设计研究与应用

基于 Docker 技术的容器隔离性研究

刘思尧，李强，李斌

(宁夏电力设计院，银川 750001)

摘 要: Docker 的核心思想是利用扩展的 LXC(Linux Container)方案实现一种轻量级的虚拟化解决方案。Docker 主要利用 kernel namespace 来实现容器的虚拟化隔离性，保证每个虚拟机中服务的运行环境的隔离。Docker 的隔离机制降低了内存开销，保证了虚拟化实例密度。本文从 Docker 的工作原理出发，详细分析了 Docker 的虚拟化隔离技术和容器隔离方案的实现。

关键词: Docker；LXC；虚拟化；隔离性

中图分类号: TP311 **文献标识码:** A **DOI:** 10.3969/j.issn.1003-6970.2015.04.025

本文著录格式: 刘思尧，李强，李斌. 基于 Docker 技术的容器隔离性研究[J]. 软件，2015，36(4)：110−113

Research on Isolation of Container Based on Docker Technology

LIU Si-yao, LI Qiang, LI Bin

(*Ningxia Electric Power Design Institute*, *Yinchuang* 750001, *China*)

【Abstract】: The core idea of Docker is to use the extended LXC (Linux Container) scheme to achieve a lightweight virtualization solution. By using kernel namespace, Docker realizes isolation of container to ensure the operating environment isolation among each virtual machine service. The isolation mechanism of Docker can reduce the memory overhead and ensure the virtual instance density. From the working principle of Docker, this paper demonstrates the realization of virtual isolation technology and the vessel isolation scheme of Docker in detail.

【Key words】: Docker; LXC; virtualization; isolation

0 引言

Docker[1]是 PaaS[2]提供商 dotCloud 开源的一个基于 LXC(Linux Container)的应用容器引擎，让开发者可以将应用程序、依赖的运行库文件打包并移植到一个新的容器中，然后发布到任何系统为 Linux 的机器上，也可以实现虚拟化[3,4]解决方案。容器是完全沙箱机制的实现方式，任意容器之间不会有任何接口，具有安全访问资源的特性，可以实现系统的隔离；而且容器的运行资源开销小，可以很容易地在机器和数据中心中运行。最重要的是 Docker 容器不依赖于任何特定需求实现的编程语言、编程框架或已打包的系统。Docker 目前在业界非常受欢迎，包括 dotCloud，Google Compute Engine 和百度应用引擎(BAE)，都使用了 Docker。

LXC 是一种共享 Kernel 的操作系统级别的虚拟化解决方案，通过在执行时不重复加载内核，且虚拟容器(Container)与宿主机(Host)之间共享内核来加快启动速度和减少内存消耗。Docker 扩展了 LXC 特性并使用高层的 API，提供轻量级虚拟化解决方案来实现所有容器间的隔离机制。

Docker 具有 LXC 轻量虚拟化的特点，相比较传统的虚拟化，可以做到启动快且占用资源少。本文首先介绍了虚拟化技术，然后阐述了 Docker 的总体系统结构以及每个功能模块的工作原理，最后重点描述 Docker 的隔离性。

1 虚拟化技术

虚拟化技术是通过虚拟机监视器(virtual machinemonitor，VMM)对底层硬件资源进行管理，支持多个操

作者简介: 刘思尧（1987-），女，宁夏银川人，助理工程师，硕士学位，从事电力信息通信方面的系统建设及运维工作；李强（1987-），男，宁夏银川人，助理工程师，硕士学位，从事电力信息通信方面的系统建设及运维工作；李斌（1985-），男，宁夏银川人，工程师，学士学位，从事电力信息通信方面的系统建设及运维工作。

作系统实例同时运行。虚拟化技术的目标是实现资源利用率的最大化，同时将底层的物理设备与上层操作系统应用软件分离，从而实现计算资源的灵活性。对于大型的实验或软件设备，使用虚拟化技术能充分利用高性能的硬件资源，而且有利于设备的综合管理和维护。虚拟化已经在仪器仪表测量技术有运用，在虚拟仪器、自动化测试中，已有很多已经虚拟化的虚拟仪器测量和实验环境。

虚拟化技术在操作系统和硬件之间增加了一层虚拟机监控器，使得操作系统和硬件的紧耦合性大大降低。操作系统和服务运行在虚拟机监控器之上，虚拟机监控器可以灵活地分配虚拟机所使用的硬件资源。同时，虚拟化技术在商业系统中的应用，增强了数据中心的可维护性、促进了服务器的整合(SeverConsolation)。通过虚拟化技术实现的服务器的整合，可以保证每个虚拟机中服务的运行环境的隔离，使得同一台物理机上的服务之间互不干扰。但是，虚拟化技术带有好处的同时也增加了系统复杂度和性能开销。

1.1 完全虚拟化

完全虚拟化是指虚拟机模拟了完整的底层硬件，包括处理器、物理内存、时钟、外设等，使得为原始硬件设计的操作系统或其它系统软件完全不做任何修改就可以在虚拟机中运行。操作系统与真实硬件之间的交互可以看成是通过一个预先规定的硬件接口进行的。完全虚拟化 VMM 以完整模拟硬件的方式提供全部接口(同时还必须模拟特权指令的执行过程)。

完全虚拟化技术最大的好处就是可以无需修改操作系统，直接移植到虚拟环境中，支持多个 GuestOS。但是完全虚拟化的缺点就是虚拟机的 GuestOS 的系统性能会受到影响，而且往往比原有的系统性能下降不少。完全虚拟化虽然能够更容易地支持商业版本的操作系统，但是却大大降低了性能。

1.2 基于容器的虚拟化

容器(coniainer)是虚拟化操作系统环境的软件。在容器下只有一个底层操作系统内核，容器为进程组之间提供增强的隔离性，这样就可以在一个单操作系统下运行多组服务实例，每一个实例可以单独自己的容器内运行。容器并不模拟任何的底层硬件，而是被虚拟的操作系统或应用程序和宿主机交互，宿主机再通过合适的调用从而使用真实的硬件。

基于容器的虚拟化技术根据操作系统内核对不同的进程提供了不同的系统视图，它可以在本地 CPU 核心运行指令，避免了完全虚拟化中指令级模拟或即时编译的系统开销，同时也避免了准虚拟化和系统调用替换的复杂性。

基于容器的虚拟化技术提供一个共享的虚拟化的操作系统，包括唯一的根文件系统，一个安全共享的系统可执行程序和库文件集合。每个 VM 可以像一个普通操作系统那样启动和关闭，在必要的时候甚至可以几秒钟内重启。对于基于容器系统的应用程序和用户来说，VM 就像一个分离的主机。

监控程序从本地服务器和网络上和他连接的服务器分配系统资源给各个容器。基于容器的虚拟化技术最有名的就是 Docker，关于 Docker 容器引擎是如何实现隔离性的分析会在下面详细描述。

2 Docker 的工作原理

2.1 Docker 系统架构

Docker 容器采用如图 1 所示的客户端与服务器(client-server)的架构模式。Docker 服务端会处理复杂的操作任务，例如创建(pull)、运行(run)、保存(commit)Docker 容器等；Docker 客户端则作为服务端的远程控制器，可以用来连接并控制 Docker 的服务端进程。一般情况下，Docker 客户端和守护进程可以运行在同一个宿主机的系统上，也可以使用 Docker 客户端连接一个远程的 Docker 守护进程。Docker Daemon 和 Docker Client 之间可以通过 RESTful API 或者 socket 进行进程间通信。

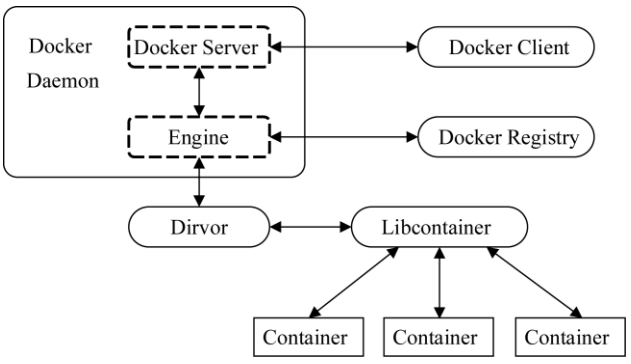

图 1 Docker 系统架构

1) Docker Registry

Docker Registry 是存储容器镜像的管理仓库，而容器镜像是在容器被创建时，被用来加载并初始化容器的文件系统与目录。在容器的运行过程中，Docker Daemon 会与 Docker Registry 进行进程间通信，具有搜索镜像、下载镜像、上传镜像等基本功能。在 Docker 系统结构中，可以使用公有的 Docker Registry，即 Docker Hub。Docker 获取公有的容器镜像文件时，必须通过互联网访问共有的管理仓库；也允许用户构建本地私有的 Docker Registry，这样可以保证容器镜像可以在本地网络获得。

2) Docker container

Docker 容器(Docker container)是 Docker 系统结构中服务交付的最终体现方式。通过 Docker 的需求与下发的命令，订制相应的服务并运行容器：

指定容器镜像，Docker 容器可以自定义 rootfs 等文件系统；

指定计算资源的配额，Docker 容器只能使用指定范围内的资源；

配置网络参数及其安全策略模式，容器具有安全且相互独立运行的网络运行环境；

指定运行的命令参数，使得 Docker 容器可以执行所需的服务进程。

3) Docker Daemon

Docker Daemon 的结构可以分为三部分：Docker Server，Engine 和 Job。

Docker Daemon 是服务端中一个常驻留在后台的守护服务进程，该守护进程在后台启动了一个 Server 进程，其工作职责是接受客户端发送的请求；在 Engine 模块中根据客户端的请求类型， Server 通过路由与分发调度机制，找到相应的 job 来处理客户端的请求。

Engine 是系统结构中的运行引擎，同时也是容器运行的核心模块，通过创建 job 的方式来操纵管理所有客户端发送的请求。

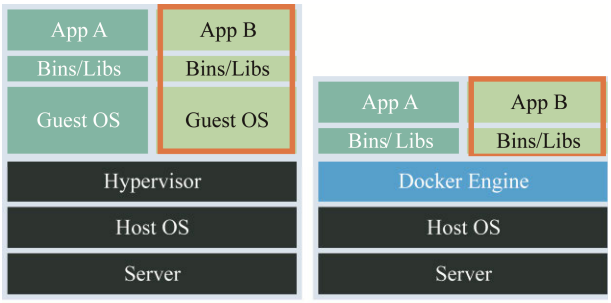

图 2 VM 与 Docker 设计图对比

2.2 Docker 的优越性

相对于完全虚拟化而言，Docker 是基于操作系统级别的轻量级虚拟化技术，而且其底层所依赖的 LXC 技术完全属于内核特性，与 VM(virtual machine)相比不需要对硬件进行仿真就可以共享宿主机所使用的系统内核，没有任何中间层资源的开销，对于资源的利用率非常高且接近物理机的性能，并且使用 AUFS (AnotherUnion File System)和 LXC 技术进行虚拟化的实现。如图 2 所示。

3 隔离性

3.1 虚拟化隔离

通过虚拟化技术实现的服务器的整合，可以保证每个虚拟机中服务的运行环境的隔离，使得同一台物理机上的服务之间互不干扰。

基于容器的虚拟化技术的原理主要是通过隔离操作系统内核对象(例如 PID、UID、系统共享内存、IPC、ptys 等等)来完成安全性的隔离。具体来讲就是运用命名空间和访问权限控制等等技术，将原来的全局对象(句柄、UID 等)隔离到完全不同的命名空间中，不同虚拟机之间是完全不可见的，因此，它们也不能访问到命名空间之外的对象。全局对象在每个容器内本地化了，换言之，全局对象的标记仅仅是在每个容器内存全局。另外，还需要使用一个过滤器，以便在虚拟机运行时检测其是否有权限访问内核对象。容器模型通过创建虚拟操作系统实例来实现虚拟层，宿主操作系统通过 chroot 机制来转换虚拟操作系统的文件系统，内存开销比 Hypervisor 系统级虚拟化平台更低，因此，它的虚拟化实例密度可以很大。由于容器共享文件系统的优势，每个容器都直接使用宿主操作系统为其应用程序提供服务。同时，任何适用于宿主操作系统系统文件的属性子容器都会继承，方便系统管理员对容器进行管理。但是这种机制也存在潜在的风险，一旦宿主操作系统遭到破坏，每个容器也不可避免的遭受破坏，因此需要为容器虚拟化技术加入容错机制来

保证其运行的可靠性[5-7]。

3.2 Docker 的隔离性

Docker 核心解决的问题是利用 LXC 来实现类似 VM 的功能，从而利用更加节省的硬件资源提供给用户更多的计算资源。为了实现每个用户实例之间相互隔离，互不影响，一般的硬件虚拟化方法给出的方法是 VM，而 LXC 给出的方法是 container，更细一点讲就是 kernel namespace。其中 pid、net、ipc、mnt、uts、user 等 namespace 将 container 的进程、网络、消息、文件系统、UTS(UNIX Time-sharing System)和用户空间隔离开。

1) pid namespace

不同用户的进程就是通过 pid namespace 隔离开的，且不同 namespace 中可以有相同 pid。所有的 LXC 进程在 docker 中的父进程为 docker 进程，每个 lxc 进程具有不同的 namespace。同时由于允许嵌套，因此可以很方便的实现 Docker in Docker。

2) net namespace

有了 pid namespace，每个 namespace 中的 pid 能够相互隔离，但是网络端口还是共享 host 的端口。网络隔离是通过 net namespace 实现的，每个 net namespace 有独立的 network devices，IP addresses，IP routing tables，/proc/net 目录。这样每个 container 的网络就能隔离开来。docker 默认采用 veth 的方式将 container 中的虚拟网卡同 host 上的一个 docker bridge: docker0 连接在一起。

3) ipc namespace

container 中进程交互还是采用 linux 常见的进程间交互方法(interprocess communication - IPC)，包括常见的信号量、消息队列和共享内存。然而同 VM 不同的是，container 的进程间交互实际上还是 host 上具有相同 pid namespace 中的进程间交互，因此需要在 IPC 资源申请时加入 namespace 信息，每个 IPC 资源有一个唯一的 32 位 ID。

4) mnt namespace

类似 chroot，将一个进程放到一个特定的目录执行。mnt namespace 允许不同 namespace 的进程看到的文件结构不同，这样每个 namespace 中的进程所看到的文件目录就被隔离开了。同 chroot 不同，每个 namespace 中的 container 在/proc/mounts 的信息只包含所在 namespace 的 mount point。

5) uts namespace

UTS(UNIX Time-sharing System) namespace 允许每个 container 拥有独立的 hostname 和 domain name，使其在网络上可以被视作一个独立的节点而非 Host 上的一个进程。

6) user namespace

每个 container 可以有不同的 user 和 group id，也就是说可以在 container 内部用 container 内部的用户执行程序而非 Host 上的用户。

4 结论

Docker 利用 LXC 来实现基于容器的虚拟化功能，从而将节省的硬件资源提供给用户更多的计算资源。Docker 扩展了 LXC 特性，提供轻量级虚拟化解决方案 kernel namespace 来实现所有容器间的隔离机制，通过 pid、net、ipc、mnt、uts、user 等 namespace 将 container 的进程、网络、消息、文件系统、UTS 和用户空间隔离。

参考文献

[1] Dua, Bangalore, Raja. Virtualization vs Containerization to Support PaaS. IEEE, 2014, 41: 610–614.

[2] 杨莎莎, 邹华. 托管Paas平台安全容器的设计与实现[J]. 软件, 2012, 33(12): 1–5.

[3] Smith JE, Nair R. The architecture of virtual machines[J]. IEEE Transaction on Computer, 2005, (5): 32–38.

[4] Ye K, Jiang X, Ye D, Huang D. Two optimization mechanisms to improve the isolation property of serverconsolidation in virtualized multi-core server 2010.

[5] Love, R. "Linux Kernel Development, Third Edition"[M]. China MachinePress, Beijing. 2011.

[6] Bovet, D. P., and M. Cesati. 深入理解Linux内核(第三版)[M]. 中国电力出版社, 北京. 2007.

[7] Smith, J. E., and R. Nair. 虚拟机: 系统与进程的通用平台[M]. 机械工业出版社, 北京. 2009.