文献翻译

|  |  |  |
| --- | --- | --- |
| 题 目 | Docker: 软件即服务，一个 | |
|  | 操作系统级别的虚拟机框架 | |
| 学生姓名 | 张超 | |
| 专业班级 | 电子信息科学与技术14-01 | |
| 学 号 | 54140720149 | |
| 院 （系） | 计算机与通信工程学院 | |
| 指导教师（职称） | | 陈冬冬（讲师） |
| 完成时间 | 2017年4月25日 | |
|  |  | |

Docker: 软件即服务，一个操作系统级别的虚拟机框架

John Fink

(麦克马斯特大学数字奖学金馆员)

摘 要：Docker 是本地用于64位 Linux 的相对较新的虚拟化方法。与更传统的虚拟化技术相比，Docker 在系统资源方面更轻，提供类似 git 的提交和标签系统，并且可以从笔记本电脑扩展到云。

关键词：虚拟化；自建；定制；Docker

1. 简述

Docker 是本地用于64位 Linux 的相对较新的虚拟化方法。与更传统的虚拟化技术相比，Docker 在系统资源方面更轻，提供类似 git 的提交和标签系统，并且可以从笔记本电脑扩展到云。

1. 介绍

如果你过去几年从事过软件库工作，你可能会记得你的服务器机房的模样。运行 Novell 网络操作系统的机箱连接到庞大的多盘 CDROM 阵列。冰箱大小的太阳盒子。数字的 AlphaServers 运行你的软件库目录。为了运行大部分软件库的重要业务，你需要认真的部署，结果，机房经常混杂着大量的书架，电线和空调设备。随着运行在微型计算机上的 Linux 的出现，这些房间变得稍微小一些，也许稍微不那么复杂，但直到 21 世纪初，服务器机房才真正发生了巨变， 快速采用易于实现的虚拟化，它允许在一台机器上运行多个分立的操作系统。

尽管现代意义上的虚拟化实际上早在 20 世纪 60 年代就发生在利用 360 台机器和 IBM 系统的 VM 上，而且 28 6和 386 芯片都包含了一些虚拟化技术，但直到 2001 年 VMWare 推出其基于 Linux 在 x86 虚拟化产品的成功，虚拟化才真正受到人们关注。由于像 KVM 和 Xen 这样的技术在本地部署和诸如亚马逊网络服务和 OpenStack 框架这样的声称云计算的产品也日渐崛起，混乱的白色 PC 机箱还有利用 CDROM 塔式服务器机房或多或少已经灭绝。今天，你已经可以使用最少的物理机创建多个虚拟机。

virsh # list

Id Name State

----------------------------------

1 trotsky running

2 eris running

3 funhouse running

4 gorgar running

5 shoah running

6 balder2 running

7 asgard running

在作者工作场所的单个服务器上列出虚拟机。

1. 库上的虚拟化

现代虚拟化对软件库的需求因机构不同而有很大差异。一个中小型的IT商城不会做大量的软件开发工作 - 正如许多库中传统的那样 - 可能只需要少量的虚拟机。例如，它可能需要一个已经在网络上有的库，一个用于库管理系统，另一个用于控制用于客户的工作站的镜像。尽可能地，这些将成为那些老式白色机盒的替代品。某些情况下，像给定的库管理系统一样，会受到约定或许可的约束，以尽可能少地偏离传统安装。对于其他人来说，定制安装到虚拟化框架的努力可能不值得付出;在性能或便利方面的收益可能很小。

有开发软件人员的图书馆可能有更复杂的要求。现代的开发环境是不断迭代的。编写代码。编写测试。代码中断。写更多的代码。写更多的测试。打破更多的事情。想象一下，每次你想从头开始测试你的代码时，手动设置一台新机器，甚至是一台虚拟机器。任何使这个过程更高效的事情都可以让开发人员花更多时间完成他们擅长的特定工作。例如，流行的 Virtualbox 软件非常重视面向 GUI，因此已经构建了一个完整的框架，用于创建和销毁 Virtualbox VMs 脚本。

典型的图书馆虚拟化方案主要利用机器级虚拟化。已经提到的系统 - KVM，VMware，Xen，Virtualbox 以及 DOSBox（专门用于运行 DOS 游戏的多平台模拟器）等产品都是机器级仿真器。他们试图尽可能多地模拟软件中的计算环境，直到磁盘驱动器，RAM 分配，图形，硬盘驱动器空间甚至处理器类型。例如，一些机器级虚拟化平台可以在基于 Intel 的平台（如台式PC）上仿真基于 ARM 的系统，如 Raspberry Pi。但是这种类型的虚拟化可能是资源密集型和效率低下的。当在一台计算机上运行多个虚拟机时，您可能会通过分配大型磁盘驱动器或为每个 VM 实例分配多个千兆字节的专用 RAM 来快速运行物理机器的限制。这些问题的解决方法相对比较简单（例如将外部驱动器安装为网络共享），但其他问题则更加困难（如处理有限的RAM）。

进入操作系统级别的虚拟化，该虚拟化尝试在实例之间共享资源，而不是模拟尽可能多的实际机器。典型的操作系统级虚拟化方案将共享 RAM，磁盘空间和内核与客户实例。因此，与同等数量的机器级虚拟化实例相比，任意数量的操作系统级虚拟化实例不太可能耗尽主机系统资源。但是这种灵活性需要付出代价。由于来宾实例必须共享一个内核，因此需要共享一个处理器和操作系统类型，因此无法在 Linux 主机下的 x86 或 Windows 上运行该虚拟 Raspberry Pi。尽管存在这些限制，但由于操作系统级别的虚拟化易于部署和轻量级，因此它正成为开发工作中非常有吸引力的工作流程选项。

1. 使用 Docker 进行虚拟化

Docker 是开源操作系统级虚拟化的开源实现，正在迅速获得开发人员的关注。像大多数优秀的开源项目一样，Docker 结合了许多现有的 Linux 技术和新功能;它使用已经存在的技术，如写入时复制联合文件系统（通常为 AUFS）和 Linux 容器，并将它与许多使其成为以开发人员为中心的功能相结合（因此与传统虚拟机不同试图尽可能多地嘲讽机器的隐喻）：像部署可移植性，版本化，重用和可重复性。

这些功能值得考虑。他们将虚拟机配置的概念从耗时的以系统管理员为中心的模型转变为更注重于面向开发人员的工作流程，具体表现在 Docker 版本系统的 git 类本质与它的差异和标签）。

Docker 的核心是一个虚拟化框架，专注于运行应用程序，而不是模拟硬件，这看起来很容易，但强调了 Docker 等操作系统级虚拟化软件与机器级虚拟化之间的关键区别。机器级别的虚拟机关于忠实的硬件重建 - 直到 RAM 分配，分配多少 CPU，模拟 NIC 等等，而操作系统级别的虚拟化是关于应用程序，而不是机器。大多数情况下，当我们开发，测试和发布软件时，我们关心应用程序，而不是真正的或者虚拟的 - 我们正在开发的特定硬件环境，可能除了模拟历史硬件和其他边缘案例。当我们写邮件列表寻求Drupal的帮助时，我们没有说“我有一台带有4个 Intel Core2 Duo 处理器的 Dell PowerEdge 5100，16GB 的 70ns 内存和两块 Atheros 100GB NIC 卡，我无法将此 Drupal 模块正确地工作。“除非我们有一个相当强的指标，说明我们的问题是硬件绑定的，否则我们专注于软件。碰巧，Docker 也是这样做的。通常没有理由尝试为 Docker 定义应该具有多少内存，硬盘应该多大，或者占用多少 CPU，您通常不会为您在桌面上编写的代码执行此操作。

在实践中，这意味着运行 Docker 的计算机可以运行比运行典型虚拟机的同一台计算机更多的同时运行的虚拟实例。为了说明：在2007年的一台贫血台式机上与机器级虚拟化搏斗时，我无法同时运行两个 Virtualbox 实例。但是，在同一台主机上，我能够运行超过40个 Docker 容器，而不会明显降低主机系统性能。

1. 总结

Docker 经历了一年多的沉重的开发，在那段时间里，它的创建者或多或少地阻止人们试图将Docker 作为生产就绪框架来运行。然而，在 6 月 9 日，Docker 达到了1.0 的状态，因此应该考虑准备好生产实例。重要的是，Docker 1.0 与所有先前版本的 Docker 兼容，所以当 Docker 成为移动目标时开始的项目仍然可以继续使用。

然而，将更深奥的应用程序移植到 Docker 并不是一个简单的过程。 Docker 希望在前台进程中运行，因此有必要将常用程序（如 MySQL 和 Apache ）从通常的后台模式转换为前台模式，并且 Docker 专注于每个容器的一个应用程序（通过 docker-wordpress 和许多其他 Docker 应用程序通过审慎使用 supervisord ）使运行中的产品具有复杂的安装过程，这种过程略低于最佳状态，现在可能更适合于传统的虚拟机或裸机部署。

尽管 Docker 还有很多工作要做，但它正在迅速接近稳定状态，并显示出将其纳入图书馆的虚拟化活动的明确承诺。对于软件开发，当程序员检查他们的代码到 git 中时，Dockerfile 可以包含在源代码中，允许在远程服务器上快速测试代码，或者作为一个演示工具，让其他人快速创建他们自己的应用程序版本，而无需担心特定的建筑指令或依赖管理。 Docker可以用作备份策略 - 作为图书馆网站备份脚本的一部分，Docker 映像可以在服务中断时快速部署。对于常规的系统管理任务，谁不愿意看到一个巨大的安装程序归结为一个单一的命令？

参考文献

1. Fink J. Docker: a software as a service, operating system-level virtualization framework[J]. Code4Lib Journal, 2014, 25: 29.