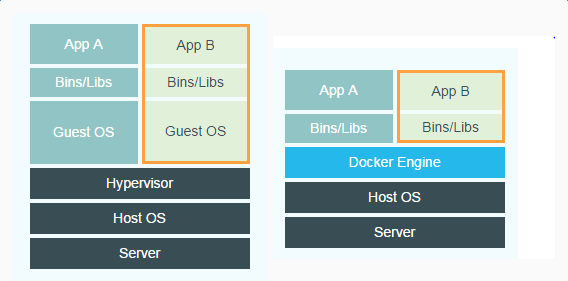
2017218052 燕昭

Docker是一个构建在LXC之上，基于进程容器的轻量级VM解决方案，实现了一种应用程序级别的资源隔离及配额。Docker起源于PaaS提供商dotCloud基于go语言开发，遵从Apache2.0协议。

目前虚拟机所拥有的的问题主要包括资源利用效率低，单物理机多应用无法有效隔离，运维部署不方便，测试、版本管理复杂，迁移成本高，空间占用大、启动慢等，但是使用docker容器技术可以改进这些不足，提高启动速度，降低空间资源占用，可创建上百个docker实例，并且简化版本管理，减少环境配置等复杂操作，加快开发部署测试速度。

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。cgroups 实现了对资源的配额和度量。libcontainer 是Docker中用于容器管理的包，它基于Go语言实现，通过管理namespaces、cgroups、capabilities以及文件系统来进行容器控制。你可以使用Libcontainer创建容器，并对容器进行生命周期管理。从0.9版本起，Docker由Libvirt的LXC与Systemd nspawn提供支持，改为开始采用Libcontainer库直接使用由Kernel提供的虚拟化设施。简单的说，Docker利用Namespaces实现系统环境的隔离，利用Cgroups实现资源限制，利用镜像实现根目录环境的隔离。

**Docker与虚拟机的差异**



左图虚拟机的Guest OS层和Hypervisor层在docker中被Docker Engine层所替代。虚拟机的Guest OS即为虚拟机安装的操作系统，它是一个完整操作系统内核；虚拟机的Hypervisor层可以简单理解为一个硬件虚拟化平台，它在Host OS是以内核态的驱动存在的。 虚拟机实现资源隔离的方法是利用独立的OS，并利用Hypervisor虚拟化CPU、内存、IO设备等实现的。

对比虚拟机实现资源和环境隔离的方案，Docker就显得轻量很多，只是对已有技术的封装，Docker利用的是目前Linux Kernel本身支持的方式实现资源和环境隔离。简单的说，Docker利用Namespaces实现系统环境的隔离，利用Cgroups实现资源限制，利用镜像实现根目录环境的隔离。

计算效率上，Docker有着比虚拟机更少的抽象层。由于不需要Hypervisor实现硬件资源虚拟化，运行在Docker容器上的程序直接使用的都是实际物理机的硬件资源。因此在CPU、内存利用率上Docker将会有明显优势（最少提高10%，IBM的文章说由于Hypervisor对CPU指令集、NUMA支持的不完整性，浮点运算会提高50%，个人感觉也没那么多）。在IO设备虚拟化上，Docker的镜像管理有多种方案，比如利用Aufs文件系统或者DeviceMapper实现文件管理。

启动效率上，由于不需要Guest OS，当新建一个容器时，Docker不需要和虚拟机一样重新加载一个操作系统内核。我们知道，引导、加载操作系统内核是一个比较费时费资源的过程，当新建一个虚拟机时，虚拟机软件需要加载Guest OS，这个新建过程是分钟级的。而Docker由于直接利用宿主机的操作系统，省略了这个过程，因此新建一个Docker容器只需要几秒钟。快速启动、低系统资源消耗的优点使Docker在弹性云平台和自动化运维系统方面有着很好的应用前景。

交付与部署上，Docker做到一次创建或配置，可以在任意地方运行，大量地节约开发、测试、依赖部署等的时间。

相较于虚拟机，Docker也有一些“缺憾”。第一个是资源隔离方面，Docker利用Cgroups实现资源限制，只能限制资源消耗的最大值，而不能完全隔绝其它程序占用自己的资源。第二个是安全性方面，包括鉴权管控、fork炸弹防范等等Docker还是有一些短板，网络资源管理相对简单。另外，Docker对宿主机OS的兼容性也是短板，毕竟现在企业级部署还处在Redhat 5时代。

Docker的优势和劣势：

优势

1. 因为不需要启动内核，所以应用扩缩容时可以秒速启动。
2. 一键启动所有依赖服务，测试不用为搭建环境犯愁，PE也不用为建站复杂担心。
3. Docker命令简单、易用，社区十分活跃，Bug提交一周内就改好，且周边组件丰富。
4. 镜像增量分发，由于采用了Union FS， 简单来说就是支持将不同的目录挂载到同一个虚拟文件系统下，并实现一种layer的概念，每次发布只传输变化的部分，节约带宽。
5. 直接使用宿主机内核调度资源，性能损失小。
6. 通过Docker能实现物理机的冷迁移（docker commit 、docker push）。
7. 动态CPU、内存资源调整。
8. 使用的人再也不用担心如何搭建服务，清理还原服务。镜像一次固化，随处使用，多个应用版本可以并存在机器上。Jenkins已经支持自动化构建镜像，解决了打包发布麻烦的问题。
9. 测试、生产环境高度一致（数据除外）。
10. 应用的运行环境和宿主机环境无关，完全由镜像控制，一台物理机上部署多种环境的镜像测试。
11. 能实现秒级快速回滚。
12. 在高性能计算的场景中，容器热迁移可以保证运行了许多天的计算结果不会丢失，只要周期性的进行检查点快照保存就可以。

劣势

1. 缺乏成熟的开源资源调度、集群管理产品，目前的K8s自成体系，性能问题，使用成本高，Swarm作为官方推荐，功能相对单一，Mesos使用太复杂。
2. Engine发展速度快、平滑升级困难，一般不支持业务无感知升级。
3. 存在隔离与安全的问题。
4. 网络资源管控稍微差一些。
5. 容器中加载、卸载内核模块会影响其他容器。
6. 无法像qemu那样模拟嵌入式系统运行环境。
7. Docker的App Container体系对现有运维体系有着巨大的冲击，在企业级领域推进有很大的阻力（观念、实操等多个层面）。