、工作队列和 Docker 的案例研究。

[12] 控制台现有解决方案：使用 Docker 的容器化案例研 究 https://developer.ibm.com/ technologies/containers/articles/containerization- docker-case-study。