讨论课 03

复用云技术

软件工程　潘舜达　1354366

一．概要

容器技术虚拟化技术已经成为一种被大家广泛认可的容器技术服务器资源共享方式，容器技术可以在按需构建容器技术操作系统实例的过程当中为系统管理员提供极大的灵活性。由于 hypervisor 虚拟化技术仍然存在一些性能和资源使用效率方面的问题，因此出现了一种称为容器技术（Container）的新型虚拟化技术来帮助解决这些问题。

容器技术已经引起了业内的广泛关注，有充分的证据表明，容器技术能够大大提升工作效率。

因此我们可以选择使用容器技术。只需要通过简单的观察我们便能够发现容器技术的出现是为了解决多操作系统/应用程序堆栈的问题：

（1）在容器技术单台服务器当中为所有虚拟机实例使用相同的操作系统对于大部分数据中心来说都不算是真正的限制。流程管理可以轻松处理这种变化

（2）许多容器技术应用程序堆栈都是相同的

（3）对于容器技术大规模集群来说，在本地硬盘当中存储操作系统副本将会使得更新过程变得更为复杂

最为重要的是，容器技术可以同时将操作系统镜像和应用程序加载到内存当中。容器技术还可以从网络磁盘进行加载，因为容器技术同时启动几十台镜像不会对网络和存储带来很大负载。之后的镜像创建过程只需要指向通用镜像，容器技术大大减少了所需内存。

二．容器技术创业公司

1.Docker

Docker 既是一个开源项目的名称也是一个公司的名称。这个开源项目由一个包括 Docker 公司员工以及其他公司的代码贡献者组成的董事会主导。随着公司的发展,公司也已经在其产品上增加管理功能，例如容器需要的网络控制。开源的 Docker 已成为容器运行的事实标准，这使得 Docker 在商业化容器管理方便提供了一个巨大的机遇，Docker 首席技术官 Solomon Hykes 被认为在推动容器运动过程中发挥了重要作用。

2.ClusterHQ

ClusterHQ 是一家帮助客户构建容器数据层的公司，使得开发和运维团队在容器中运行无状态的应用程序，但是应用的数据库、查询以及 Key-value 数据得到持久化存储。相当于是应用和数据隔离。ClusterHQ 主要基于 Docker 平台实现该服务。

在 2014 年 ClusterHQ 发布了名为 Flocker 的开源软件，这是一个数据卷管理器和多主机的 Docker 集群管理工具，可轻松实现对 Docker 及其数据的管理。同时还有另外一个项目 Powerstrip，为 Docker API 实现了一个可配置、可插入式的 HTTP 代理，可以让你插入很多 Docker 扩展原型到同一个 Docker 守护进程。

3.CoreOS

CoreOS 的创始人认为容器是伟大的，但他们不喜欢 Docker 的一些关于安全和管理方面的设计决策。所以 CoreOS 团队除了推出自己的容器软件 rkt 还精心制作了一个轻量级的基于 Linux 内核的操作系统。类似于开源项目 Docker，rkt 是一个允许容器创建的容器运行版本。

CoreOS 也有成熟的 Tectonic——一个 Kubernetes 商业发行本，如果 Docker 在短期内会有竞争对手，它一定是 CoreOS。

4.BlueData

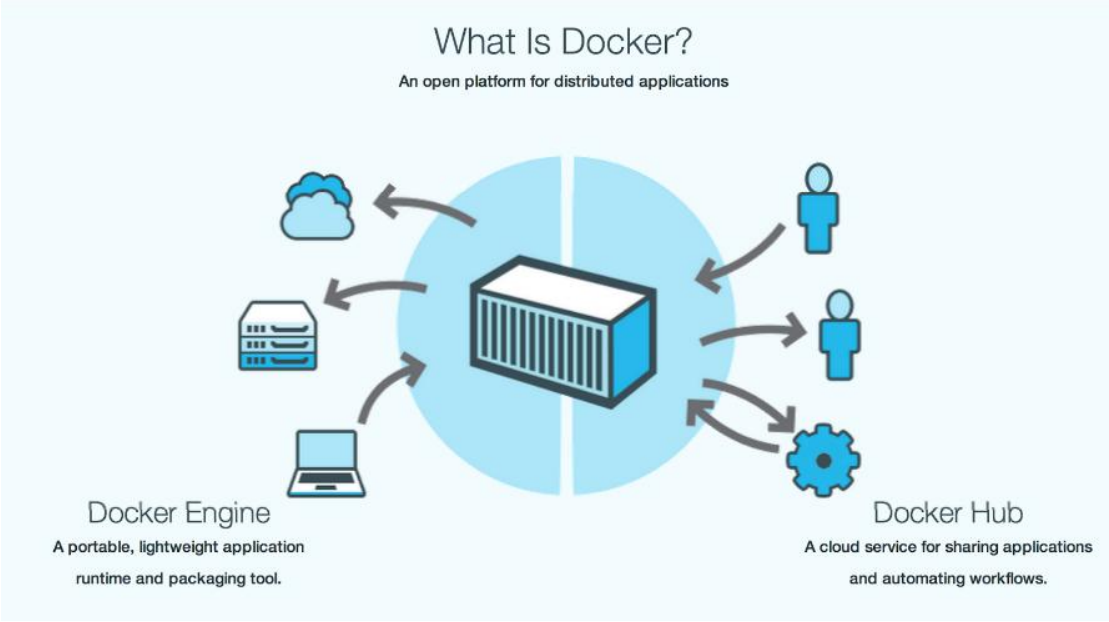
容器技术被视为一个简化应用程序开发的方式，但是一些初创公司正在利用容器技术寻找创新的用例管理应用程序。BlueData 正在前 VMware 研发副总裁 Kumar Sreekanti 的带领下开拓这样的创举。

该公司的目标是通过让大数据更易消费来“民主化”大数据部署，容器技术越来越成为公司战略中重要的一环。BlueData 允许用户在 Docker 容器上部署大数据平台 Hadoop 和 Apache Spark，并且可以通过其 EPIC 平台的免费试用获取容器化版本，它可以作为一个下载程序或一个托管应用程序运行在 AWS ES2 上。BlueData 希望在今年秋天提供更全面的产品。

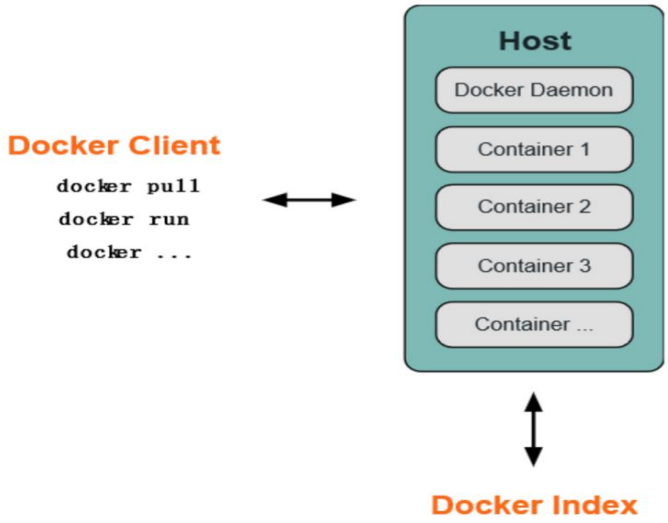
其中规模最大,应用范围最广的毫无疑问是 Docker，所以接下来就 Docker 进行具体的分析。

三．Docker 分析

1.Docker 简介



Docker 是 Docker.Inc 公司开源的一个基于 LXC 技术之上构建的 Container 容器引擎，源代码托管在 GitHub 上，基于 Go 语言并遵从 Apache2.0 协议开源。 Docker 在 2014 年 6 月召开 DockerConf 2014 技术大会吸引了 IBM、Google、RedHat 等业界知名公司的关注和技术支持，无论是从 GitHub 上的代码活跃度，还是 Redhat 宣布在 RHEL7 中正式支持 Docker，都给业界一个信号，这是一项创新型的技术解决方案。 就连 Google 公司的 Compute Engine 也支持 docker 在其之上运行，国内“BAT”先锋企业百度 Baidu App Engine(BAE)平台也是以 Docker 作为其 PaaS 云基础。



2.核心技术预览

（1）隔离性: Linux Namespace(ns)

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。

2.安全性

（1）Docker 的安全性问题：

单单就 Docker 来说，安全性可以概括为两点：

不会对主机造成影响

不会对其他容器造成影响

所以安全性问题 90%以上可以归结为隔离性问题。而 Docker 的安全问题本质上就是容器技术的安全性问题，这包括共用内核问题以及 Namespace 还不够完善的限制：

/proc、/sys 等未完全隔离

Top, free, iostat 等命令展示的信息未隔离

Root 用户未隔离

/dev 设备未隔离

内核模块未隔离

SELinux、time、syslog 等所有现有 Namespace 之外的信息都未隔离

（2）Docker 的解决方案

开源社区在解决 Docker 安全性问题上的努力包括：

1) Audit namespace

作用：隔离审计功能

未合入原因：意义不大，而且会增加 audit 的复杂度，难以维护。

2) Syslognamespace

作用：隔离系统日志

未合入原因：很难完美的区分哪些 log 应该属于某个 container。

3) Device namespace

作用：隔离设备（支持设备同时在多个容器中使用）

未合入原因：几乎要修改所有驱动，改动太大。

4) Time namespace

作用：使每个容器有自己的系统时间

未合入原因：一些设计细节上未达成一致，而且感觉应用场景不多。

5) Task count cgroup

作用：限制 cgroup 中的进程数，可以解决 fork bomb 的问题

未合入原因：不太必要，增加了复杂性，kmemlimit 可以实现类似的效果。(最近可能会被合入)

6)隔离/proc/meminfo 的信息显示

作用：在容器中看到属于自己的 meminfo 信息

（3）企业的努力

一些企业也做了很多工作，比如一些项目团队采用了层叠式的安全机制，这些可选的安全机制具体如下：

1)文件系统级防护

文件系统只读：有些 Linux 系统的内核文件系统必须要 mount 到容器环境里，否则容器里的进程就会罢工。

写入时复制（Copy-On-Write）：所有运行的容器可以先共享一个基本文件系统镜像，一旦需要向文件系统写数据，就引导它写到与该容器相关的另一个特定文件系统中。

2)Capability 机制
Linux 对 Capability 机制阐述的还是比较清楚的，即为了进行权限检查，传统的 UNIX 对进程实现了两种不同的归类，高权限进程（用户 ID 为 0，超级用户或者 root），以及低权限进程（UID 不为 0 的）。
3)NameSpace 机制
Docker 提供的一些命名空间也从某种程度上提供了安全保护，比如 PID 命名空间，它会将全部未运行在开发者当前容器里的进程隐藏。
4)Cgroups 机制
恶意进程会通过占有系统全部资源来进行系统攻击。Cgroups 机制可以避免这种情况的发生，如 CPU 的 cgroups 可以在一个 Docker 容器试图破坏 CPU 的时候登录并制止恶意进程。
5)SELinux
SELinux 是一个标签系统，进程有标签，每个文件、目录、系统对象都有标签。SELinux 通过撰写标签进程和标签对象之间访问规则来进行安全保护。
6)便携性: AUFS
AUFS (AnotherUnionFS) 是一种 Union FS, 简单来说就是支持将不同目录挂载到同一个虚拟文件系统下(unite several directories into a single virtual filesystem)的文件系统, 更进一步的理解, AUFS 支持为每一个成员目录(类似 Git Branch)设定 readonly、readwrite 和 whiteout-able 权限, 同时 AUFS 里有一个类似分层的概念, 对 readonly 权限的 branch 可以逻辑上进行修改(增量地, 不影响 readonly 部分的)。通常 Union FS 有两个用途, 一方面可以实现不借助 LVM、RAID 将多个 disk 挂到同一个目录下, 另一个更常用的就是将一个 readonly 的 branch 和一个 writeable 的 branch 联合在一起，Live CD 正是基于此方法可以允许在 OS image 不变的基础上允许用户在其上进行一些写操作。Docker 在 AUFS 上构建的 container image 也正是如此，接下来我们从启动 container 中的 linux 为例来介绍 docker 对 AUFS 特性的运用。

参考资料
(1).［ECUG 专题回顾］《深入理解容器技术》－田琪（京东资深架构师）
http://blog.qiniu.com/archives/1038
Docker 简介与入门
https://segmentfault.com/a/1190000000448808
(3)深入浅出 Docker（一）：Docker 核心技术预览
http://www.infoq.com/cn/articles/docker-core-technology-preview/
(4)在 Docker 容器中实现安全与隔离
http://www.csdn.net/article/1970-01-01/2826266
(5)容器 VS 虚拟化之安全性
http://www.csdn.net/article/2014-11-07/2822530