文章编号:1009-2552(2016)01-0121-03 DOI:10.13274/j.cnki.hdzj.2016.01.031

基于 Docker 的虚拟化技术研究

伍 阳

(武汉邮电科学研究院,武汉 430074)

摘 要:对于容器级的虚拟化技术 Docker,通过分析其关键技术并与虚拟机技术横向比较,得出了其相较于传统技术的优劣势,并对其适用的应用场景做了探讨。分析结果表明,Docker 作为一种轻量级的虚拟化技术,在应用的大规模快速部署方面具有极大的优势。

关键词:Docker;虚拟化;容器

中图分类号:TP305 **文献标识码**:A

Research on virtualization technology based on Docker

WU Yang

(**Wuhan Research Institute of Posts and Telecommunications**, **Wuhan** 430074, **China**)

Abstract: As a container level virtualization technology, this paper discusses the advantage of Docker by analyzing the key techniques it used comparing with virtual machine and the situation suitable for it. The results show that, to be a lightweight virtualization technology, Docker is a better choice when it tries to deploy massive application in a short time.

Key words: Docker; virtualization; container

0 引言

现今,云平台逐渐成为应用部署的主流方式,作为云平台必不可少的核心元素,虚拟化技术的价值日益体现。集中部署 IT 资源并抽象化以实现多用户共享,极大程度地提高了资源的利用率,降低了使用成本[1],另一方面,通过在物理层面和用户操作系统层面之间插入一个精简的软件层,降低上下两层之间的耦合程度,缓解了复杂的软硬件适配问题[2]。目前,虚拟机技术是最主流的虚拟化技术,它实现了极高的隔离度和标准化,但是同样给宿主机资源带来了极大的压力[3]。当虚拟机规模部署时会在宿主机上形成大量重复的内存分页,一方面占用了宿主机的资源,同时也使得虚拟机本身的启动速度非常缓慢[4]。在以应用为中心,强调快速部署的场景下,需要一种新的虚拟化技术。Docker 作为一种新的进程/容器级的虚拟化技术,虽然自身还不完全成熟,但是在隔离度、资源消耗、启动速度等方面具有诸多先天优势,在对 Docker 关键技术探讨的基础上,本文分析得出了其相较于传统技术的优势,并对其应用场景做了探讨。

1 Docker 简介

Docker 是 PaaS 供应商 dotCloud 开发的一个高级容器引擎,Docker 的核心是 LXC,它在 LXC 之上做了进一步封装,重点提高了标准化和可移植性,基于 go 语言开发,并且遵循了 Apache2.0 协议开源规范。LXC 技术是一种轻量级的虚拟化技术,它可以在单个宿主机操作系统上同时运行多个 Linux 系统,在隔离方面,使用 cgroups 技术来实现处理器、硬盘、内存、IO、网络等设备的隔离。LXC 提供了一个拥有自己进程和网络空间的虚拟环境,因而可以在单一节点上实现多个容器同时运行并保证良好隔离,但由于只是容器级的虚拟化技术,所以比之于传统的虚拟机,它是一种轻量级技术,很好地平衡了虚拟化的初衷和对资源消耗,同时其启动速度也非常快。

收稿日期:2015-01-04

作者简介:伍阳(1988-),男,硕士,研究方向为互联网技术。

万方数据

Docker 中有两个重要概念，一个是镜像（Image），另一个是容器（Container）。容器即利用 Docker 构建出来的虚拟环境，可理解为一个自给自足的 Linux 环境，在功能上，与一个完整的 Linux 环境基本一致。镜像类似于虚拟机中的镜像，利用 Docker 运行镜像即可还原出构建的容器。对于镜像，会有基础镜像（base image），就好比一个纯净的操作系统，可以在基础镜像上做修改，比如安装应用、配置某些环境变量，形成我们自己的镜像，利用镜像就可以还原出一个完全相同的环境。

2 隔离性和资源控制实现

作为虚拟化技术必须实现的几大要素之一，隔离性必不可少。由于 Docker 的虚拟化并不像虚拟机那样通过在硬件和用户操作系统之间插入中间层来实现，虚拟环境作为容器/进程运行在宿主机的操作系统上，所以 Docker 隔离性的实现必定要依赖于宿主机所支持的相关隔离特性。Docker 容器的隔离性能主要通过 Linux 内核的 namespace 实现，借助于 pid、net、mnt、uts 等 namespace 分别将各容器的进程、网络空间、文件系统和用户隔离开。用户在容器中的进程都是 lxc-start 进程的子进程，不同用户的进程通过 pid namespace 隔离开，且不同 namespace 中可以有相同进程号。每个 namespace 中的进程只对同一个 namespace 空间中的进程产生影响，借此就能实现各容器中的进程互不影响[5]。

对 Docker 而言，一方面要实现各容器的良好隔离，另一方面还需要对容器资源进行限制和管理。通过 namespace，各容器对外部视角而言实现了独立，但从宿主机内部看来各 namespace 之间存在竞争关系，共同争夺使用宿主机资源，尤其是当应用大规模部署时，必定会出现有冲突的资源请求，此时，避免某一容器过分占用资源，平衡好各容器的需求就成为了一个必须要解决的问题。同时，对云平台而言，要实现 IT 资源的按需使用和计费，对资源的标准化度量和分配也就必不可少。

Docker 依靠 Linux 提供的 cgroup 技术来实现资源控制。cgroup 是由 Linux 内核提供的一种对各进程组所使用的物理资源（处理器时间片，IO 时间片，内存等）进行记录、限制和隔离的技术。在 cgroup 之前，Linux 系统中已经有分立的子系统用于对应资源的管理，如 CPU 子系统就是一个 CPU 资源管理器，用于 CPU 时间片的分配，其他子系统与此类似，而 cgroup 整合了所有的 CPU、IO、内存等子系统，为资源管理提供了一个统一的标准化的框架。

cgroup 提供了以下功能：

（1）资源限制：限制每个进程组可以分配的资源数量，比如内存子系统可以为某一进程组设定一个内存分配上限。

（2）优先级设定：为进程组指定特定的优先级，当各容器进程抢占资源时，宿主机依据优先级算法和各进程组设定的优先级分配资源。

（3）记录资源使用量：对云平台而言，记录某一容器/应用的资源使用量不仅是为了标准化度量以用于计费等增值服务，也是实现平台管理的重要手段。比如，通过在 T1 和 T2 两个时刻读取某一容器的 CPU 使用时间，如果这一时间与（T2 - T1）接近，则可以判断，这一应用/进程很可能发生了死锁，应当予以关注[6]。

（4）进程组控制：如某一应用或者容器发生了死锁，这可以中断这一进程，自然也可以对其恢复。

cgroup 提供的以上功能是 Docker 对容器资源管理的基础。

3 移动性的实现

所谓移动性是指，能够通过某种手段比较方便地还原出文中所构建的环境。LXC 在设计之初就没有考虑移动性的问题，几乎没有任何可移植性，即无法通过标准化的镜像/模板实现虚拟容器的制作、复制、重建，因而 Docker 在 LXC 基础上进行了进一步封装，引入了 AUFS 技术，很好地解决了容器的快速移动和更新问题。

AUFS 主要功能是将不同路径的文件挂载到一个虚拟的文件系统中，并将该虚拟文件系统下的某一个目录设定为不同的权限。在 AUFS 里存在一个类似于分层的机制，对设定为只读权限的分支可以只进行逻辑上的修改而并不实际影响该只读部分。Docker 在构建容器镜像时采用了 AUFS 技术，它首先将根文件系统进行加载，并设定为只读操作权限，完成系统检查后，将一个可写的文件层加载到该根文件系统上，这样一组只读的下层根文件系统和上层的可写文件系统就构成了容器的运行目录。通过 AUFS 技术，每一个容器对下层可读层的修改只会在逻辑上存在于上层可写层中而不会实际影响到下层的只读层，这样多个容器就可以共享同一个只读层而不会互相影响，这个只读层在 Docker 中就称作镜像。

4 Docker 的优势分析

作为一种新的虚拟化技术，Docker 在很多方面相较于虚拟机具有明显的优势，主要表现在以下几个方面。

万方数据

(1)轻巧的技术:在传统的虚拟机技术中,由于为每一个环境虚拟了一整套物理硬件并搭载完整的操作系统,因而在规模部署时会在宿主机上形成大量重复的内存分页,这是虚拟机占用资源的本质原因。而 Docker 的各容器直接共享了宿主机的 Linux 内核,不需要为每个容器重复加载,如图 1 所示,这一方面节约了大量的内存空间,另一方面也大大加快了启动速度,其秒级别的启动时间相较于虚拟机有了数量级的下降,在大规模应用时,这种优势无疑更加明显。

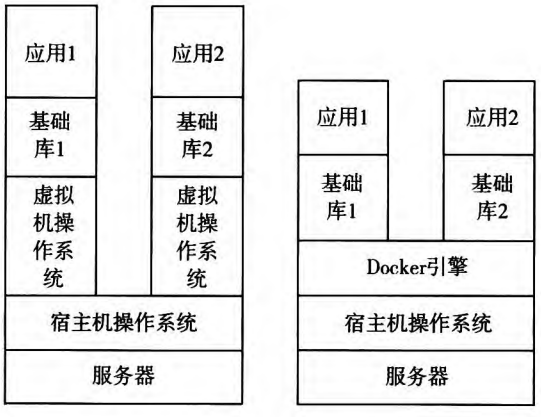

图 1　虚拟机与 Docker 结构图

(2)彻底解决环境适配问题:Docker 的出现非常完美地解决了环境适配问题,虽然是一种进程级的虚拟化技术,但其提供的容器却是自己自足的,拥有独立、完整的运行空间,任何一台机器和系统,只要安装了 Docker,运行特定的镜像,就能构建出一套完全相同的环境,不仅包括操作系统本身,也包括其上搭载的所有应用及配置。

(3)无状态的虚拟化技术:Docker 中的容器是无状态的,也就是说在不改动镜像的前提下,容器中做的改动不会被保存,这样,当用户重新启动容器时,容器会恢复到更改前的状态,这一无状态的特点在一些特定的场合下具有优势。

5　Docker 的应用场景

结合 Docker 的特点,可以对 Docker 的应用场景做以下构想。

(1)快速构建软件开发/测试环境:在企业实践中,搭建开发/测试环境往往是一件非常复杂的工作,开发人员需要完成种类繁多的软件安装和配置工作,尤其是当开发团队规模增长抑或是环境升级变更使得这一工作需要重复进行时整个流程的效率无疑会非常低下。而采用 Docker 构建开发/测试环境会极大地改观这种状况,只需要相应的 ETA(环境工具自动化)小组制作一份符合要求的 Docker 镜像,其他开发人员使用该镜像启动即可还原出完全一致的环境,这无疑会为企业节约大量的时间,显著提高了效率。

(2)快速部署大量标准化应用:如今,应用本身越来越成为整个系统的中心,在很多场景下需要快速部署大量标准化的应用,同时还要保证良好的隔离。此处的快速,不仅是指配置管理方便,解决环境依赖问题,还在于启动速度。如果使用传统的虚拟机技术,由于单台虚拟机的启动时间典型值就达到数分钟,大量虚拟机的部署,即使是在借助相应脚本完全实现自动化的情况下,也将非常耗时。而 Docker 秒级别的启动时间无疑具有非常大的优势,所以 Docker 非常适用于大量应用的快速部署,尤其是当这些应用实现标准化之后,借助于简单的脚本工具就可以非常轻松的完成。

6　结束语

作为一种新的虚拟化技术,Docker 以其极小的性能损耗、秒级别的启动速度和优秀的可移植性在很多场合显现出了相对传统虚拟机技术的优势,但是 Docker 并不是作为现有虚拟机技术的替代者而存在。

虽然同样作为虚拟化技术,Docker 和虚拟机技术存在着很多相似之处,但是从本质上说,Docker 是一种面向应用的虚拟化技术,它以应用为中心,轻巧便捷,因而适合于大量应用的快速部署和管理以及特定环境的还原,同时由于容器镜像小,也便于网络传输;而虚拟机技术面向架构,初衷在于实现 IT 资源的抽象,用于构建统一的 IT 资源池,实现物理资源的抽象和标准化以掩盖底层细节,所以两者的应用场景并不完全一致。在进程级/容器虚拟化技术不成熟的过去,在部分以应用为中心的场合,由于技术成熟度的考虑,选择面向系统的虚拟机技术成为不得已的选择,而在今后,这部分市场必将回到 Docker 等进程级虚拟化技术的怀抱。作为一种更大的可能性,Docker 和虚拟机技术并不会互相替代,而是作为两个层面的虚拟化技术互相配合,取长补短,通过虚拟机构建资源池,解决物理资源的抽象问题,而通过 Docker 实现应用的快速部署和管理。可以预见,在应用本身越来越成为市场核心的今天,Docker 的应用也将越发广泛,尤其是在业界的大力支持下,很有可能成为市场的标准化技术。

参 考 文 献:

[1] 江国忠. 云计算技术浅析[J]. 物联网技术,2013(12):50 - 51.

[2] 陈俐. 服务器虚拟技术在企业信息化建设中的应用[J]. 中国信息化,2012(24):78 - 79.

(下转第 128 页)

万方数据

个通道，采样深度高达 128kB。设计中可以同时例化多个分析仪，每个分析仪均有 10 级触发输入/输出，保证了采样精度。

测试中，首先在系统工程中加入 SignalTap II 模块插件。然后完成选择采样时钟（由锁相环生成，250Mhz）、添加被测信号（包括并行输入输出、时钟等）、配置采样数据（设置采样深度为 128K）和设置触发条件（上升沿连续触发模式）等设置。最后将整体工程重新编译并下载至 FPGA，即可对所添加信号进行实时采样检测，观察数据传输系统运行情况。

4.3 实验结果分析

为了能够首先定性地观察数据传输结果，将发送端和接收端并行数据的最高位经速率调整后接至 LED。系统稳定运行时，可观察到 LED 灯同频正常闪烁，说明发送、接收端数据以相同频率读入和解出。

为了能够进一步更精确地查看数据及控制码传输情况，接下来使用 SignalTap II 在线逻辑分析仪，分别在闭环和开环情况下对相应数据流端口进行采样观察。实验结果如图 5 - 6 所示。

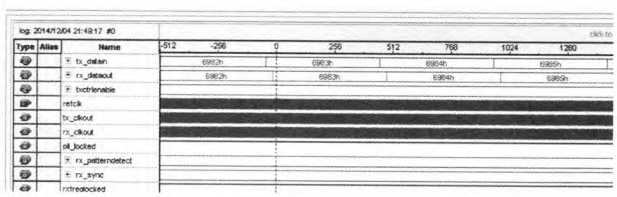

图 5　闭环（收发器 rx_seriallpbken 置高位）采样测试结果

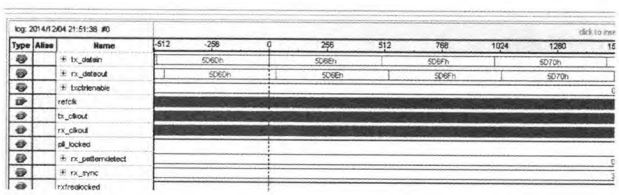

图 6　开环（收发器 rx_seriallpbken 置低位）采样测试结果

可以看到，无论在闭环还是开环情况下，高速收发器模块状态信号均显示正常（pll_lock，rx_sync，etc.），发送端和接收端均可正确发送（tx_datain）和接收（rx_dataout）待传输并行低速数据。实验结果由实际电路芯片测试得到，相较于软件仿真结果有更高的真实度和可信度，证明了该物理层系统的正确性和可用性。

5　结束语

本文设计并实现了一种基于 Altera FPGA 的高速光纤数据传输系统物理层。通过对高速收发器及其相关控制逻辑、重配置模块和时钟生成模块等功能的配置和设计，实现了低速并行数据的高速串行光纤传输。本方案可与其他高层传输协议相组合，通用性高，可移植性强；使用光纤进行传输数据，满足数据传输系统需求；不需增加其他额外设备或芯片，且避免了设计系统物理层的重复性工作，节约了成本、缩短了研发周期。该物理层具有设计简便、易操作的特点，可以根据不同需求完成特征性的配置，具有很高的实用价值。

参 考 文 献：

[1] 邱昆. 光纤通信导论[M]. 成都：电子科技大学出版社，1995：2 - 5.

[2] HU Shan-qing, LIU Feng, LONG Teng. Design and realization of phased array radar optical fiber transmission system[J]. 北京理工大学学报：英文版，2007，16(1)：87 - 92.

[3] 卞敏杰，张锐. 高速数据传输系统的设计与实现[J]. 数字技术与应用，2014 (1)：29 - 30.

[4] 刘卓夫，彭侠夫，李福义. FPGA 在高速数据采集系统中的应用[J]. 计算机工程，2003，29(10)：166 - 167.

[5] Zhang H, Kong L, Huang X, et al. Design of a High Speed XAUI Based on Dynamic Reconfigurable Transceiver IP Core[J]. arXiv preprint arXiv:1401.6370, 2014.

[6] Altera, Inc. Arria II GX Handbook[EB/OL]. http://www.altera.com.cn.2009.

[7] 姚秦. 测控通信中高速数据传输系统的设计与实现[D]. 西安：西安电子科技大学，2013.

[8] Altera, Inc. Serial Digital Interface Reference Design for Arria II GX Devices[EB/OL]. www.altera.com.cn.2010.

[9] 吴朝晖，张持健. 嵌入式逻辑分析仪在 FPGA 实验教学中的应用[J]. 河北工业科技，2010 (1)：41 - 45.

责任编辑：么丽芊

(上接第 123 页)

[3] 蔡志平，陈彬. 虚拟计算环境中的虚拟网络[J]. 计算机工程与科学，2008(11)：1 - 4.

[4] Gao Yi-cong, Feng Yi-xiong, Cheng Jin. Dynamic Composition Method for Innovative Design Based on Virtual Cloud Manufacturing Resource Generators[J]. Advances in Mechanical Engineering, 2013 (10)：23 - 26.

[5] 邵泽云，刘正岐. 云计算与虚拟化技术研究[J]. 数字技术与应用，2014(1)：69.

[6] Robert Denz K, Stephen Taylor. A survey on securing the virtual cloud[J]. Journal of Cloud Computing, 2013(1)：35 - 36.

责任编辑：薛慧心

万方数据