软件复用第三次讨论内容

1352875 黄安娜

复用云技术

讨论内容

侧重于容器技术,讨论各种方案以及挑战

- 容器技术的理解
  - 关键技术
    - 隔离
    - 安全
- 如何复用
  - 我们需要和希望解决的问题
  - 选取什么样的容器技术方案

云技术

定义

云技术是指在广域网或局域网内将硬件、软件、网络等系列资源统一起来，实现数据的计算、储存、处理和共享的一种托管技术。云技术（Cloud technology）是基于云计算商业模式应用的网络技术、信息技术、整合技术、管理平台技术、应用技术等的总称，可以组成资源池，按需所用，灵活便利。

容器技术

容器是什么

- 很多人将容器比作虚拟化，但是容器并不能运行所有的东西。例如，你不能在一个容器中设置iptables的防火墙规则;因为容器共享了宿主机的内

核，它们不像虚拟机一样抽象化了硬件层。在开始使用容器之前，将容器看成一种“简单虚拟化”工具会给业务层埋下潜在的危险

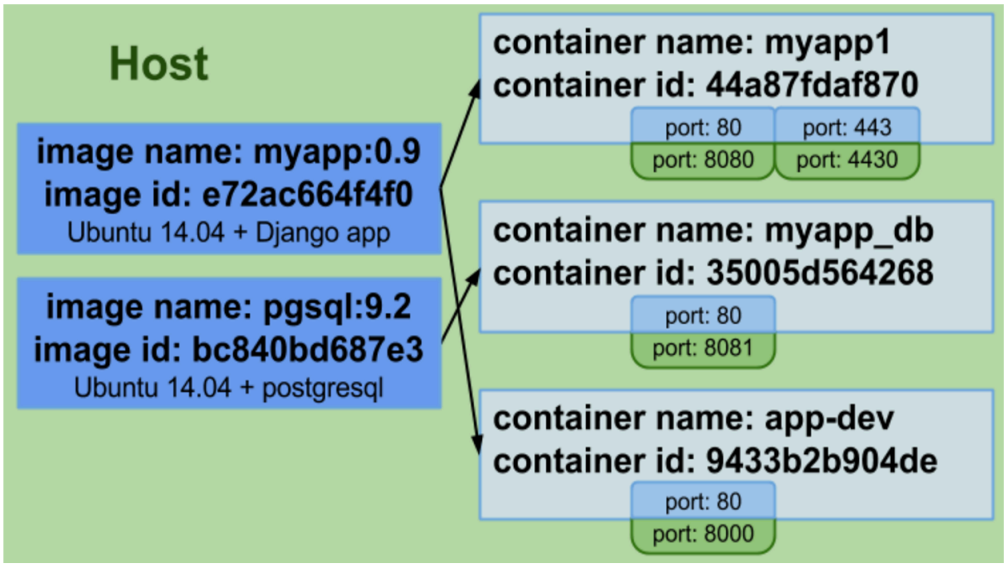
- 容器是一种技术，这种技术让开发者下载一些基础镜像，载入他们的应用程序，拆分成几个组件，在持续集成系统上一一部署和测试，然后将这些组件注册到仓库中，让系统工程师在现存的基础设施上进行部署，然后发布给外界使用。

- 一个容器是操作系统层虚拟化特征的一种抽象。通过开拓这些特征，容器系统能够将进程和网络隔离成类似于系统沙盒的东西，因此将一个应用程序载入一个容器就好像把它从操作系统上隔离出来一样：在容器中的进程是独立的;它们看到的是自己的文件系统，也能够与外界进行通信等等。很多能够跑在虚拟机里的应用程序也能够跑在容器里面。

- 容器和虚拟机的不同在于解耦了虚拟化堆栈;虚拟机将内核到用户空间二进制文件与库文件再到应用程序本身的一切都压入了堆栈中进行隔离和独立存放。而容器管理程序则让容器使用相同的内核，共享相同的用户空间二进制文件，共享其他的库文件，然后让应用程序直接运行在这些层上。

- 一个镜像可以作为其他镜像的Base。在Docker中，在一个镜像之上构建新镜像来获得功能结果非常正常。你也许会在一个Ubuntu镜像上添加Apache 2.4 web服务器来构建一个基于Web的微服务，然后生成新的镜像。

- 镜像是容器运行的基础，你需要在镜像(building block)上启动一个容器(runtime)。镜像中包括了容器的应用程序，类似于一个快照但是还具备两个额外的特征。首先，它们在用户空间文件系统之上进行构建，能够进行堆叠和共享一些内容。其次，它们可移植。这就允许用户在不同的系统上使用相同的镜像，重新利用这些镜像或者通过公共仓库提供给其他用户使用，并且能够轻松地更新这些镜像，然后存放到任何地方。



容器与虚拟机的区别

- 容器提供的隔离对于更密切的软件集成可以声明性地减少。虚拟机则是刚性的。
- 容器运行的是不完整的操作系统（尽管它们可以）。虚拟机必须运行完整的。
- 容器比虚拟机使用更少的闲置资源。它们不运行完整的操作系统。
- 容器在在云硬件（或虚拟机）中可以被复用，就像虚拟机在裸机上可以被复用。所不同的是容器需要毫秒分配。虚拟机需要几分钟。所以，你可以另配、重新平衡、释放以及使用容器比虚拟机的迭代更加迅速。

Docker

Docker是什么

- Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）。几乎没有性能开销,可以很容易地在机器和数据中心中运行。最重要的是,他们不依赖于任何语言、框架或包装系统。
- Doker 使用类似git的非常强大的方法来管理Image，使得它能很方便地管理大量的、不断变化的环境，他的Image分层系统不仅节省空间而且使我

们拥有更细区分度的images。

Docker能做什么

Docker可以解决虚拟机能够解决的问题，同时也能够解决虚拟机由于资源要求过高而无法解决的问题。Docker能处理的事情包括： - 隔离应用依赖 - 创建应用镜像并进行复制 - 创建容易分发的即启即用的应用 - 允许实例简单、快速地扩展 - 测试应用并随后销毁它们

隔离

隔离是什么

云计算中的容器指的就是对计算资源（CPU、内存、磁盘或者网络等）的隔离与划分，比如Docker（基于LXC），就是在Linux系统中划分出了一个不受外界干扰区域（它自己会有自己的文件系统、CPU的配额以及内存及网络使用的配额），然后你可以在这个容器里干自己想干的事，同时又不影响宿主系统和其他容器。

Docker怎么做到隔离的

仅仅使用linux的原始功能来实现安全与隔离，比如namespaces, cgroups, capabilities等

操作系统级的虚拟化、容器、空间以及“chroot with steroids”，其实都定义了同一个概念：用户空间隔离。类似Docker的产品都使用了操作系统级的虚拟化，通过用户空间隔离可以提供额外的安全性。

0.9版本起，Docker包含了libcontainer库作为它直接虚拟化的方法，这个功能由Linux内核提供。 此外，它还通过 LXC,systemd-nspawn,和libvert使用了抽象虚拟接口。 这些虚拟化库全部利用了Linux的原始容器

- namespaces
- cgroups
- capabilities等等。

Docker在一个包装中联合了以上功能，并称之为容器格式。

libcontainer

默认的容器格式被称为libcontainer。

Docker也支持使用LXC的传统Linux容器。在将来，Docker可能会支持其他的容器格式，比如结合BSD jails或者Solaris Zones。

执行驱动程序是一种特殊容器格式的实现，用来运行docker容器。在最新的版本中，libcontainer有以下特性：

- 是运行docker容器的默认执行驱动程序。
- 和LXC同时装载。
- 使用没有任何其他依赖关系的Go语言设计的库，来直接访问内核容器的API。

目前的Docker涵盖的功能有：命名空间使用，cgroups管理，capabilities权限集，进程运行的环境变量配置以及网络接口防火墙设置——所有功能是固定可预测的，不依赖LXC或者其它任何用户区软件包。只需提供一个根文件系统，和libcontainer对容器的操作配置，它会帮你完成剩下的事情。Docker支持新建容器或者添加到现有的容器。事实上，对libcontainer最迫切的需求是稳定，开发团队也将其设为了默认。

在Docker 0.9中，LXC现在可以选择关闭。 如果想要重新使用LXC驱动，只需输入指令docker -d –e lxc,然后重启Docker。 用户命名空间

Docker不是虚拟化，相反的，它是一个支持命名空间抽象的内核，提供了独立工作空间(或容器)。当你运行一个容器的时候，Docker为容器新建了一系列的namespace。

namespaces

一些Docker使用的linux命名空间：

- pid namespace 用作区分进程（PID: Process ID）。 容器中运行的进程就如同在普通的Linux系统运行一样，尽管它们和其他进程共享一个底层内核。
- net namespace 用作管理网络接口。 DNAT允许你单独配置主机中每个用户的的网络，并且有一个方便的接口传输它们之间的数据。 当然，你也可以通过使用网桥用物理接口替换它。
- ipc namespace 用作管理对IPC (IPC: InterProcess Communication)资源的访问。
- mnt namespace 用作管理mount-points (MNT: Mount)。

- uts namespace 用作区分内核和版本标识符(UTS: Unix Timesharing System)。

cgroups

Linux上的Docker使用了被称为cgroups的技术。因为每个虚拟机都是一个进程，所有普通Linux的资源管理应用可以被应用到虚拟机。此外，资源分配和调度只有一个等级，因为一个容器化的Linux系统只有一个内核并且这个内核对容器完全可见。

总之，cgroups可以让Docker：

- 实现组进程并且管理它们的资源总消耗。
- 分享可用的硬件资源到容器。
- 限制容器的内存和CPU使用。
- 可以通过更改相应的cgroup来调整容器的大小。
- 通过检查Linux中的/sys/fs/cgroup对照组来获取容器中的资源使用信息。
- 提供了一种可靠的结束容器内所有进程的方法。

Capabilities

Linux使用的是“POSIX capabilities”。这些权限是所有强大的root权限分割而成的一系列权限。在Linux manpages上可以找到所有可用权限的清单。Docker丢弃了除了所需权限外的所有权限，使用了白名单而不是黑名单。

一般服务器（裸机或者虚拟机）需要以root权限运行一系列进程。包括：

- SSH
- cron
- syslogd
- 硬件管理工具 (比如负载模块)
- 网络配置工具 (比如处理DHCP, WPA, or VPNs)等。

每个容器都是不同的，因为几乎所有这些任务都由围绕容器的基础设施进行处理。默认的，Docker启用一个严格限制权限的容器。大多数案例中，容器不需要真正的root权限。举个例子，进程（比如说网络服务）只需要绑定一个小于1024的端口而不需要root权限：他们可以被授予CAPNETBIND_SERVICE来代替。因此，容器可以被降权运行：意味着容器中的root权限比真正的root权限拥有更少的特权。 Capabilities只是现代Linux内核提供的众多安全功能中的一个。为了加固一个Docker主机，可以使用现有知名的系统：

- TOMOYO
- AppArmor
- SELinux
- GRSEC, etc. 如果你的发行版本附带了Docker容器的安全模块，就可以直接使用。比如，装载了AppArmor模板的Docker和Red Hat自带SELinux策略的Docker

安全性

容器技术正是通过用户空间隔离以及权限控制来提供安全性的。总结如下： - 通过kernel namespaces进行隔离 - 附加的安全层 - TOMOYO - AppArmor - SELinux - GRSEC - 每个容器拥有自己的网络栈 - 资源限制的控制组

挑战

Docker作为Runtime对于网络和存储等基本功能的支持尚不完整。作为虚拟化技术在隔离和安全性上有待改进。目前来说，真正生产环境下大规模应用的部署及管理才刚刚起步。使用Docker对于单一容器、单一主机诸如构建、部署及运行的操作十分方便，这也是Docker迅速崛起、极受欢迎的原因之一。然而，Docker对于生产环境下大规模应用部署的支持几乎为空白。

Containers及大力支持它的厂商Docker势必要克服一些潜在的问题，才能得到大众的采用。正如最初设想的那样，Docker原本对单一计算机上运行的应用程序而言是一种格式化引擎。一个容器或一系列关联容器仍会一起驻留在单一主机上。要是应用程序需要10台服务器、100台服务器、乃至1000台服务器，哪会怎样? Docker仍需要考虑在一个计算机上运行的一个应用程序，但是大型企业IT部门(想想跨国银行、能源公司、汽车厂商和零售商)想要知道一种工具能处理庞大规模。

谷歌解决这个问题的办法就是，建立了自己的多容器管理系统，该系统的核心现在成为了开源Kubernetes项目。Kubernetes让IT部门可以建立运行容器的集群，提供了联网功能和容器命名系统，让容器管理员可以同时管理多得多的容器。它让容器管理员可以跨诸多计算机运行多个容器中的大型　应用程序，不需要知道容器集群管理的所有细枝末节。 该项目只是自去年7月以来才在开发开源代码，所以还有一段路要走。但值得注意的是，谷歌将运行其业务所特有的代码放到了公众领域。IBM、红帽和微软还有几十家主攻容器运营的初创企业对此极有兴趣。VMware同样对Kubernetes有兴趣，而且热衷于跟Docker合作，

确保其虚拟化环境与容器很好地协同运行。企业用户最终会看到这一切活动的成效。

协调大规模Docker容器应用

问题

对于Docker未来部署规模达到万级别后，还有很多技术难题有待解决

网络问题

万级别的容器部署，势必会挑战现有的网络基础设施，交换机的转发表项会遇到瓶颈。腾讯万台规模的Docker容器网络问题的解决方法： 网卡及交换链路的带宽资源是有限的。如果某个作业不受限制产生过量的网络流量，必然会挤占其它作业的网络带宽和响应时延。因此Gaia将网络和CPU、内存一样，作为一种资源维度纳入统一管理。业务在提交应用时指定自己的网络IO需求，我们使用TC（Traffic Control）+ cgroups实现网络出带宽控制，通过修改内核，增加网络入带宽的控制。具体的控制目标有：

- 在某个cgroup网络繁忙时，能保证其设定配额不会被其他cgroup挤占；
- 在某个cgroup没有用满其配额时，其他cgroup可以自动使用其空闲的部分带宽；
- 在多个cgroup分享其他cgroup的空闲带宽时，优先级高的优先； 优先级相同时，配额大的占用多，配额小的占用少；
- 尽量减少为了流控而主动丢包。

向后兼容性不够

该领域的快速创新虽然是一个优势，但是也存在缺点。其中之一是向后兼容性差。在多数情况下，我们遇到的问题主要是是命令行语法、API的改变，从产品角度来说这不是一个严重的问题。但在某些情况下，它影响了操作性能。例如，在任何启动容器后引发的Docker错误，要解析STDERR并根据错误类型进行响应（例如重试）。非常不幸的是，错误的输出格式随着版本不同变化，不得不在不断变化的结果中调试。

前景

容器已在大规模环境证明了自身的价值，比如在谷歌搜索中。

谷歌搜索(Google Search)是世界上实施Linux容器的最庞大系统，谷歌将它用于内部运营。谷歌还擅长将容器托管在其应用引擎(App Engine)或计算引擎(Compute Engine)服务中，但是与其他云供应商一样，谷歌将来自不同客户的容器放入到单独的KVM虚拟机中，因为虚拟机之间的界限更清晰。 然而想运行谷歌搜索操作，它自己使用容器，每秒启动约7000个容器，每周就相当于约20亿个。谷歌支持世界各地不同地方的搜索，在每个数据中心的活动量有高有低，这取决于时间段和影响所在地区的事件。容器是谷歌搜索引擎确保速度快、顺畅运营的秘诀之一。这种例子只会促进世人对容器的兴趣越来越大。

大规模Docker集群实践－微博春晚红包：

微博平台Docker集群的规模情况：

```
- Docker集群规模达到1000+节点
- QPS峰值达到800K/s
- 4个9的服务SLA达到150ms
- 共覆盖23个核心服务
- 春晚共调度近300节点完成动态扩容
```

面对这种流量峰值，传统按照业务峰值部署集群的方式，设备成本将无法接受。所以平台需要一种能够在集群间快速调度业务的技术方案。为什么原有的集群管理方式，无法实现快速业务切换呢，关键问题是环境的差异性。虚拟化可以实现隔离软件运行环境差异性。目前虚拟化技术有以OpenStack为代表传统VM技术，和以Docker为代表的Container技术两大类。比较如下：

	Docker	OpenStack	结论
启动速度	秒级	分钟级	面对流量峰值，速度就是一切
复杂度	基于内核的namespace技术，对现有基础设施的侵入较少	部署复杂度较高，并且很多基础设施不兼容	因为平台是对已有的线上生产系统进行改造，必须选择侵入性较小的容器化技术
执行性能	在内核中实现，所以性能几乎与原生一致	对比内核级实现，性能较差	微博核心业务对服务SLA要求非常苛刻
可控性	依赖简单，与进程无本质区别	依赖复杂，并且存在跨部门问题	生产系统集群可控性是核心竞争力能力
体积	与业务代码发布版本大小相当，MB级别	GB级别	当集群大规模部署时，体积小就代表更大的并发调度量

可以看出，Docker是目前能够实现这一目的的最佳方案。

基于Docker的云雀云平台的构件：

> – 从产品角度来说，Docker提供的管理功能只限于单机或少数的几台机器，并没有大规模管理的工具，包括网络、存储等都是空缺，云雀从一个云平台的角度完善了Docker的服务，使得它能够真正的被用于生产环境。
> – 比起基于VM的云平台，首先我们基于Docker技术整合了从测试、集成、部署、运营等整个后端服务的开发流程，大大提高了开发效率。其次，Docker的特性决定了它的部署速度比VM提高了一个数量级，这就使得基于Docker的云服务更加弹性，真正的做到大规模应用的秒级部署。最后，Docker通过提高资源利用率，增加了密度，降低了成本。

如何复用

应用场景： Docker对于应用依赖封装完整，同一镜像可重复的在测试、集成、生产等环境部署，做到“一次构建，处处运行”，适用于持续集成、持续部署流程。容器技术所提供的轻量级虚拟化适用于云端微服务架构，实现秒级部署、弹性扩展，并更高效的利用计算资源。Docker镜像正逐渐成为应用交付的标准，加之迅速成长的生态系统，将会是应用发布、分享的首选方式。具体来说，有以下几个方面：

- 简化配置

虚拟机的最大好处是能在你的硬件设施上运行各种配置不一样的平台（软件、系统），Docker在降低额外开销的情况下提供了同样的功能。它能让你将运行环境和配置放在代码中然后部署，同一个Docker的配置可以在不同的环境中使用，这样就降低了硬件要求和应用环境之间耦合度。

- 代码流水线（Code Pipeline）管理

前一个场景对于管理代码的流水线起到了很大的帮助。代码从开发者的机器到最终在生产环境上的部署，需要经过很多的中间环境。而每一个中间环境都有自己微小的差别，Docker给应用提供了一个从开发到上线均一致的环境，让代码的流水线变得简单不少。

- 提高开发效率

这就带来了一些额外的好处：Docker能提升开发者的开发效率。如果你想看一个详细一点的例子，可以参考Aater在DevOpsDays Austin 2014 大会或者是DockerCon上的演讲。

不同的开发环境中，我们都想把两件事做好。一是我们想让开发环境尽量贴近生产环境，二是我们想快速搭建开发环境。

理想状态中，要达到第一个目标，我们需要将每一个服务都跑在独立的虚拟机中以便监控生产环境中服务的运行状态。然而，我们却不想每次都需要网络连接，每次重新编译的时候远程连接上去特别麻烦。这就是Docker做的特别好的地方，开发环境的机器通常内存比较小，之前使用虚拟的时候，我们经常需要为 开发环境的机器加内存，而现在Docker可以轻易的让几十个服务在Docker中跑起来。

- 隔离应用

有很多种原因会让你选择在一个机器上运行不同的应用，比如之前提到的提高开发效率的场景等。

我们经常需要考虑两点，一是因为要降低成本而进行服务器整合，二是将一个整体式的应用拆分成松耦合的单个服务（译者注：微服务架构）。如果你想了解为什么松耦合的应用这么重要，请参考Steve Yege的这篇论文，文中将Google和亚马逊做了比较。

- 整合服务器

正如通过虚拟机来整合多个应用，Docker隔离应用的能力使得Docker可以整合

多个服务器以降低成本。由于没有多个操作系统的内存占用，以及能在多个实例之间共享没有使用的内存，Docker可以比虚拟机提供更好的服务器整合解决方案。

- 调试能力

Docker提供了很多的工具，这些工具不一定只是针对容器，但是却适用于容器。它们提供了很多的功能，包括可以为容器设置检查点、设置版本和查看两个容器之间的差别，这些特性可以帮助调试Bug。你可以在《Docker拯救世界》的文章中找到这一点的例证。

- 多租户环境

另外一个Docker有意思的使用场景是在多租户的应用中，它可以避免关键应用的重写。我们一个特别的关于这个场景的例子是为IoT（译者注：物联网）的应用开发一个快速、易用的多租户环境。这种多租户的基本代码非常复杂，很难处理，重新规划这样一个应用不但消耗时间，也浪费金钱。

使用Docker，可以为每一个租户的应用层的多个实例创建隔离的环境，这不仅简单而且成本低廉，当然这一切得益于Docker环境的启动速度和其高效的diff命令。

你可以在这里了解关于此场景的更多信息。

- 快速部署

在虚拟机之前，引入新的硬件资源需要消耗几天的时间。Docker的虚拟化技术将这个时间降到了几分钟，Docker只是创建一个容器进程而无需启动操作系统，这个过程只需要秒级的时间。这正是Google和Facebook都看重的特性。

你可以在数据中心创建销毁资源而无需担心重新启动带来的开销。通常数据中心的资源利用率只有30%，通过使用Docker并进行有效的资源分配可以提高资源的利用率。

参考文献

- 云计算格局：容器技术在云中如何定位 http://www.chinacloud.cn/show.aspx?id=19352&cid=14
- docker官网：https://www.docker.com
- docker中文社区：http://www.docker.org.cn

- 8种使用docker的方式：http://blog.flux7.com/blogs/docker/8-ways-to-use-docker-in-the-real-world
- 开源中国社区docker: http://www.oschina.net/p/docker
- 容器详解：你不可不知的九个基本事实: http://network.51cto.com/art/201507/484974.htm