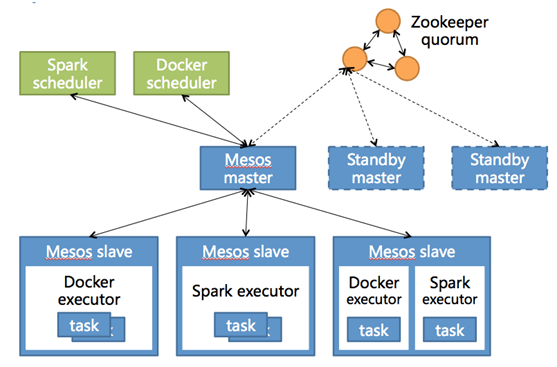
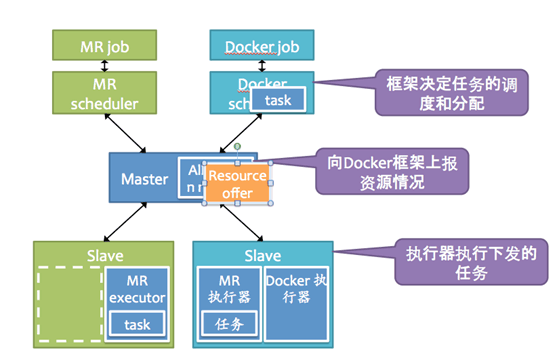
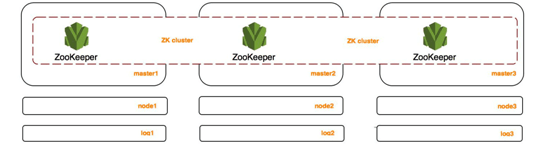
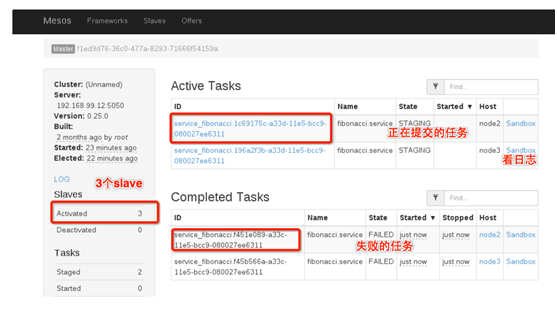
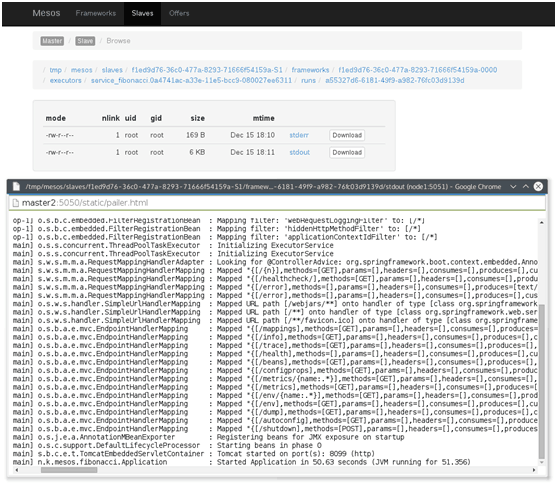
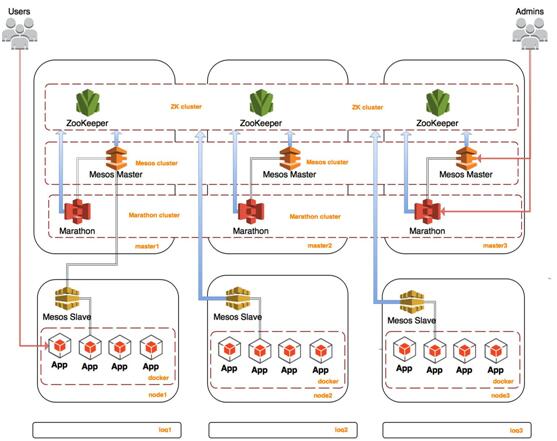
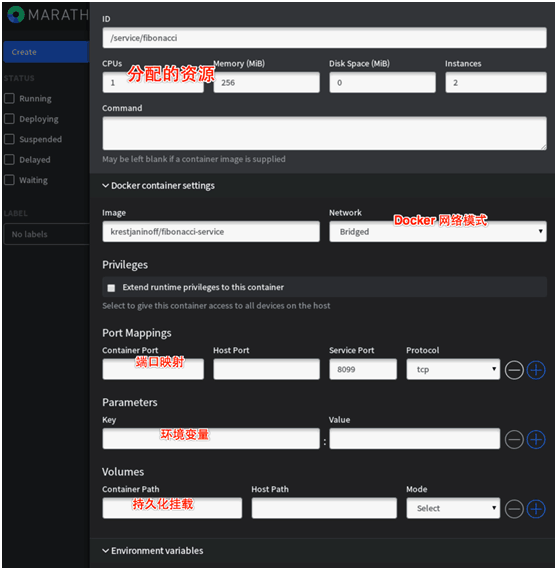
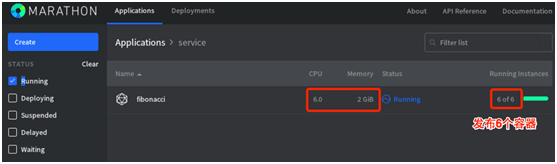
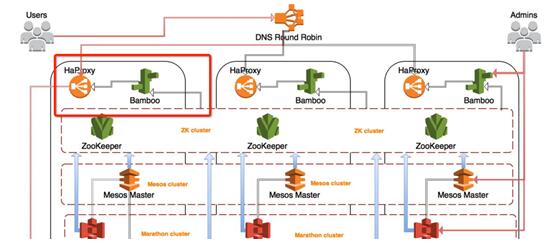
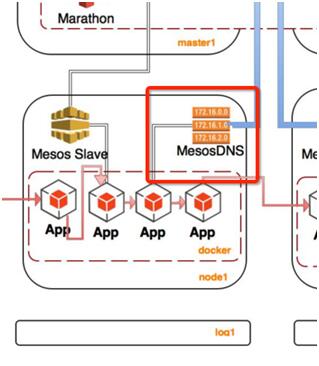
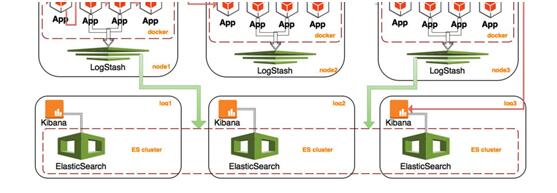
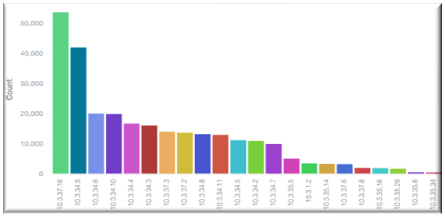
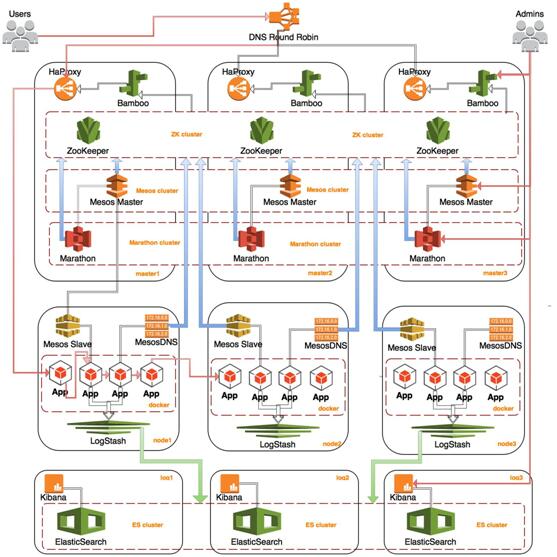
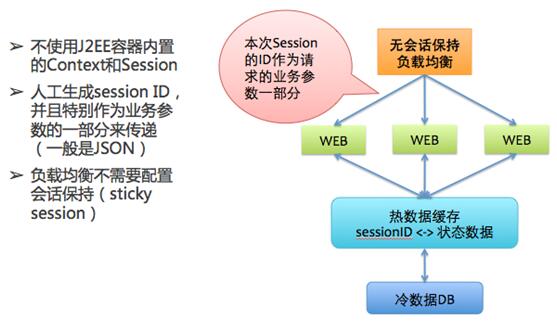
Mesos 在企业生产中的应用

分享嘉宾，数人云COO 谢乐冰

在德国工作十年，回国后加入惠普电信运营商部门，拥有多年项目经验和创业公司工作经验。在数人云负责产品售前和运营，专注行业的技术应用领域，为金融、电信、电商等行业提供服务。

北京-数人云-转播员(457353903) 2016/3/17 20:01:44  
之前两次数人云CEO王璞和CTO老肖分别介绍了微服务架构和Docker的来龙去脉。首先我还是费点时间把思路和各位运维同学捋一下，“容器”到底能解决什么问题，为什么一个如此单薄的小技术点能够引起行业这么大的反响。  
回顾Java的发展轨迹看容器技术  
因为我自己写了十几年的Java，经常把容器和十年前的Java做比较。一个公司说自己是做“Java”的，实际上涵义是背后一整套企业IT基础架构。软件一般都是各个集成商（东软、文思）大量码农兄弟们开发，主要还是用Windows。打成了WAR、EAR包之后交付给甲方，就可以在Linux环境下跑起来。同样Weblogic WAS这些中间件在底层计算集群之上，实现了企业服务的大规模运行。  
北京-数人云-转播员(457353903) 2016/3/17 20:03:52  
中间件之下是IOE昂贵的高性能硬件，虽然也是集群化，主要依靠Scale up来提升性能。虽然中间件理论上实现了应用和硬件资源解耦，但实际上依然对硬件有非常苛刻的要求，特别是跑数据库的部分。

北京-数人云-转播员(457353903) 2016/3/17 20:05:22  
整个架构为了向上实现SOA的架构，虽然现实中90%以上顶多做到了“面向对象”，但并不妨碍Java（J2EE）作为企业服务交付的“通用”形式，成为了开发单位和运行单位共同接受的标准。所以Java背后代表的不仅仅是一个语言，还是一个完整的IT基础架构和产业链 —— 昂贵的高性能硬件、闭源的中间件软件、Java作为交付接口、SOA架构和开发与运维分离的模式。  
北京-数人云-转播员(457353903) 2016/3/17 20:07:58  
背后还有两个隐性的英雄，一个是北大青鸟这样的培训机构，大量产出Java程序员。另外就是Oracle数据库，当然这些年轻程序员写的代码效率不太高的时候，全靠数据库来救场了。当然这一切还是传统的企业业务决定的，例如ERP、CRM等等，并发较低、强事物性和强一致性、逻辑和关联关系复杂。  
北京-数人云-转播员(457353903) 2016/3/17 20:08:59  
  
北京-数人云-转播员(457353903) 2016/3/17 20:10:10  
如今时代发展到了蓝色的部分，云计算时代的IT架构底层不再是几台小机或者一堆刀片，更多的是企业私有云甚至是公有云，IaaS实现了资源层管理的自动化和标准化。  
北京-数人云-转播员(457353903) 2016/3/17 20:10:47  
容器就像当年的Java一样，成为了开发和运维共同认可的接口。容器成为了应用上“云”的标准交付方式，不管是Java、Python还是C，只要用Docker打包，就可以丢到这个那个“云”上跑起来。  
当然在底层各种计算资源（公有云、私有云甚至物理机）之间，也需要一个中间件来作为容器的大规模运行环境。下面是成千上万的主机，上面是乌央央的容器，中间的云计算中间件实现了两者的解耦。  
北京-数人云-转播员(457353903) 2016/3/17 20:12:15  
上面支撑的软件架构是“微服务”架构，就像当年的SOA。整体上也是实践了Devops一套运维开发方式。  
就像传统中间件包括了运行环境、消息队列、ESB（服务发现）和数据抽象等等，云计算中间件也都有类似的服务，例如Mesos、K8s这些容器运行环境，就对应着跑EJB的Weblogic Application Server。  
总之“容器”背后不是单个技术，而是完整的以开源软件为主的云计算IT基础架构和相应的开发和运维流程。当年虚拟机出现让大家尝到一点点云的滋味，但是毕竟局限于资源层，对开发、业务和软件架构没有影响。今天容器影响这么大，大家终于成为了应用上云的突破口，将对大家未来的职业生涯产生巨大的影响。就像今天很难招聘到懂EJB的大学毕业生，过两年很快容器和背后的互联网开源技术栈就会成为主流。  
北京-数人云-转播员(457353903) 2016/3/17 20:14:11  
下面我们说说第二个点，有关Mesos与K8s的老生常谈  
言归正传，下面我来一步一步地介绍Mesos的实战。说起Mesos，大家往往第一个问题是Mesos和K8s有啥区别，哪个更好。我觉得这两个就像iOS和安卓，已经成为了新一代轻量调度框架的主流。两者都是源于Google的Borg，但Google自己没有使用任何一个。K8s胜在开发者多，用Go语言开发，社区活跃。Mesos是Apache项目，已经诞生了7年，目前有过超过万台规模的部署。  
北京-数人云-转播员(457353903) 2016/3/17 20:15:28  
总体上我们认为Mesos比较适合目前阶段的大规模生产环境部署，K8s目前还处于快速更新的阶段，两者都有很好的未来。当然Mesos也能兼容大数据等框架，未来目标是逐步把各种集群化的应用（Kafka集群例如）都搬到Mesos上来，实现一键安装和自动扩展。  
下面是一点点Mesos的科普，其实市面上类似的文章已经不少，这里我特别推荐平安科技余何老师的《PaaS实现与运维管理：基于Mesos +Docker+ELK的实战指南》，内容非常详细。  
北京-数人云-转播员(457353903) 2016/3/17 20:16:36  
用两句俗话说Mesos和K8s的原理，就是像使用一台电脑一样使用整个集群，类似集群的操作系统。单机的操作系统是管理单机的计算、存储和IO，集群操作系统是管理管理一堆机器的资源。目前聚焦在计算和内存之上，存储部分需要单独的分布式存储（例如Ceph和GlusterFS），网络需要SDN的支持。不过传统上IOE也是各管一摊了。  
  
北京-数人云-转播员(457353903) 2016/3/17 20:18:00  
原理看起来也不复杂，Mesos 在每台Slave主机上安装一个Agent，不断地把剩余资源上报到Master。报告内容类似 { (node1, <2 CPUs, 4 GB>),(node2, <3 CPUs, 2 GB>) }，这样Master就知道各个机器的剩余资源情况了，非常简单。  
Master上面有很多框架Framework，例如Docker和Spark。你就可以把他们理解为Linux里面安装的JRE和Golang、C的运行类库。你想在Mesos上跑啥“语言”，就要部署个框架，例如跑Docker的框架就是Marathon。Mesos会把整个集群的资源按照一定的算法分配给各个框架，这个就是所谓资源调度的过程。因为Slave上报资源情况是不断更新的，所以就是所谓动态资源调度。  
北京-数人云-转播员(457353903) 2016/3/17 20:20:20  
  
每个框架收到分配的资源之后，会自行决定将任务和资源匹配，然后通过Master将任务下发到Slave上执行。Slave上面有每种任务的执行器（Executor），就是运行环境。例如Docker任务的执行器是Mesos预装，其他类型任务执行器可能会实时下载。所以通过安装不同的框架+执行器，就可以支持各种“分布式”的任务系统。请注意这里说的一定是集群化的系统，如果是单点部署一个MySQL之类的就意义有限了。  
北京-数人云-转播员(457353903) 2016/3/17 20:21:55  
以管理Docker任务的Marathon框架为例，它收到了Master提供的资源之后，一个是负责进行任务调度，而且还能够通过Health Check监控任务是否还活着，发现失败就重新下发任务。  
这些都是常规性的解释，下面我们看看Mesos集群，看看如何一步步搭建。初始一般需要准备3台主机承载Master节点，任意多的Slave，这里建议也是3台。还有几台机器存放log等等。下面的一些图片来自前两天数人云公众号（dmesos）翻译的文章《初次微服务体验：从Docker容器农场说起》。  
北京-数人云-转播员(457353903) 2016/3/17 20:23:27  
  
第一步 部署Zookeeper，负责整个集群的分布式一致性，例如Master领导选举  
  
北京-数人云-转播员(457353903) 2016/3/17 20:24:11  
第二步，部署Mesos本身。我们的分布部署了3个Master，管理3个Slave节点。大家注意到，配置Mesos的时候最重要的参数就是Zookeeper，不但Master要通过ZK来进行领导选举，而且Slave也可以通过ZK来知道谁是活跃的Master.  
北京-数人云-转播员(457353903) 2016/3/17 20:25:03  
  
到这一步，理论上已经可以用Mesos来管理集群下发任务了，大家看见下图里面资源（Slave）、任务（正在执行的已经介绍的）。   
北京-数人云-转播员(457353903) 2016/3/17 20:26:14  
  
甚至还能看到该任务的Stdout输出，就和SSH进去操作一样。  
北京-数人云-转播员(457353903) 2016/3/17 20:28:04  
  
不过仅仅有Mesos，还要自己来编写框架调用接口发布任务，非常不方便。所以需要一个框架来跑容器任务，那就是马拉松（Marathon）。顾名思义用来跑各种长时间运行的服务，类似Linux里面的Inti.d，例如各种网站服务。马拉松是用Scala编写的，本身提供自己的Web管理界面，通过这个界面我们可以“遥控”Mesos来下发并保证Docker任务长久稳定执行。  
北京-数人云-转播员(457353903) 2016/3/17 20:30:09  
  
马拉松的界面也非常直接，大家看看发布Docker任务的界面，基本就是填入Docker Run后面的那些参数，然后告诉马拉松要发布多少份。马拉松会匹配每个Task和Mesos提供的资源，然后通过Mesos将任务下发下去。  
北京-数人云-转播员(457353903) 2016/3/17 20:31:13  
  
结果  
北京-数人云-转播员(457353903) 2016/3/17 20:32:45  
服务发现  
  
服务发现是个比较晦涩的翻译（Service Discovery），大概不妨粗略地理解成负载均衡算了。例如马拉松下发了100个网站的容器，每个容器有自己IP（一般是宿主机）和端口。显然前面需要挡一个负载均衡来分配流量。对外暴露的就是负载均衡的某个服务URL，后面自动将流量转发到某个容器的IP+端口上。  
  
北京-数人云-转播员(457353903) 2016/3/17 20:34:22  
我们这里用HAProxy来做负载均衡，有个服务叫Bamboo会不断从ZK读出Mesos状态并且更新HAProxy的配置文件。这样新发下来的Docker会自动添加上HAProxy，而死掉的会被移除。  
还有一直办法是用内网的DNS，这个DNS会维护现有的容器列表（IP+端口），并且返回任意一个的IP+端口，页实现了负载均衡和服务发现功能。不过目前Mesos DNS还不太成熟，我们一般用HAProxy。  
北京-数人云-转播员(457353903) 2016/3/17 20:35:30  
  
几百个Docker撒出去，绝对不可能再登到主机上去找看日志。日志必须集中收集，并且提供检索功能，才能有效的Debug。解法也不新奇，无非是ELK。请注意Docker日志可以直接从API读出，另外需要增加一些应用、主机和容器有关的Meta Data。  
北京-数人云-转播员(457353903) 2016/3/17 20:37:49  
  
此外分布式系统不能没有监控，黑盒子等于无法运行，所以监控要分为如下三个层面。  
- 主机监控：这个并非Mesos的关注点，因为主机是资源层，本身也有自己的监控体系  
- 容器层面的监控，主要是用cAdvisor，包括CPU、内存和IO  
- 最最重要的是应用层监控，因为PaaS本身对外提供服务，所以监控的关注点应该是全局最终结果和逻辑正确性，而不是太纠结于个别主机和容器的  
北京-数人云-转播员(457353903) 2016/3/17 20:39:37  
这个是分布式系统和传统系统最大的区别，关注点不再是个别容器和主机，而是业务本身。这种系统设计本来就是希望软件脱离对特定和单点硬件的依赖，通过集群化实现大规模系统的高性能和高可用。所以监控不再是着眼于“源头”，而是看重效果。很多时候平台的自愈机制甚至“埋没”了底层的一些故障，那么就让他被埋没了，只要最后效果能够得到保证。  
北京-数人云-转播员(457353903) 2016/3/17 20:40:26  
分布式系统在应用层监控要求远远大于普通的IT系统，例如下面是一个HTTP返回状态吗的直方图，这样能很快发现是否出现大规模异常，并且通过日志系统来定位问题。  
  
北京-数人云-转播员(457353903) 2016/3/17 20:41:24  
分布式系统和传统IT区别，就像市场经济和计划经济一样，不是要处处完全可控有计划，在最终结果保持可控情况下，突出灵活性、自由度和弹性，支持业务多变和快速发展。  
  
北京-数人云-转播员(457353903) 2016/3/17 20:43:11  
这样一个基本的分布式系统就搭建完毕，当然如果是生产级别还需要有大规模集群运行调优、集群化HAProxy，监控和报警对接、多租户管理、F5的对接、和Openstack等等的IaaS对接等，这样就需要数人云这样的商业化开源方案来支持了。  
北京-数人云-转播员(457353903) 2016/3/17 20:44:28  
此外经常有用户问到，啥样的应用可以上云呢，下面的表格回答了这个问题。  
  
北京-数人云-转播员(457353903) 2016/3/17 20:45:48  
可以看到，这个问题的回答并不是黑白分明。最理想的当然是完全的微服务架构，可以发挥全部的作用。当然90%应用目前还是有状态应用，所以可以快速扩张，但是无法收缩，需要实现Graceful Stop功能，慢慢地收缩。所谓的无状态化改造，无非就是很标准互联网架构，不要用J2EE内置的Session就好。  
  
北京-数人云-转播员(457353903) 2016/3/17 20:48:54  
本来今天还要展示一个我们的客户案例，如何将一个分布式系统迁移到Mesos之上，因为时间关系，那就直接放一篇线下活动分享的一个实录吧，Kuick，很酷的一个团队。

http://mp.weixin.qq.com/s?\_\_biz=MzI0MTAxNzI2MQ==&mid=402560734&idx=1&sn=5711f1f6527f6dff9d9ab305f6c63900&scene=23&srcid=03172gUUiPdqAbkmoE4btEdQ#rd