

电子与信息工程学院

毕业实习调研报告

——docker技术在云计算行业的应用调查

班号：1401311

学号：1140131110

姓名：张昊民

指导教师（签章）：吕志峰

|  |
| --- |
| 2017～2018年度第一学期 |

**目 录**

[1.前言 1](#_Toc492729887)

[2.云计算 1](#_Toc492729888)

[3.Docker 2](#_Toc492729889)

[4.Docker与云计算 3](#_Toc492729890)

[5.Docker改变了什么 4](#_Toc492729891)

[6.Docker的优点 4](#_Toc492729892)

[7.总结 5](#_Toc492729893)

docker技术在云计算行业的应用调查

# 1.前言

随着企业信息化建设的推动，企业信息管理系统和业务系统之间进行数据传输和通信的需求不断加强。与此同时，随着企业发展的壮大，企业内部数据突飞猛进的增长，形成海量数据，企业必须通过强大的计算力来存储、处理和传输这些数据。云计算是近几年出现的一种新型计算模式，它的计算能力和处理数据的能力极大的满足了当今企业的需求，并且能够节约存储和计算数据的成本，为企业在硬件方面节省了大量的投资。

另一方面，由于制造业网络化和全球化进程的加快，现代企业最重要的信息管理平台——企业资源计划（ERP）系统也面临着严重的挑战。传统ERP系统是通过企业的管理模式和业务模型共同驱动下的由多个模块构成的信息管理系统，信息管理系统一旦形成，其结构相对固化、应用比较复杂、柔性差，无法及时有效地跟进企业规模扩大带来的业务以及管理创新的脚步，也无法对企业信息量激增进行有效地处理，具有明显的滞后性。为此，建立一个新型高效的信息系统应用架构将是解决上述问题的关键技术核心。

# 2.云计算

　　云计算（CloudComputing）是基于互联网实现的超级计算机模式。它能够根据用户的需求，通过互联网提供计算资源环境和实现服务，同时能够平衡负载资源，有效调节资源使用情况。云计算不仅仅带来技术上的革新，同时也带来了企业商业模式上的革新。

　　云计算从使用范围上划分主要包括狭义云计算和广义云计算。1）广义云计算通过计算机网络以按需使用服务的方式来为用户提供服务。它所提供的服务可以和互联网及软件相关，也可以是无关的，提供其他类型服务。我们通常所说云计算均是广义云计算，它的使用范围是最广的。2）狭义云计算通过计算机网络以按需使用资源的方式来为用户提供所需要的各种资源，主要包括软件、硬件以及平台等资源，它是基础设施的使用和交付。“云”是计算机网络的一种，为用户提供资源。用户只需要根据自己的需要租用“云”中的各种资源即可，一旦资源紧缺，能随时进行扩展，一旦充裕，能随时进行释放，按需使用。云计算从服务范围上来划分主要包括公有云和私有云。1）公有云主要是指第三方提供给客户使用的云平台，公有云一般是通过互联网来进行访问，以低廉的价格，租给客户进行使用。公有云作为一个支撑平台，能够整合上游的服务提供者和下游的最终用户，打造新的价值链和运营系统。2）私有云是指为客户企业单独使用而构建的云平台，通过部署在企业内部的局域网中，在企业的基础设施上部署企业内部的信息系统，从而实现企业云的安全平台以及对数据的安全性和服务质量达到最有效的控制。

总之，云计算通过对网络中的软硬件资源进行调度，以冗余存储的方式来确保系统的可靠性和可用性，通过虚拟化技术将海量数据或计算处理程序自动拆分成多个较小的相互间耦合性比较低的子数据或子程序，然后将这些子程序或子数据再通过计算机机群组成的云计算平台系统进行分布式计算程序处理，最终把结果以快速、可靠、透明的方式返回给使用者的新型商业计算模式。

# 3.Docker

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。Docker 是 PaaS 提供商 dotCloud 开源的一个基于 LXC 的高级容器引擎，源代码托管在 Github 上, 基于go语言并遵从Apache2.0协议开源。Docker自2013年以来非常火热，无论是从 github 上的代码活跃度，还是Redhat在RHEL6.5中集成对Docker的支持, 就连 Google 的 Compute Engine 也支持 docker 在其之上运行。一款开源软件能否在商业上成功，很大程度上依赖三件事 - 成功的 user case(用例), 活跃的社区和一个好故事。 dotCloud 自家的 PaaS 产品建立在docker之上，长期维护且有大量的用户，社区也十分活跃，接下来我们看看docker的故事。环境管理复杂 - 从各种OS到各种中间件到各种app, 一款产品能够成功作为开发者需要关心的东西太多，且难于管理，这个问题几乎在所有现代IT相关行业都需要面对。

云计算时代的到来 - AWS的成功, 引导开发者将应用转移到 cloud 上, 解决了硬件管理的问题，然而中间件相关的问题依然存在 (所以openstack HEAT和 AWS cloudformation 都着力解决这个问题)。开发者思路变化提供了可能性。虚拟化手段的变化 - cloud 时代采用标配硬件来降低成本，采用虚拟化手段来满足用户按需使用的需求以及保证可用性和隔离性。然而无论是KVM还是Xen在 docker 看来,都在浪费资源，因为用户需要的是高效运行环境而非OS, GuestOS既浪费资源又难于管理, 更加轻量级的LXC更加灵活和快速。LXC的移动性 - LXC在 linux 2.6 的 kernel 里就已经存在了，但是其设计之初并非为云计算考虑的，缺少标准化的描述手段和容器的可迁移性，决定其构建出的环境难于迁移和标准化管理(相对于KVM之类image和snapshot的概念)。docker 就在这个问题上做出实质性的革新。这是docker最独特的地方。

VM技术和容器技术对比，VM技术和容器技术对比。面对上述几个问题，docker设想是交付运行环境如同海运，OS如同一个货轮，每一个在OS基础上的软件都如同一个集装箱，用户可以通过标准化手段自由组装运行环境，同时集装箱的内容可以由用户自定义，也可以由专业人员制造。这样，交付一个软件，就是一系列标准化组件的集合的交付，如同乐高积木，用户只需要选择合适的积木组合，并且在最顶端署上自己的名字(最后个标准化组件是用户的app)。这也就是基于docker的PaaS产品的原型。

# 4.Docker与云计算

云计算是如今互联网的前沿技术，而虚拟化技术是云计算系统的核心组成部分之一，是将各种计算及存储资源充分整合和高效利用的关键技术。虚拟化是为某些对象创造的虚拟化（相对于真实）版本，比如操作系统，计算机你系统，存储设备和网络资源，如今有很多虚拟化技术，例如主流的RedHat KVM，VmWare ESX等，相比与这些虚拟化技术，Docker有很多自己的优势，本文将通过具体的实例对Docker在云计算中的应用场景以及它的长处进行进一步的阐述。云计算（Cloud Computer）是由分布式计算，并行处理，网格计算发展来的，是一种新兴的商业计算模式。目前对云计算还没有普遍一致的定义。中国网格计算、云计算专家刘鹏给出如下定义：“云计算将计算任务分布在大量计算机构成的资源池上，使各种应用系统能够根据需要获取计算力、存储空间和各种软件服务”。也就是说，云计算的“云“就是存在于互联网上的服务器集群上的资源，它包括硬件资源(服务器、存储器、CPU等）和软件资源（如应用软件、集成开发环境等），本地计算机只需要通过互联网发送一个需求信息，远端就会有成千上万的计算机为你提供需要的资源并将结果返回本地计算机，这样，本地计算机几乎不需要做什么，所有的处理都在云计算提供上所提供的计算机集群来完成。而虚拟化技术就是云计算技术落地的关键技术，通过虚拟化技术，可以实现云计算系统的部署，使云计算成为真正的服务。 虚拟化是指计算机元件在虚拟的基础上而不是真实的基础上运行。虚拟化技术 可以扩大硬件的容量，简化软件的重新配置过程。CPU的虚拟化技术可以单CPU模拟多CPU并行，允许一个平台同时运行多个操作系统，并且应用程序都可以在相互独立的空间内运行而互不影响，从而显著提高计算机的工作效率，传统的虚拟化软件例如VmWare ESX，RedHat KVM等已经在云计算领域中应用很多了。

Docker使用了基于操作系统的虚拟化技术，它使用命名空间技术区分不同的容器，当运行一个容器时，容器内部的进程、网络接口、进程间通信资源、挂载点以及系统内核等都在该容器的命名空间内部，这样保证了container的隔离性。 为了保证各个每个container的可度量/可配额性，Docker采用了Control Groups。这样就可以保证将硬件资源公平的分享给container，同事设置相关限制，入CPU核心，优先级以及容器的最大内存。cgroups 的使用非常简单，提供类似文件的接口，在 /cgroup目录下新建一个文件夹即可新建一个group，在此文件夹中新建task文件，并将pid写入该文件，即可实现对该进程的资源控制。 groups可以限制blkio、cpu、cpuacct、cpuset、devices、freezer、memory、net\_cls、ns九大子系统的资源，以下是每个子系统的详细说明： 1. blkio 这个子系统设置限制每个块设备的输入输出控制。例如:磁盘，光盘以及usb等等。 2. cpu 这个子系统使用调度程序为cgroup任务提供cpu的访问。 3. cpuacct 产生cgroup任务的cpu资源报告。 4. cpuset 如果是多核心的cpu，这个子系统会为cgroup任务分配单独的cpu和内存。 5. devices 允许或拒绝cgroup任务对设备的访问。 6. freezer 暂停和恢复cgroup任务。 7. memory 设置每个cgroup的内存限制以及产生内存资源报告。 8. net\_cls 标记每个网络包以供cgroup方便使用。。 同时Docker使用UnionFS来为容器提供会计设备，UnionFS是一个堆栈式的文件系统，可以把多个目录的内容整合在一起，而目录的物理分布是离散的，这样保证了Docker的可移动性，基于此方法可以在os镜像不变的基础上允许用户在其上进行一些写操作。

Docker作为云计算的虚拟化技术，安全性也是不可缺少的，但是安全永远是相对的，这里有三个方面可以考虑Docker的安全特性： 1. 由kernel namespaces和cgroups实现的Linux系统固有的安全标准; 2. Docker Deamon的安全接口; 3. Linux本身的安全加固解决方案,类如AppArmor, SELinux; 4 云计算体系下的Docker实例 4.1 云计算下Docker的使用 Docker使我们不必要像使用一般PaaS一样在充满着限制的条件下开发应用，可以就如同平常一样，自由的使用各种资源。要使用Docker，需要在机器上安装Docker Engine： 1. 创建一个Container。实际上是一个Linux Container，Docker会将网络，存储这 些事情都配置好。

# 5.Docker改变了什么

docker出现之前的部署情况。在docker出现之前，比如说要部署一个django应用，要做哪些事情？首先得有个python环境，比如这个要部署的应用基于python3，而你机器上是python2，那ok，先装个python3吧，一看装起来还挺麻烦，要先装各种依赖，还要解决一些可能的冲突，没办法硬着头皮上吧。装完python之后，因为有pip这些神奇的工具，很快就装完django及各种需要的python库了。咦，发现还要装mysql，还用了redis。没办法，继续下载，安装，配置。费了九牛二虎之力终于搞完了。一天就这么过去了。啥？你告诉我原来的服务器不用了，要换一台服务器？我靠，那重新来一遍吧，有了昨天的经验，只用了大半天就搞定了。啥？你说咱们的应用做的太好，要进行推广，需要指导其他厂商部署？我选择狗带，删代码走人上面的描述可能有些夸张，但也绝不是罕有发生。在docker出现之前，各种安装、配置环境正是运维人员经常做的事情之一，在重复工作上浪费了巨大的资源。

docker出现之后。标准的交付件说到docker最像集装箱的地方，关键就是理解它是软件领域的一种「标准化」，这种标准化的具体产物，简单来说就是镜像（image）。在docker中镜像是指，把你的应用按照一定的格式封装（其实就是执行一些符合特定规的命令行）成一种具有某种标准规格的东西（就像集装箱把你的货物封装起来类似）。形象的说，就是把你的应用按照一定的格式抽象的画了个画像。在docker中，镜像是无法直接运行的，我猜想这并不是技术上的原因，而是出于工程设计上的考虑。因为一般来说，一个软件的某个具体版本只会打包成一个镜像。如果镜像可以配置，运行的话，在使用过程中很可能会对镜像造成破坏。那怎么样避免镜像损坏的问题呢？就是再加一层，相当于分身术，只要本尊没问题，分身怎么扑街都不会真正的跪掉。多加的这一层分身，就叫容器（container），这个名字也挺形象，它就像个盒子一样，你的应用在里面运行，而且多了一层安全机制。你想使用服务或把你的应用跑起来的话，只需要使用镜像新创建一个容器就可以了（也是一条命令搞定），而镜像还放在那里不动，没办法，金贵嘛。

# 6.Docker的优点

docker在部署过程中，将安装，配置等重复的部分，由docker自动化完成。只需要在第一次部署时，构建完可用的docker镜像（装好集装箱），在以后使用中，短短的几行命令，就可以直接拉取镜像，根据这个镜像创建出一个容器，把服务跑起来了。所需要的仅仅是安装了docker的服务器，一个Dockerfile文件，以及比较流畅的网络而已。真可谓『一次构建，到处部署』。需要python3环境？直接 from python:3.x 搞定。需要迁移服务器？ 直接把应用连带Dockerfile，备份数据拷贝到新服务器上，几条命令又搞定需要作为服务给别人使用？Dockerfile即是最清晰的部署文档，维护一个官方镜像即可，谁需要就直接拉下来几条命令部署上就行了。到这里你可能已经发现了，docker镜像成为了一种像集装箱那样的标准货件。它不像传统的软件交付方式那样，只把代码以及说明文档之类的给你就完了，而是直接给你一个标准docker货件，它可能是Dockerfile，或者直接就是镜像，这个标准件不仅包括了代码本身，还包括了代码运行的OS等各种整体环境。于是，谁想用我的服务，直接拉取镜像，实例化一个容器就可以了，能直接提供你所要的服务，不再像之前那样有繁复的安装过程————这些都有人给你做过了。

当然docker的优点不止于此就像集装箱带来的标准化，这种标准化不仅是指集装箱本身，而且是指包括运输器械，物流管理方法等在内的，整个领域的标准化和效率的提高。也就是说基于一件核心事物的标准化，可以做更多的事情，比如集装箱的机械自动搬运，再远的比如自行车上的螺丝，轮胎等都有全球通用的标准。docker也是类似的，一旦这种软件标准建立起来之后，就可以基于标准件和相应的管理方式带来更多的改变。随便举一些例子： 统一的管理服务使用docker部署的应用，都会在docker的管理范围之内。这也是docker的另一个非常大的优点，它提供了一种隔离的空间，把服务器上的零散的部署应用集中起来进行管理。举个例子，比如我一个服务器上部署了n多服务，有mysql，redis，rabbitmq，其他还有一堆应用。有一天我服务器突然断电重启了，那些没有设置自动重启的应用，那些重启出问题的应用，那些你甚至都不知道隐藏在某个角落里的重要应用没启动成功….然而使用docker，一眼就可以看出那些应用正常启动了，那些应用又出问题了。接下来只用有条不紊的处理就ok了。

# 7.总结

　　很多人说docker改变了运维世界，这句话是从群体统计角度来说的，像mysql，python这样被大规模使用的应用，docker化之后对整个群体所节省的时间，是非常巨大的。所以有人可能会问，我只有一台服务器，也不太可能会迁移。我的python服务，mysql服务，只需要部署一次，就可以在以后重复使用了。那这样docker对于我来说还有优势吗？毕竟docker也是有学习成本的。如果你确信你的应用都是一次性的，而且只提供给自己使用，那么docker在这种场景下的优势不是特别明显：即便是docker，最初的构建也是需要有人做的，这和直接在机器上部署一次的工作量差不多。也就是说，docker并不能把部署的工作减少为0，比较好的情况下是基本减少为1。但是，你真的真的确信，你所做的工作只是一次性的吗？