

云生态

Cloud Ecosystem

专刊

第2期
Vol.02

InfoQ
CLOUD

生态圈新闻

观点&趋势

对话大咖

Docker专栏

技术热点



打造中国最优质的云生态媒体

ArchSummit

International Architect Summit

全球架构师峰会 2015

2015.7.17-18 中国·深圳·大梅沙京基海湾大酒店

侧重业务场景，引领技术趋势



HOT

9大热门技术专题

研发体系构建 电商和零售业的转型

在线教育 移动化机会

智能硬件 大数据背后的价值

开源与企业发展 企业云化的痛点与实践

互联网金融

INVITE

邀请制闭门会议

在线教育机遇与挑战

开源与企业发展

企业云化的痛点与实践

互联网基因的金融



议题提交开放，折扣购票启动，详情查阅大会官网

购票热线：010-89880682 会务咨询：arch@cn.infoq.com 大会官网：www.archsummit.com



AWSome Day

让您大有作为

合作伙伴



大连站、北京站、深圳站、上海站

[点击查看详情](#)



本刊由InfoQ中国
制作出品

出品人：霍泰稳

总策划：崔康

本期主编：魏星

流程编辑：丁晓昀

读者反馈/投稿
editors@cn.infoq.com

商务合作
sales@cn.infoq.com

目 录

卷首语《向长跑者致敬》	2
嘉宾寄语	4
热点回顾	5
对话大咖	8
篇章嵩：开放与高可用 是阿里云角力海外市场关键	9
技术热点	12
公有云故障案例分析 ——Microsoft Azure 的飞来人祸	13
Google 发布论文 披露大规模集群管理工具 Borg 的细节	16
Docker 专栏	19
Docker: 现在和未来	20
腾讯万台规模的 Docker 应用实践	27
观点&趋势	31
[观点] 运维的本质——可视化	32
私有云之殇：公有云这座大山（上）	37
生态圈新闻	41



卷首语

《向长跑者致敬》

作者：杨赛

有时候我会想，在能源行业内部人士眼中的中石油中石化、国家电网、燃气公司这些卖能源的公司，是不是跟我们这些 IT 人士眼中的亚马逊、英特尔、阿里云这些卖 IT 资源的公司有着类似的地位？非业内人士对云计算的看法，跟我们这些能源的普通用户对油气电的看法是不是也类似？行业内部的生态如何发展，对行业之外的普通用户而言几乎毫无意义；但如果是以玩资本的身份介入，那么任何行业内部的生态发展，却又变得有意义了。

生态系统 (ecosystem) 一词最早公开出现于英国学者 Arthur Tansley 发布于 1935 年的一篇论文——按那时的分类应该属于植物学范畴。Tansley 的前半生致力于植物学，是英国生态学科的重要先驱之一，然而人到中年，遭遇一战，战后的 Tansley 师从弗洛伊德，研究精神分析。1930 年代的生态学界主要关注有机生物的个体，与个体之间形成的关系，而 Tansley 则建议将有机生物所生存的物理环境也纳入生态学的考虑，研究有机生物与物理环境之间的物质交换(水、碳、能量)，将 “生态系统” 作为研究自然的基本单元。

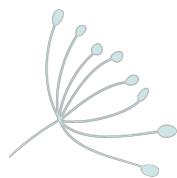
一个 “生态系统” 到底是一个真实的独立单元，还是一个学术上的抽象定义？一个单元与其他单元的分隔，是时间和空间的分隔，还是能量角度的动态分隔？命名难，定义更难，当时战后各个国家的各方各面都跟意识形态绑定，即使是一个学术名词的定义统一都成了不可能的一件事。但是人脑的神奇之处就在于，这种极度抽象的名词即使定义不统一，也并不妨碍它的使用。

于是我们今天得以用 “生态系统” 这个词来作为了解云计算行业（其实就是 IT 资源行业）的一个角度。而且从某种层面来看，我们今天对这个行业的认知和植物学家们在上世纪三十年代对生态学的认知是类似的，即，越来越多的人不仅关注 IT 堆栈各层的技术运转，也看到了资本、

能源和人才流动在这个系统中的推动作用。大体来说，是环境造就了个体。

研究生态系统，相比研究有机生物个体，似乎要来的更困难一些，研究结果看起来也更不实用——今天的“实用生态学”的主要作用是物种保护与生态环境重建，这件事目前来说在人类社会的任务列表中还没有排上很高的优先级，然而这是个有关流动性与时间周期的问题。如果一个牧羊人计划一生都在一座山里放羊，那么这座山的生态延续性一定是对他而言他至关重要的问题。而那些玩玩就跑的人，则不会觉得这是一个问题。

看一个行业，多少是认真打算长期驻扎的，多少是打算玩玩就跑的，这里的生态是否能够良好发展，从中可知一二。为此，向长跑者致敬！



嘉宾寄语

随着云计算技术的不断成熟和深化，如何在技术的基础之上深入打造价值链和生态圈变得越来越重要。对于云生态圈的建立，IBM 一直是重要的参与者和引领者，像基于 OpenStack 的 IaaS，基于 Cloud Foundry 的 Bluemix (PaaS)，都是 IBM 在着力推进，并希望和合作伙伴共同发展壮大的。InfoQ 的《云生态专刊》是一个很好的尝试，衷心希望《云生态专刊》越办越好，帮助中国的云生态发展壮大。

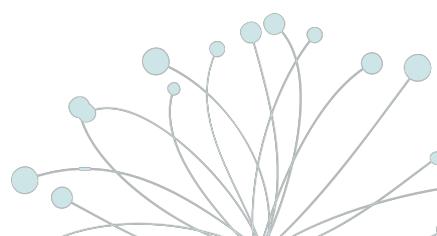
——IBM 中国开发中心 Bluemix 总经理 兰健

很高兴看到 InfoQ 出品的《云生态专刊》，立意新颖、内容扎实，是我们关注云生态的重要途径。对于云计算而言，生态系统的建设是重中之重，青云 QingCloud 也会把今年的工作重点放在培育生态系统上。近期，QingCloud 会针对云生态发布一系列激动人心的新策略、新产品，与合作伙伴一起打造开放、丰富、高效的云生态系统。

——青云 QingCloud 联合创始人&CEO 黄允松 (Richard Huang)

祝贺 InfoQ 《云生态专刊》创刊，“云”这个年轻的领域，需要专业的媒体给予更多支持与关注。云计算是个很大的概念，它几乎可以涵盖 IT 的各个不同领域，甚至可以说 IT 的各个领域进化的终极状态都可以落在云端。作为数据服务提供者，TalkingData 从创业之初就致力于将成熟的大数据能力开放出来，让大家都能从“数据”中获益，只有借助“云”，才有可能快速高效的让数据流转起来，产生更大的价值。这就需要一个健全的云生态作为基础，只有生态完整，才有可能让数据流畅快速的运转起来。希望《云生态专刊》能够用专业的视角，分享来自各行业的经验，促进云生态健康快速的发展。

——TalkingData CTO 肖文峰



热点回顾



云生态专刊

Spring Cloud 1.0 正式发布

Pivotal 于去年 6 月份公布了 Spring Cloud 1.0 源码，代码托管在 GitHub。在经过将近半年的开发和发布 3 个里程碑版本和 3 个 RC 版本后，其正式版终于发布了，目前已提供在 Maven 中央仓库和 Spring 的仓库中。

Spring Cloud 是一个基于 Spring Boot 实现的云应用开发工具，它为基于 JVM 的云应用开发中的配置管理、服务发现、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等操作提供了一种简单的开发方式。

Redis 3.0 正式版发布

近日，Redis 3.0 在经过 6 个 RC 版本后，其正式版终于发布了。3.0 版最重要特征是对 Redis 集群的支持。Redis 是一个开源、基于 C 语言、基于内存亦可持久化的高性能 NoSQL 数据库。

Salvatore Sanfilippo (Redis 之父) 对 3.0 正式版的发布这样说到：“Redis 3.0 标志着一个新阶段和新开发模式的开始，相信 (Redis 3.0) 能够完全改变 Redis 的面貌。人们将认识到 Redis 是一个全新的东西，它的自动扩展、容错和高可用性都有了很大的改进。它将能够在更大范围内承担更关键的任务。”

阿里云 Xen Hypervisor 热修复技术解析

2015 年 3 月 10 日，Xen 社区安全团队公开披露了高危漏洞 XSA-123，该漏洞可能导致一台 Guest VM 读取到其他 Guest VM 的敏感数据，这是自 Xen 诞生 10 多年来公布的 125 个漏洞报告中影响最大的两个（另一个为 XSA-108）之一。由于此漏洞对公有云服务的影响重大，各个公有云厂商分别对此漏洞进行了重启修复或热补丁修复。

阿里云团队 hotfix 的基本思路是，截获硬盘设备读文件时的 DMA 请求，并修改 DMA 目的地地址，以将补丁信息写入 Xen Hypervisor 内存来达到修复功能。经过测试、优化，3 月 9 日阿里云给用户发布公告，同时开始给线上机器批量应用补丁，到 3 月 10 日发布完毕，同日漏洞公开。

GitHub 推出大文件存储

Large File Storage (LFS) 扩展

早在 2013 年 6 月，GitHub 明确表示 Git 不适合储存大的二进制文件，用户上传 50 MB 的文件会收到警告，100 MB 以上的大文件则禁止添加或修改。然而近日，GitHub 推出了 Large File Storage (LFS) 扩展。从而改进了大文件的版本控制。

免费用户的 Git Large File Storage (LFS) 有限额，文件储存限制在 1GB，带宽每月流量限制在 1GB。用户如果需要更高的配额，可以轻松购买更大的存储和带宽。

谷歌云平台集成日志管理功能

谷歌云平台 (GCP) 最近增加了云日志 , 所有谷歌云平台客户都可以免费使用云日志 , 使用谷歌云存储和 BigQuery 时 , 将根据数据转换、存储和查询结算附加的费用。计算引擎、应用引擎、云存储、云监控和 BigQuery 构成谷歌云平台的基本构件。谷歌使用云日志把这些功能关联到一起集成了日志管理。客户可以把计算引擎和应用引擎的日志合并到云日志中去执行实时的查询和分析。日志可以和云监控相结合实现可视化。

谷歌建议按以下步骤使用云日志 : 提取、查看、探查、分析和存档。亚马逊是谷歌最有竞争力的对手 , 它通过 AWS CloudTrail 和 Amazon CloudWatch 提供了类似的能力。

Swiftype 从亚马逊 EC2 切换到真正的服务器

2012 年 ,Swiftype 创建之初选择了 Amazon Web Services 作为基础设施 (Swiftype 是一家搜索解决方案提供商 , 目前已为超过 10 万个网站和应用程序提供搜索服务。) 。但在 Amazon 云上 , 他们经常会遇到网络问题、 VM 实例宕机问题、不可预见的性能衰减。因此 , 他们决定放弃 EC2 , 转而使用真正的硬件。基于以往与托管供应商打交道的经验 , 他们决定选择 SoftLayer 。

从 EC2 切换到真正的硬件之后 , 发生了以下几个方面的变化 :

- 稳定性提升 : 严重故障次数由每周 2 到 3 次降到了每月 1 到 2 次 ;
- 性能提升 : 所有后端服务的性能都获得了提升 , IO 密集型服务 (如数据库和搜索集群) 比 CPU 密集型服务提升更明显 , 性能有了更强的可预见性 ;
- 成本降低 : 月度成本至少降低了 50% ;
- 配置灵活度提升 , 但配置时间增加了。



对话大咖

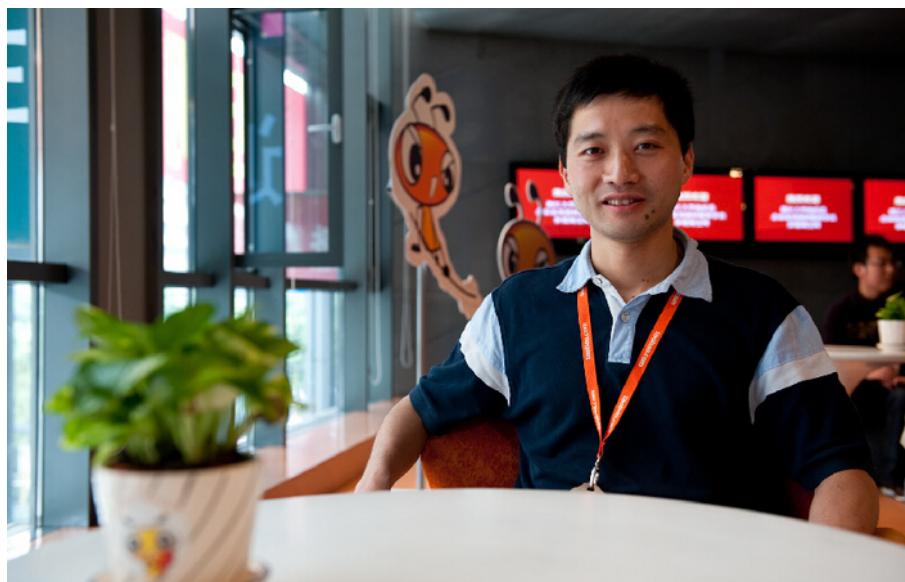
云生态专刊

InfoQ

章文嵩：开放与高可用是阿里云角力海外市场的关键

作者：魏星

2015年3月4日，阿里云宣布北美数据中心正式启用，这也意味着国内云厂商开始登上国际舞台。为此，我们采访了阿里云现任CTO章文嵩博士。



章文嵩，国防科学技术大学博士毕业，国防科技大学计算机学院副教授，LVS(Linux Virtual Server)开源软件创始人，现任阿里巴巴高级研究员，现担任阿里云CTO。

InfoQ:我们知道淘宝技术团队在运维方面的积累很多，阿里云在初期曾经历了一些波折，现在的运维与应急响应机制是怎样的？目前的系统能否达到您四个九或者五个九的要求？

章文嵩：阿里云的运维和淘宝背后是同一个团队在支撑。大家感觉到淘宝的稳定性，是基于底层和应用层都有高可用的设计，这使得底层的部分硬件和组件故障一般不会造成应用本身的不可用。而在阿里云平台上承载上百万的客户，任意故障对一个云客户造成影响，对这个客户来说就是全部。也使得故障更为可感知。阿里云在云平台内部从光纤网络、硬件到软件上都做了很多高可用的设计，最终要实现整体的高可靠性也建议客户在应用中也要做高可用设计，两者配合以达到应用五个九以上更高的可用性。

若针对每个客户可用性的均值，我们的ECS和RDS都达到五个九以上。但是我们对各个

云产品的整体可用性指标定义更为严格，不是按均值，而是按对用户造成的影响度，例如有云产品定义对 10% 用户造成影响就记为云产品整体的不可用时间。严格的度量体系可以指引我们把云产品做得更好。

InfoQ:我们知道您在淘宝的底层基础设施上做了很多改进，大大降低了运营成本。您能从开源技术、去 IOE、低功耗服务器、阿里骨干传输网以及硬件红利等几个方面谈谈阿里云在降低运营成本方面所取得的成就吗？

章文嵩：你提到这些方面我们都有很好的工作在开展，在开源技术上我们有相当的积累，例如我们在 XEN 和 KVM 有很好的技术专家，在全球云计算厂商如 Amazon、linode 等应对最近 Xen 缺陷需要重启 VM 来解决时，我们找到方法作热升级解决这个缺陷而无需重启 VM，旭卿（[张献涛，阿里云资深专家及虚拟化技术总监](#)）也将在 QCon 全球软件开发者大会上分享；在去 IOE 上，我们积累丰富的高性能服务器经验，这些经验都融入到我们 RDS 服务中；低功耗服务器上我们有 64 位 ARM 服务器已跟 ODPS 在作测试，阿里骨干传输网 ABTN 在国内互联网公司中我们运行 BGP 是最早的，服务质量应该是最好的；服务器我们在定制并在组件级作精确的供应链管理来降低云平台的成本。

但是我觉得最重要的工作是我们过去一两年对全线云产品建立的相对精确度量体系。全链路监控与实时异常分析平台可以让我们知道每个请求所对应的各个环节处理时间和以及时问波动，有机会来改进和缩短响应时间减少波动，异常分析与处理让阿里云平台更稳定，动态热点迁移和调度技术逐步提高云平台的资源复用率，这会是阿里云平台的最大竞争优势。

InfoQ:提到开源，您本人是开源技术的倡导者，淘宝从开源受益的事例(例如 TaobaoJVM)已经广为人知。阿里云在近几年的发展中有哪些受益于开源又反哺到开源的例子可以分享吗？

章文嵩：这样的例子有很多。因为淘宝和阿里集团的基础平台团队现在都合并到阿里云了，阿里云承担对外公有云和对内基本平台的职责，在阿里是一个统一的基础平台研发团队。我们受益于开源回馈开源的项目有 Linux 内核、LVS、MySQL、Tengine、jstorm，还有你所提到的 TaobaoJVM 等。最近一个很好的例子是我们 RDS 团队因为 MySQL 上的能力和贡献应邀成为 WebScaleSQL 开源项目的全球第五家公司成员。

InfoQ:我们也看到阿里云在安全方面有许多建树。您能谈谈阿里云安全相关的工作吗？

章文嵩：阿里在安全上有很多像云舒这样国内顶级的安全专家，我们把安全的技术和经验做成云盾安全产品为我们云客户提供安全服务。云盾在 2014 年 12 月 20 至 21 日为一位网游客户防住持续 14 小时的拒绝服务攻击，DDoS 峰值流量为 453.8Gbps 每秒。

InfoQ:在政府大力推广云计算的背景下，各地新建过很多大型数据中心。这些政府主导的基础设施建设对云服务企业会不会构成另一个竞争因素？您怎么看待这个问题。

章文嵩：不会构成竞争，阿里云和阿里巴巴集团（有些是和集团签署的）已经跟 12 个省/市/地区/直辖市签订战略合作。地方政府建设数据中心，我们可以输出云平台的软件和运营经验，可以联合运营云平台，达到共赢。

InfoQ:除了国内云服务商，国外的 AWS、Azure 等也一直在抢滩国内市场，你认为国内云服务商应该怎样面对这样的竞争局面？阿里云在海外市场的进展如何？

章文嵩：据我了解，阿里云规模在国内是第二名的近 10 倍，并且我们的增速依旧高于第二名。我们有二十多种云产品，产品线比较丰富，未来会有更多的阿里技术能力变成云产品对外输出。国内企业在 IT 需求上多样性和复杂性跟国外会很不一样，如何建设云的生态系统来应对这些复杂需求是大家面临的共同挑战。

我们在 3 月 4 日宣布美国硅谷云平台投入试运营，为北美和全球客户提供云服务，一个多月销量非常好。今年将会继续考虑在北美、欧洲等全球各地选址建设云平台。阿里云更多地思考是用户需求在哪里，我们就去哪里发展。比如，我们选择美国市场作为海外的第一步，就是因为很多国内客户有在美国拓展业务的需求，我们为了满足客户对云计算部署的需要，建立海外数据中心。我们同时也非常愿意和在专业领域里有建树的伙伴达成合作，共同有效地为客户提供云计算服务。

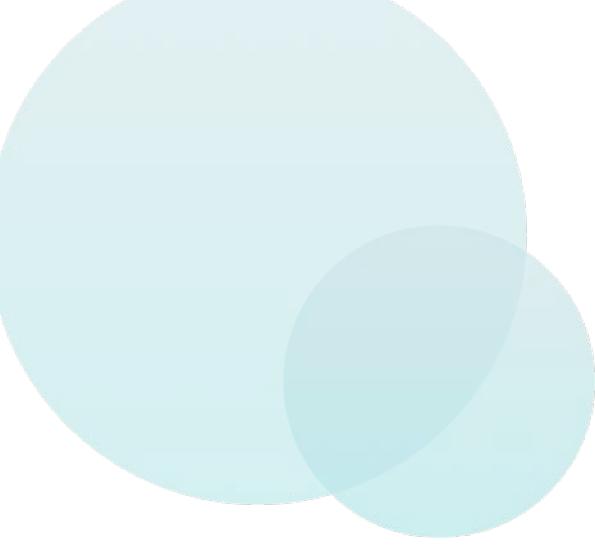
InfoQ:越来越多的创业团队把产品和服务部署到云上，阿里云对创业者的支持和服务是怎样的？您对这方面的业务有什么规划？

章文嵩：创业公司最适合部署在云上，随着访问量的增长可以在云上快速扩展。例如，前几天趣分期因为获得融资的消息，导致网站访问量有意想不到的成倍增长，当天晚上在我们云上就进行快速扩展。阿里云联合百亿资金对创业者推出了“创客+”计划，提供从开发组件、分发推广、办公场地、前后期投资到云服务资源的一系列创业扶持，帮助创客实现梦想。目前，“创客+”平台上优质项目众多，仅试运营期间，就有 141 个创客项目成功融资 2.7 亿元。

技术热点



云生态专刊



公有云故障案例分析 ——Microsoft Azure 的飞 来人祸

作者：丁雪丰

公有云早已飞入寻常百姓家，除了初创的企业，很多大公司也将自己的服务部署在共有云平台上，因此公有云的稳定性和可靠性是十分重要的。平日里谈起公有云，大家总把注意力放在行业老大 Amazon Web Services 上，不太提及 Microsoft Azure，今天就让我们来看一下去年 11 月 Azure 发生的将近 11 个小时的故障。

首先，来回顾一下故障的经过，根据 [《Update on Azure Storage Service Interruption》](#) 这篇官方博客的介绍，这起编号为 3071402 的故障名为“Microsoft Azure Service Incident : Connectivity to multiple Azure Services – Partial Service Interruption”，影响了 19 种 Azure 服务，涉及 12 个 Region，当时似乎只有澳大利亚数据中心幸免于难。

从 11 月 19 日 00:50 发现该问题后的 5 个小时中，多个主要 Region 的存储服务出现问题，大量客户在此之间受到影响；上午 11 点存储故障恢复，大部分客户服务已经恢复，但少数客户的虚拟机由于此前的故障仍存在问题，自动恢复一直持续到 21 日早晨。

讽刺的是多个微软自己的服务也受到了牵连，Windows Store 和 Xbox Live 都受到不同程度的影响。而 Azure 的 Service Health Dashboard 也受故障影响，在故障发生之初尽然显示一切正常，Azure 也真是“醉”了。

一个月后，Azure 团队在其官方博客上就此次故障发表了详细的说明——[《Final Root Cause Analysis and Improvement Areas: Nov 18 Azure Storage Service Interruption》](#)，文中剖析了造成故障的原因。本次变更主要是针对 Azure Storage Table Front-Ends 的，目的是减少 CPU 开销，提升存储服务的性能。在测试和预生产环境中，本次变更顺利地通过了健康检查，同时显著提升了系统性能，还解决了几个已知的问题。但是在后续的增量部署中，不幸发生了，Azure Blob Storage Front-Ends 错误地开启了该功能，触发了一个 Bug，导致系统进入死循环无法再提供服务了。几分钟后，监控发现问题，工程师团队在 30 分钟内回滚了 Azure Blob Storage Front-Ends 的变更，对于已经进入死循环的系统进行了重启。

存储系统的故障还影响了虚拟机服务，主要可以将问题归纳为三类，都发生在存储故障和故障恢复阶段：

1. 磁盘挂载超时 (大部分虚拟机重启即可恢复)
2. 虚拟机配置失败 (仅影响 Windows 虚拟机 , Linux 虚拟机没有问题)
3. 虚拟网络编程错误 (几小时后工程师团队打了补丁 , 受影响的虚拟机无需重启即可恢复)

如果说代码的 Bug 未在测试中被发现尚情有可原 那么在[换一个灯泡都需要将近 50 个工程师参与，流程极为繁琐苛刻的微软](#)，不遵守流程就是不可原谅的了，《[Microsoft confirms Azure outage was human error](#)》一文直接就在标题上将其称为“人祸”。

Azure 的部署遵循名为 “flighting” 的流程，这个流程大致如下：

1. 在内部测试环境进行部署，随后测试并验证其功能；
2. 在预生产环境进行部署，在正常的生产级负载下进行测试；
3. 经过上两步的验证，就能在生产环境中进行小规模部署，一般会部署多个小集群；
4. 每个小集群都通过验证后就能进行更大规模的**增量部署**了，通常这种部署在地理上是隔离的。

负责本次性能优化的工程师认为该变更已经在生产环境的小集群上正常运行几个星期了，应该没有问题，在整个 Azure 上开启该功能没有太大风险。而配置工具也没有强制遵循增量部署变更的策略，导致该变更被部署到了整个系统上。

所谓没有规矩不成方圆，有了规矩不能贯彻执行也没用，在小公司或者初创团队，避免繁琐的流程的确能够带来很多便捷，但是在大型团队或者一些核心系统中，流程还是必须的。同时，**必须还要有系统能够保证流程得以正确执行**，人是会犯错的，人是会走捷径的，就像造成本次故障的那位同学，所以才要系统来进行约束。

Hacker News 上围绕 Azure 团队的故障分析展开了[讨论](#)，大家都对 Azure 团队的公开透明表示赞赏（其实在 Azure 的官网有个[页面](#)专门记录故障，相比某些公司出了问题遮遮掩掩，这种做法显然更受欢迎），同时不少人也在关心造成这次故障的那位同学的命运，一位读者引用了 IBM 的 Thomas Watson 的话：

最近有人问我会不会开除那个给公司造成 60 万美元损失的员工，我说不会，我刚花了 60 万美元培训了他，为什么要让别人来坐享他的经验？

曾经在支付宝的运维团队也有位朋友这么告诉我——“对运维操作要有敬畏之心”，这句话一直被我记在心中，相信他一定是在背了重大故障之后才会有此感悟。估计 Azure 的那位同学后续一定在操作时会更加仔细，遵循规范。

不知各位读者在了解了 Azure 的这次故障后，是否也能有所收获，当然，如果您在工作中也有过类似的经验教训，不妨也和大家分享一下吧。

Google 发布论文 披露大规模集群管理 工具 Borg 的细节

作者 : Abel Avram , 译者 : 邵思华

Google 最近发布了一篇名为 “[Google 使用 Borg 进行大规模集群的管理](#)” 的论文，披露了这个在过去极少提及的技术的细节。

Borg 是一个集群管理器，它负责对来自于几千个应用程序所提交的 job 进行接收、调试、启动、停止、重启和监控，这些 job 将用于不同的服务，运行在不同数量的集群中，每个集群各自都可包含最多几万台服务器。Borg 的目的是让开发者能够不必操心资源管理的问题，让他们专注于自己的工作，并且做到跨多个数据中心的资源利用率最大化。下面这张图表描述了 Borg 的主要架构：

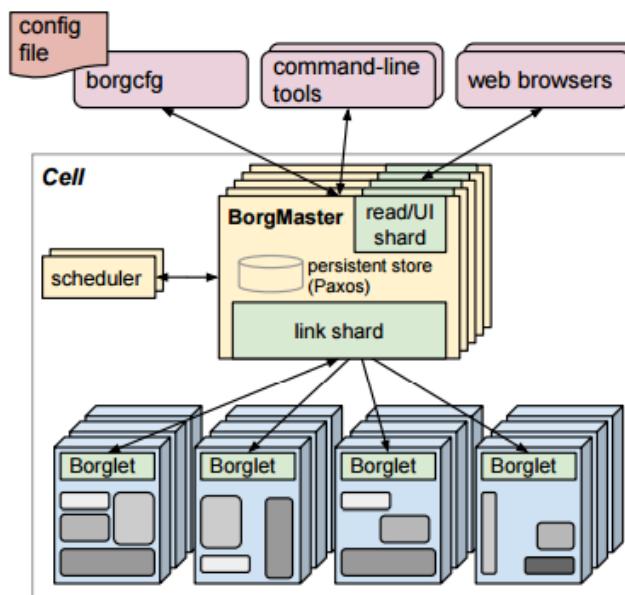


Figure 1: The high-level architecture of Borg. Only a tiny fraction of the thousands of worker nodes are shown.

图 1：Borg 的高级别架构图，其中只展示了全部几千个工作节点中很少的一部分。

这套架构中包含了以下组件：

- **单元(Cell)** —— 将多个机器的集合视为一个单元。单元通常包括 1 万台服务器，但如果有必要的话也可以增加这个数字，它们各自具有不同的 CPU、内存、磁盘容量等等。
- **集群** —— 一般来说包含了一个大型单元，有时也会包含一些用于特定目的的小单元，其中有些单元可以用做测试。一个集群通常来说限制在一个数据中心大楼里，集群中的所有机器都是通过高性能的网络进行连接的。一个网站可以包含多个大楼和集群。
- **Job** —— 是一种在单元的边界之内进行执行的活动。这些 job 可以附加一些需求信息，例如 CPU、OS、公开的 IP 等等。Job 之间可以互相通信，用户或监控 job 也可以通过 RPC 方式向某个 job 发送命令。
- **Task** —— 一个 job 可以一个或多个任务组成，这些任务在同一个可执行进行中运行。这些任务通常是直接在硬件上运行的，而不是在虚拟环境中运行，以避免虚拟化的成本。任务的程序是静态链接的，以避免在运行时进行动态链接。
- **分配额(alloc)** —— 专门为一个或多个任务所保留的机器资源集。分配额能够与运行于其上的任务一起被转移到一个不同的机器上。一个分配额集表示为某个 job 保留的资源，并且分布在多台机器上。
- **Borglet** —— 一个运行在每台机器上的代理。
- **Borgmaster** —— 一个控制器进程，它在单元的级别上运行，并保存着所有 Borglet 上的状态数据。Borgmaster 将 job 发送到某个队列中以执行。Borgmaster 和它的数据将会进行五次复制，数据将被持久化在一个 Paxos 存储系统中。所有的 Borgmaster 中有一个领导者。
- **调度器** —— 对队列进行监控，并根据每个机器的可用资源情况对 job 进行调度。

Borg 系统的使用者将向系统提交包含了一个或多个任务的 job，这些任务将共享同样的二进制代码，并在一个单元中进行执行，每个 Borg 单元由多台机器组成。在这些单元中，Borg 会组合两种类型的活动：一种是例如 Gmail、GDocs、BigTable 之类的长期运行服务，这些服务的响应延迟很短，最多几百毫秒。另一种是批量处理的 job，它们无须对请求进行即时响应，运行的时间可能会很长，甚至是几天。第一种类型的 job 被称为 prod job (即生产 job)，它们相对于批量 job 来说优先级更高，后者被认为非生产环境中的 job。生产 job 能够获得一个单元的 CPU 资源中的 70%，并且占用所有 CPU 数量的大约 60%，它们还能够分配到 55% 的内存，并占用其中的大约 85%。按照 Google 的研究员 [John Wilkes 所说](#)：在单元中混合不同类型的 job，目的在于尽可能地优化资源的使用情况，能够节约 Google 在整个数据中心上的成本。

根据论文中所写的内容，某些单元的任务量是每分钟接受 1 万个新的任务，而一个 Borgmaster 能够使用 10 到 14 个 CPU 内核，以及 50GB 的内存。一个 borgmaster 能够实现 99.99% 的可用性，但即使某个 borgmaster 或 borglet 出现停机状况，任务也能够继续运行。有 50% 的机器会运行 9 个或 9 个以上的任务，而某些机器能够在 4500 个线程中运行 25 个任务。任务的启动延迟平均时

间是 25 秒，其中的 20 秒用于安装包。这些等待时间中的大部分都与磁盘访问有关。

这套系统的主要安全机制是 Linux chroot jail 以及 ssh，通过 borglet 进行任务调试的工作。对于运行在 GAE 或 GCE 中的外部软件，Google 将使用托管的虚拟机，作为一个 Borg 任务在某个 KVM 进程中运行。

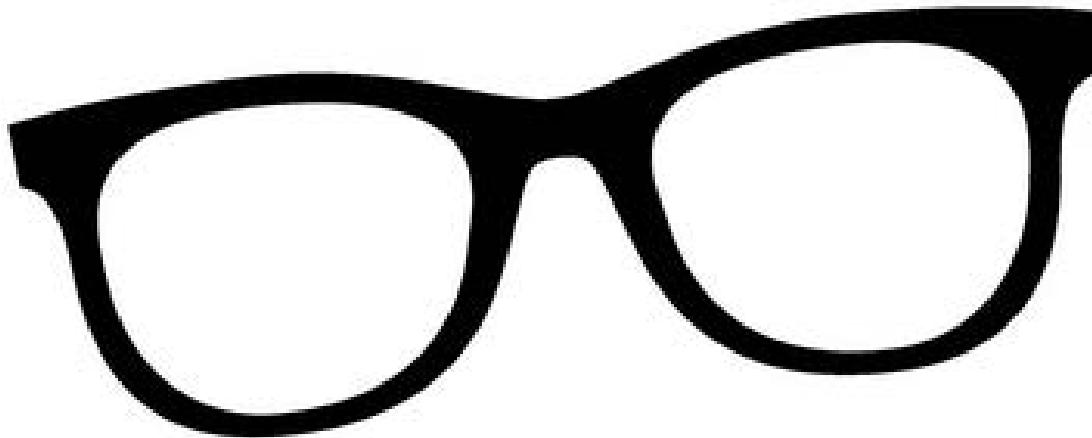
Google 还有一个名为 [Omega](#) 的集群管理器，这里简单地描述一下 Omega：

Omega 支持多个并行的、特定的“垂直任务”，其中每一个垂直任务基本类似于一个 Borgmaster，只是缺少了持久化存储与连接分片的功能。Omega 的调度器使用了优化的并发控制，对于储存在某个中央持久化储存系统中的单元状态的理想情况与观察到的情况进行操控，这个调度器通过一个独立的连接组件与 Borglet 进行双向同步。设计 Omega 架构的目的是为了支持多种不同的工作任务，每一种都有自己的应用程序特定 RPC 接口，状态机，以及调度策略（比方说，长时间运行服务器、来自于不同框架的批量 job，例如集群储存系统之类的基础设施服务，以及 Google 云平台上的虚拟机）。另一方面，Borg 提供了一种“适合所有情况”的 RPC 接口、状态机机制以及调度器的策略，它们的规模与复杂度随着时间的推移都在不断地增长，其原因是它需要支持多种不同的工作任务。目前来说它的伸缩性还没有遇到什么问题。

Google 在过去十年间在生产环境上所学到的某些经验与教训已经应用在 [Kubernetes](#) 的设计上：对属于同一服务的 job 的操控能力、一台机器多个 IP 地址、使用某种简化的 job 配置机制、使用 pods（其作用与分配额相同）负载均衡、深度反思为用户提供调试数据的方式。参加了 Borg 项目的许多工程师目前也参与了 Kubernetes 这个项目的研发。

查看英文原文：[Google Unveils Details about Borg](#)

Docker 专栏



云生态专刊



Docker: 现在和未来

作者 : Chris Swan

译者 : 张晓鹏

Docker – 迄今为止的故事

Docker 是一种 Linux 容器工具集，它是为“构建 (build)、交付 (ship) 和运行 (run)”分布式应用而设计的。作为 DotCloud 公司的开源项目，其首发版本的时间是 2013 年的 3 月份。该项目很快就受到欢迎，这也使得 DotCloud 公司将其品牌改为 Docker(并最终将其原有的 PaaS 业务出售而专注在 Docker 上)。Docker 1.0 在 2014 年 6 月发布，而且延续了之前每月发布一个版本的节奏。

其 1.0 版本标志着 Docker 公司认为 Docker 平台已经足够成熟，并可以被应用到产品中（公司及其合作伙伴们还提供了一些需要付费的支持选项）。每月的版本更新显示出该项目仍在快速发展，比如增加新的特性，解决发现的问题。这个项目成功地将“交付”和“运行”解耦，这样源自任意 Docker 版本的镜像都可以和其它任意不同版本一起工作（前向和后向均可兼容），这就为 Docker 应用提供了稳定的基础，以应对快速的变化。

Docker 发展成最受欢迎的开源项目可能会被人看作是一种炒作，但其实这个结果还是有坚实的基础来支撑的。Docker 吸引了业界众多知名大牌厂家的支持，其中包括亚马逊 (Amazon)、Canonical、CenturyLink、谷歌 (Google)、IBM、Microsoft、New Relic、Pivotal、Red Hat 和 VMware，这使得只要在有 Linux 的地方，Docker 就几乎随处可用。除了这些大厂家，许多初创企业也围绕着 Docker 来发展，或是将它们的发展方向和 Docker 更好地结合起来。所有这些合作伙伴（或大或小）驱动着核心项目和周边生态系统的快速发展。

Docker 技术的简要综述

Docker 利用了一些 Linux 核心工具，比如 [cGroups](#)、命名空间和 [SELinux](#) 来支撑容器之间的隔离。起初 Docker 只是 [LXC](#) 容器管理子系统的前端，但它在 0.9 版本中引入了 [libcontainer](#)，这个原生的 Go 语言库提供了用户空间和内核之间的接口。

Docker 容器是基于联合文件系统 (union file system) 的，比如 [AUFS](#)，利用它可以跨多个容器来共享一些组件，如操作系统镜像和安装的库文件。这种分层的方式也被 [Dockerfile](#) DevOps 工具充分利用，这可以缓存那些已经成功完成的操作。这就省掉了那些安装操作系统和应用程序依赖文件的时间，大幅度加速了测试周期。另外，在容器之间共享库还能减少内存的占用。

Docker 容器是从镜像开始的，镜像可以是本地创建的、本地缓存的，或者是从注册库中下载的。

Docker 公司运营着 [DockerHub 公有注册库](#)，上面有官方的数据仓库，是为不同的操作系统、中间件和数据库而创建的。组织和个人可以在 Docker Hub 上发布镜像的公有库，也可以将其注册成私有库。由于上传的镜像文件可以包含任何东西，所以 Docker Hub 提供了一种自动构建工具（之前被称为“可信的构建”），镜像构建于一个 Dockerfile，它作为镜像内容的清单。

容器和虚拟机的对比

Docker 容器要比虚拟机有效率的多，这是因为它们可以共享内核和相关的库。同样的原因，容器所占用的内存也要比虚拟机少得多，虽然虚拟机利用了 RAM 的过度承诺技术（RAM overcommitment）。容器也减少了对存储的占用，因为部署的容器会共享镜像的底层。IBM 的 Boden Russel 已经做了一个[基准测试（benchmarking）](#)来对比两者的不同。

容器也表现出比虚拟机更低的系统负载，所以同样的应用，在容器中相比在虚拟机中，性能通常会相当或者更好。IBM 的研究者团队发布了一个[虚拟机和 Linux 容器性能对比](#)的报告可以参考。

容器只是在隔离特性上要比虚拟机差，虚拟机可以使用 ring-1 特权的硬件隔离技术，如 Intel 的 VT-d 和 VT-x。这种隔离技术可以防止虚拟机破出（breaking out）和彼此交互。容器没有任何形式的硬件隔离，这使得它容易受到漏洞的利用。从 Shocker（可对 Docker 进行概念攻击）的验证来看，Docker 1.0 之前的版本都是很脆弱的。尽管利用 Shocker，Docker 在 1.0 版本中修复了一些特定的问题，但 Docker 的 CTO Solomon Hykes 仍然[表示](#)：“当我们感觉可以轻松地宣称 Docker 打开箱（out-of-the-box）可以安全容纳非受信的 uid0 程序（译者注：root 和超级用户权限）时，我们一定会明言相告”。Hykes 的话从另一方面承认仍然存在一些其他的漏洞和相关的风险，在容器变得可靠之前还有很多工作要做。

对于许多用户案例，在容器和虚拟机二者之间选择其一是种错误的二分法。Docker 可以在虚拟机中运行地很好，这可以让它应用在已有的虚拟化框架中，如私有云和公有云。同样也有可能在容器中运行虚拟机，这有点像谷歌在它的云平台中使用容器的方式。只要 IaaS 得到广泛应用，并可按需提供虚拟机服务，那么就有理由期待未来数年容器和虚拟机的应用可以并存。还有一种可能，即将容器管理和虚拟化技术进行融合以提供一种两全其美的方法：所以硬件信任锚微虚拟化实现支撑 libcontainer 能够与 Docker 的工具链和生态系统的前端进行集成，而使用不同的提供了更好隔离性的后端。微虚拟技术（类似于 Bromium 的 [vSentry](#) 和 VMware 的 [Project Fargo](#)）已经应用到桌面环境中为应用提供基于硬件的隔离，所以相同的方法可以使用到 libcontainer 上，作为 Linux 核心容器机制的替代技术。

“容器化”的应用

几乎所有的 Linux 应用都可以运行在 Docker 容器里，并且对编程语言或架构的选择没有任何的限制。实际上仅有的限制在于从操作系统的角度来看容器被允许做什么。即使如此，也可以通过在特权模式下运行容器来降低限制，以大幅度地减少受到的控制（与此对应的是装载到容器里的应用风险增加，并可能会导致对主机操作系统的损坏。）

容器从镜像开始，反过来镜像也可以从运行的容器中得到。从本质上说有两种方法可以把应用置入到容器中-手动和 Dockerfile。

手动构建

手动构建从启动一个基础操作系统的容器开始，然后通过交互式终端，用所选 Linux 相关的包管理器安装应用程序及其依赖项（dependencies）。Zef Hemel 在他的文章“[使用 Linux 容器以支持便捷的应用部署](#)”中提供了这个过程的细致描述。一旦应用完成安装，新的容器就可以推送到注册库（比如 Docker Hub）中或者被导出成一个 tar 文件。

Dockerfile

Dockerfile 是对 Docker 容器创建过程进行脚本化的系统。每个 Dockerfile 详细说明了开始的基础镜像，以及随后一系列在容器中运行的命令和添加到容器中的文件。Dockerfile 也可以说明容器对外的端口，启动时的工作目录和缺省执行的命令。用 Dockerfile 构建的容器可以象手动构建那样被推送到注册库中或者导出成 tar 文件。Dockerfile 也可以应用到 Docker Hub 的自动构建系统中，即在 Docker 公司的控制下，在系统中根据 Dockerfile 从头构建镜像，并且这个镜像的源对于使用它的任何人都可见的。

一个进程？

手动构建还是使用 Dockerfile 来构建镜像，考虑的关键在于容器刚启动时只能执行一个单进程。如果容器的服务目的比较单一，比如只运行一个应用服务器，那么运行单个进程就没什么问题（一些争论说容器本应该只包括一个进程）。对于那些希望在容器中运行多个进程的情况，[管理进程\(supervisor process\)](#)需要先启动，这样它可以接着孵化出其他期望的进程。此时容器中没有初始的系统，所以任何事都要依赖 systemd，不修改新兴系统或类似系统都无法工作。

容器和微服务

全面描述使用微服务架构的哲理和益处已经超出了本文的范围（[InfoQ eMag: Microservices](#) 中有全面的阐述）。容器仍然是一种方便的方法来绑定和部署微服务的实例。

虽然目前微服务大规模的部署实例还是在（大量）虚拟机上，但容器提供了一种小规模部署的机会。容器具有共享的内存和针对操作系统的磁盘占用、通用代码库，这也意味着可以非常有效地一起部署多个版本的服务。

连接的容器

一些小的应用适合放在单个容器中，但许多情况下应用需要扩展到多个容器。Docker 成功催生了一系列新的应用合成工具、编制工具以及平台即服务实现。绝大多数努力的后面，是希望能简化从一组相互连接的容器来创建应用的过程。很多工具在扩展、容错、性能管理和部署资产

的版本控制方面也提供了帮助。

连接性

Docker 的网络能力相当原始，容器中的服务对相同主机的其它容器是可见的，并且 Docker 可以映射端口到主机操作系统，使得服务在网络中也是可见的。[Libchan](#) 是 Docker 官方赞助的连接方法，它提供了 Go 语言的网络服务库，类似于 [channels](#)。在 libchan 找到自己应用的道路之前，还是有空间留给第三方程序来提供一些补充性的网络服务。例如，Flocker 采用了基于代理的方法，这使得服务（连同底层的存储）可以在主机间进行迁移。

合成

Docker 有个原生的机制来连接容器，它所依赖的元数据可以被传送到相关的容器中，这些元数据被用做环境变量和主机入口。类似 [Fig](#) 和 [geard](#) 这样的应用合成工具，可以在单文件中表达这种依赖关系图，这样多个容器就可以互相配合成为一个系统。CenturyLink 的 [Pannamax](#) 合成工具在底层采用了和 Fig、geard 相似的方法，但加入了基于 Web 的用户接口，并且直接和 GitHub 进行了集成，这样就可以分享合成后的应用了。

编制

编制系统包括 [Decking](#)、New Relic 公司的 [Centurion](#) 和谷歌公司的 [Kubernetes](#)，它们的目标都是帮助实现容器的部署和它的生命周期管理。也有很多基于 [Apache Mesos](#) 系统（特别是它为应用长期运行设计的 [Marathon](#) 框架）的商业化实现（比如 [Mesosphere](#)），它们可以和 Docker 一起使用。通过提供在应用需求和底层架构间的抽象（例如，需求表达为 CPU 核数和内存大小），编制工具提供了两者之间的解耦，这种设计简化了应用开发和数据中心的运维。还有各种各样的编制系统，这主要是因为之前许多内部系统工具冒出来了，它们之前开发出来是用于管理容器大规模部署的。例如 Kubernetes 就是基于谷歌的 Omega 系统，而 [Omega](#) 系统是用来管理整个谷歌云环境中的容器。

合成工具和编制工具功能上会有部分的重合，所以使用时它们彼此可以作为补充。例如 Fig 可以用来描述容器功能上如何交互，同时 Kubernetes pods（译者注：pods 可以被看成一个容器组）可以用来提供相关的监测和扩展功能。

平台即服务

有很多原生于 Docker 的 PaaS 实现，例如 [Deis](#) 和 [Flynn](#)，已经体现出 Linux 容器在支持开发灵活性上的强大优势（而不是那些自以为是给定的一套语言和框架）。其它的云平台如 CloudFoundry、OpenShift 和 Apcera Continuum，已经采用集成基于 Docker 相关功能到现有系统中的技术路线，这样基于 Docker 镜像（或者是创建它们的 Dockerfiles）的应用在部署和管理的同时，仍然可以使用之前系统支持的语言和框架。

所有的云

由于 Docker 可以运行在任何有合理数据内核的 Linux 虚拟机上，所以它可以运行在很多 IaaS 提供的云上。许多大的云提供商宣布了对 Docker 和它的生态系统的附加支持。

亚马逊已经引入 Docker 到它的弹性豆茎 (Elastic Beanstalk) 系统中 (这是在 IaaS 之上的编制服务)。谷歌使 Docker 成为可管理的虚拟机 (managed VMs)，它提供了在应用程序引擎的 PaaS 和计算引擎的 IaaS 之间的中间站。微软和 IBM 也都宣布了基于 Kubernetes 的服务，这样在他们的云上就可以部署和管理多容器应用了。

为了给当前使用的广泛多样的后端提供一致性的接口，Docker 团队引入了 [libswarm](#)，它会被集成到多数的云和资源管理系统中。Libswarm 一个明确的目标是“通过切换服务来源的办法来防止供应商锁定”。这通过呈现一组一致性的服务(以及相关的 API)来完成，这些服务会附着到特定后端的实现上。比如 Docker 服务器服务会呈现 Docker 远程 API 到本地 Docker 命令行工具中，这样众多的服务提供者就可以管理相关的容器了。

基于 Docker 的新的服务类型仍在起步阶段。虽然位于伦敦的果园实验室 (Orchard labs) 可以提供 Docker 容器的托管服务，但 Docker 公司表示，在它收购果园实验室后相关服务不会被置于优先位置。Docker 公司还向 cloudControl 公司出售了其先前的 DotCloud PaaS 业务。如 [OpenVZ](#) 之类基于旧的容器管理系统的服务已经比较普通了，所以在一定程度上 Docker 需要向主机提供商们证明其价值。

Docker 和它的发行版

Docker 已经成为主流 Linux 版本的标准特性，比如 Ubuntu，Red Hat Enterprise Linux (RHEL) 和 CentOS。遗憾的是这些 Linux 发行版与 Docker 项目在步调上并不一致，从而导致这些 Linux 发行版中 Docker 的版本比当前能用的老得多。例如 Ubuntu 14.04 的 Docker 版本是 0.9.1，并且在 Ubuntu 升级到 14.04.1 时 Docker 的版本也没有变 (当时 Docker 项目版本是 1.1.2)。在官方库中还有一个命名空间冲突的问题，因为 Docker 也是 KDE 系统托盘的名字，所以在 Ubuntu 14.04 中 Docker 包和命令行工具起了另一个名称 “docker.io”。

在企业级 Linux 发行版中情况也没有太大的不同，CentOS 7 中 Docker 的版本是 0.11.1，这是 Docker 公司宣布 1.0 版本产品准备就绪之前的一个开发版。Linux 发行版用户如果要使用最新的版本以保证稳定性、性能和安全，那么按照 Docker [安装指导](#) 操作并使用 Docker 公司提供的库，要比使用 Linux 发行版中的版本要好得多。

Docker 的到来也催生了新的 Linux 发行版，比如 [CoreOS](#) 和 Red Hat 的 [Project Atomic](#)，它们设计成能运行容器的最小环境系统。这些发行版相比传统 Linux 发行版本，有比较新的内核和 Docker 版本，对内存和硬盘的占用也比较小。新的发行版本中也有一些新的工具用来管理大容量的容器部署，比如 [fleet](#) 负责分布式系统启动，而 [etcd](#) 负责元数据管理。这些 Linux 发行版针对自身的分布式更新采用了新的机制，这样就可以使用最新版本的内核和 Docker 了。这表

示对 Docker 应用的其中一种效果的认可，那就是把注意力重心从发行版和包管理解决方案转到 Linux 内核(和使用内核的 Docker 子系统)上来。

尽管新发行版 (译者注 : 类似于 CoreOS) 可能是 Docker 最佳的运行方式 , 但支持容器的传统发行版和它们的包管理工具仍然非常重要。 Docker Hub 上提供了 Debian 、 Ubuntu 和 CentOS 的正式镜像 , 还有 “ 半官方 ” Fedora 的镜像库。 RHEL 没有在 Docker Hub 上的镜像 , 因为它们是直接由 Red Hat 发行的。这意味着 Docker Hub 上的自动构建机制只对那些纯开源的 Linux 发行版本可用 (并愿意信任那些起源于 Docker 公司团队策划的基础镜像) 。

Docker Hub 同时集成了源控制系统 , 如 GitHub 和 Bitbucker , 用来自动构建包管理器。包管理器可以在镜像构建过程中生成构建规格 (在 Dockerfile 中) 和最终构建镜像之间的复杂关系。构建过程的结果具有不确定性 , 这不是 Docker 的特定问题 , 而和包管理器如何工作相关。今天构建的是这个版本 , 在另一个时间构建可能会得到一个新的版本 , 这也就是包管理器需要更新功能的原因。容器抽象 (即较少关注容器中的内容) 与容器扩展 (因为轻量级资源利用率) 可能会使这种不确定性成为 Docker 相关的痛点。

Docker 的未来

Docker 公司已经建立了清晰的道路 , 即发展核心能力 (libcontainer) 、跨业务管理 (libswarm) 和容器间消息 (libchan) 。与此同时 , 通过收购果园实验室 (Orchard labs) , Docker 公司表达了利用自身生态系统的意愿。但是 , 这不仅仅关注 Docker 公司 , 这个项目的贡献者还来自于一些大牌公司 , 如谷歌、 IBM 和 Red Hat 。在仁慈的独裁者、首席技术官 Solomon Hykes 的掌舵下 , Docker 公司和 Docker 项目的技术领先有着明确的联系。在项目初始的 18 个月里 , 它已经显示出通过自己的输出来快速前进的能力 , 并且没有减弱的迹象。

许多投资者正着眼于十年前 VMware 公司 ESX/vSphere 平台的功能矩阵 , 试图找出已经由虚拟机普及而驱动的企业预期和现有 Docker 生态系统之间的差距 (和机会) 。在网络存储和细粒度的版本管理 (用于容器中的内容) 领域 , 现有 Docker 生态系统做得并不好 , 这就为初创企业和在职人员提供了机会。

随着时间的推移 , 虚拟机和容器 (Docker 中的 “ 运行 ” 部分) 之间的区别很可能变得不再那么重要 , 这将使注意力转到 “ 构建 (build) ” 和 “ 交付 (ship) ” 方面。这些变化将使 “ Docker 会发生什么 ? ” 的问题 , 相比 “ Docker 会带给 IT 业什么 ? ” 的问题 , 变得更不重要。

关于作者



Chris Swan 是云网络软件供应商 [CohesiveFT](#) 的首席技术官。作为银行业的技术专家和技术领域的银行专家 , 他曾经有十几年的时间在从事金融服务业。他大部分时间是在大型瑞士银行中与应用服务器、计算网格、安全、移动和云这些基础设施打交道。克里斯还喜欢参与互联网上的修补工作 , 包括一些

Raspberry Pi 项目。

查看英文原文 : [Docker: Present and Future](#)



腾讯万台规模的 Docker 应用实践

作者：郭蕾

Docker 提供了一种在安全、可重复的环境中自动部署软件的方式，拉开了基于云计算平台发布产品方式的变革序幕。腾讯内部对 Docker 有着广泛的使用，其基于 Yarn 的代号为 Gaia 的调度平台可以同时兼容 Docker 和非 Docker 类型的应用，并提供高并发任务调度和资源管理，它具有高度可伸缩性和可靠性，能够支持 MR 等离线业务。为了剖析 Docker on Gaia 背后的实现细节，InfoQ 专访了腾讯数据平台部高级工程师罗韩梅，另外作为 QCon 全球软件开发大会的讲师，罗韩梅将会[分享题为《Gaia——万台规模的 Docker 应用实战》的演讲](#)。关于 Docker 的交流讨论，也可以加入我们的 QQ 群：124378115。

嘉宾介绍

罗韩梅，腾讯数据平台部高级工程师，任数据中心资源调度组副组长。2009 年加入腾讯，主要从事统一资源管理调度平台的开发和运营，承担过腾讯自研云平台“台风”中 Torca 资源调度子系统的研发，目前主要专注于开源技术、分布式数据仓库、分布式资源调度平台、Docker 等领域。

InfoQ：能否介绍下目前 Gaia 平台的状态？你们什么时候开始使用 Docker 的？有多大的规模？

罗韩梅：Gaia 平台是腾讯数据平台部大数据平台的底层资源管理和调度系统，其上层业务包括离线、实时以及在线 service 服务 如 Hadoop MR、Spark、Storm、Hive 以及腾讯内部的 Lhotse、Hermes、广点通等业务。最大单集群规模达 8800 台、并发资源池个数达 2500 个，服务于腾讯所有事业群。我们是 2014 年 10 月份正式上线 Docker，之所以选择 Docker，一方面是因为 Gaia 本来就一直在使用 cgroups 类型的容器，深知其在共享机器资源、灵活、轻量、易扩展、隔离等方面的重要意义。另一重要原因，是 Gaia 作为一个通用的云操作系统，适合所有类型的业务，但是各个业务的环境依赖是一个比较困扰用户的问题，因此我们引入 Docker 来解决，主要目的还是通过 Docker 来将 Gaia 云平台以更有效的方式呈献给各个业务。

我们使用的 OS 是腾讯内部的 tlinux 1.2 版本，最新版本正在 tlinux2.0 上测试，除了 Docker，也使用了 etcd 用来做服务注册和服务发现。我们的集群都是同时兼容 Docker 应用和非 Docker 类型的应用的，MR 等应用还是使用的 cgroups 类型的容器，其它服务使用的 Docker 容器，目前，大概有 15000 多常驻的 Docker 容器，还有大量业务接入测试中。由于我们原本就是使用的 cgroups 容器，所以换成 Docker 后，性能基本也无损耗，可以满足线上需求。

InfoQ：腾讯是如何把 Yarn 与 Docker 深度结合的？

罗韩梅：在腾讯的场景下，首先一个特点就是，业务总类极大，尤其是离线处理规模很大，因此 Yarn 原生的调度器，效率远远跟不上，因此我们开发了自研调度器 SFair，解决了调度器效率和扩展性问题。另外，腾讯的业务特性多样，因此我们引入了 Docker，虽然 Yarn 支持不同的应用类型可以实现不同的 AM(应用管理器)，但是对于绝大多数应用来说，他们并不熟悉 Yarn，实现一个支持容灾、可扩展的完善 AM，困难较大，因此我们抽象了可以使用 Docker 的业务，对其进行封装，实现了统一的 AM，并且对用户透明，而对用户提供的是另一套全新的基本的、易于理解的高级接口。同时，我们为 Docker 业务实现了统一的服务注册和发现机制，并也将其封装在了新接口中。另外，在资源管理方面，我们修改了内存管理机制，引入了磁盘和网络带宽管理。

除了 Yarn 之外，其实我们对 Docker 本身也做了一定的修改和 bug 修复，对于 Registry 等服务也做了优化，保证了其服务的高可靠和性能。

实现方面，我们并没有使用社区提供的 Docker 调度器，我们研发 Gaia 的时候社区还没有相应的调度器，并且我们也有特殊要求，需要同时支持同时支持 Docker 类型应用和非 Docker 类型应用。

InfoQ：你们如何确定哪些业务适合使用 Docker？

罗韩梅：我们认为，Docker 提供了一种在安全、可重复的环境中自动部署软件的方式，拉开了基于云计算平台发布产品方式的变革序幕，因此，其实 Docker 对于 Gaia 来讲，只是一个选择，我们并不主动向业务推广 Docker，而是 Docker on Gaia 的整套方案，所以，我们对于需要共享资源、降低成本，需要支持快速的动态扩容缩容、容灾容错，以及大规模分布式系统尤为建议使用 Docker on Gaia。

InfoQ：能否详细介绍下你们对 Docker 以及 Registry 做了哪些优化？

罗韩梅：对于 Docker，我们主要做了三个方面的优化：首先是 bug 修复，比如 Docker 非 0 退出时 rm 不生效，对于 bindmount 为 true 时 config path 无法清除等 bug。其次是优化 Docker 的资源管理策略，比如内存的 Hardlimit 的管理策略，不但使用户进程容易被 kill，更加造成了资源的浪费，对用户估计自己业务的资源需求也非常高。Gaia 引入了 EMC (Elastic Memory Control) 的弹性内存管理机制。最后一个方面是资源管理纬度，Docker 在资源管理纬度方面只有 CPU 和内存两个维度，这对于共享的云环境下需要完善，也是目前相对于虚拟机不足的地方。Gaia 引入磁盘容量管理，网络出入带宽控制以及磁盘 IO 的控制维护。其实不仅仅在 Docker 层做控制，还将会引进调度器，不但实现资源的隔离，还要实现资源的保证。

对于 Registry 的优化，主要有下面几个方面：

1. 容量问题。开源的 Registry 是单机模式，其容量会受单个机器的限制。我们修改存储

driver，取缔原有的 mount 方式，开发后端存储 driver，直接使用 HDFS，实现了存储的无限容量。

2. 可靠性和可用性的问题。单机版本的 Docker Registry，其可靠性和可用性都成了最大的问题，我们引入数据平台部的 tPG 系统，实现 Registry server 的无状态化，便于实现服务的高可用性。
3. 性能问题。将单机版的 Registry 扩展成 Registry 集群，并实现在 Registry server pool 中的负载均衡，提升性能。
4. 网络问题。解决了全国不同 IDC 的 Gaia 集群对 Registry 的访问，采取就近访问的原则，不产生跨 IDC 流量。
5. 自动同步官方镜像。Docker 提供的官方镜像中，有很多还是非常有价值的，而官方的 Registry 又在墙外，为此，我们自动同步 docker 的官方镜像到我们的私有仓库中。

InfoQ：能否介绍下目前的一个 workflow？

罗韩梅：目前使用 Docker on Gaia 的方式有三种：1) 通过 Gaia Portal；2) 直接调用 Gaia api；3) 通过上层各种 PaaS 平台透明间接使用 Gaia。比如在第一种方式中，用户通过 Gaia Portal 提交应用，之后 Gaia 调度器会自动分配资源，并且部署、启动 Docker 容器，用户可以在 Portal 上直接查看每个实例的状态、日志、异常等，甚至可以直接通过 webshell 登陆。同时，也可以根据需求对应用进行扩容、缩容、重启，以及灰度变更、停止实例/应用等操作。

InfoQ：目前平台主要部署了哪些服务？服务之间的调度是如何实现的？

罗韩梅：目前平台上的服务有 Hermes、通用推荐、广点通、游戏云等服务，很多服务都需要多实例部署，因此跨主机部署非常普遍，而不同服务直接也经常会有调用的需求，主要是通过 Gaia 提供的服务注册和服务发现机制。具体地，NM（Yarn 的一个组件）在启动 Docker 容器时，会将该 Docker 的真正地址，包括 ip 和所有的端口映射，都会通过 etcd 做自动的服务注册。对于 Docker 内部的服务，我们通过修改 Docker 源码，扩展了几个 Gaia 相关的环境变量，将 IP 以及端口映射传入。服务的注册和发现本质上一种名字服务，因此不难理解为什么在创建应用的时候，让用户填一个应用 name 的字段。而这种基于名字的服务是贯穿这个 Gaia 的过程的：在提交作业时，用户不需要指定 Gaia master 的地址，而是通过指定 Gaia 集群的 name 即可；在获取应用的地址时，也是通过应用的名字获取；本质上 port mapping 也是一种名字，只不过是将用户原来 expose 的端口作为 name，将实际端口作为 value。至此，不难理解为什么 name 需要检查冲突。

InfoQ：万台规模的 Docker 容器，网络问题是如何解决的？

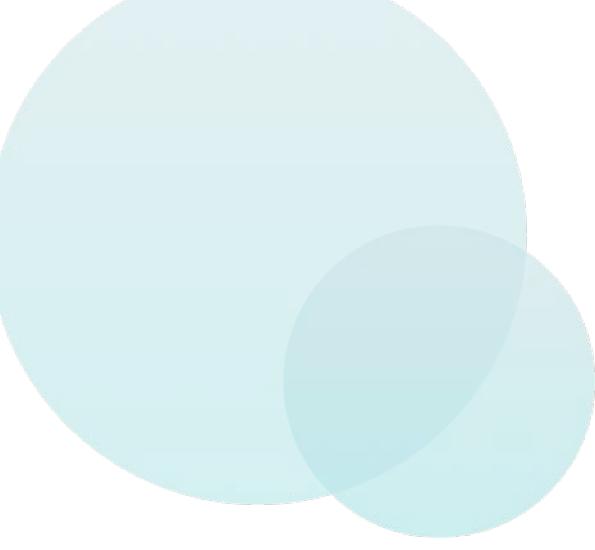
罗韩梅：网卡及交换链路的带宽资源是有限的。如果某个作业不受限制产生过量的网络流量，必然会挤占其它作业的网络带宽和响应时延。因此 Gaia 将网络和 CPU、内存一样，作为一种

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：

1. 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；
2. 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；
3. 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；
4. 尽量减少为了流控而主动丢包。

观点&趋势

云生态专刊



[观点] 运维的本质

——可视化

作者：王津银

没有比“可视化”更好的一个词能概括运维的本质，而“可视化”又应该分成两部分：**可视化的服务交付和可视化的服务度量**！

第一部分：可视化的服务交付

早期的运维是从 ITIL 开始的，那个时候大家都不知道运维是什么，幸好找到了一个 **IT 服务最佳实践——ITIL**。开始了互联网运维的摸索之路，从 CMDB、服务台、事件管理、变更管理、可用性管理、容量管理等逐步去了解，并同步建设对应的管理平台。但我们很快发现，这一完备的流程框架如果遇到了大规模运维的情况，就无法应对，原因在于过多的聚焦于流程以及规范，我们发现很难提升运维敏捷度和精细性，并且我们还是不知道一个完整的 IT 服务边界在哪儿？如何实现它？

不过在 ITIL 的实践过程中，其实提出了一个很好的概念——**IT 服务**。对于运维来说，提供一种高效、一致性、透明化、面向用户的服务是运维的价值所在，这样就要求运维屏蔽其提供的服务背后的所有实现细节。

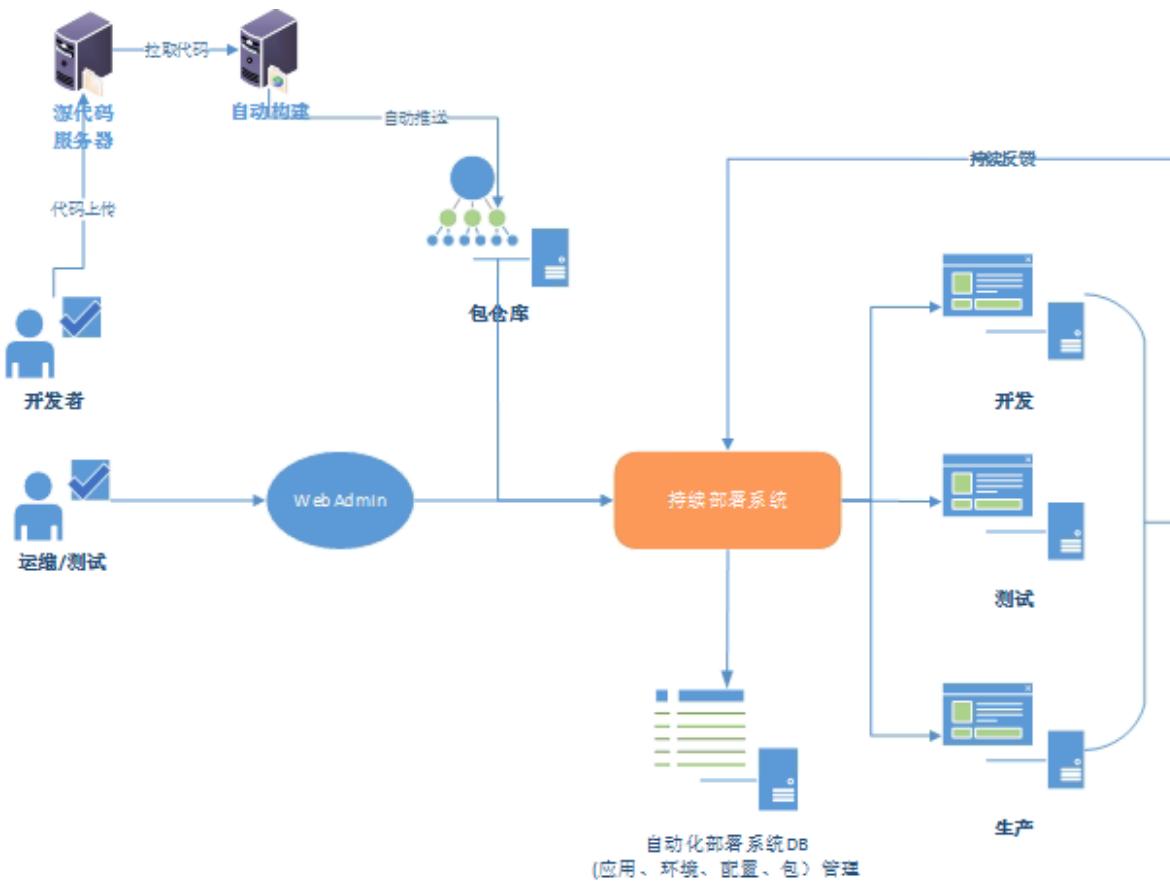
从运维具体事务或者活动的角度来说，如何对其进行一次或者多次的组合封装，把它们变成一个完整的 IT 运维服务，是此时的运维自动化重点方向。毕竟繁杂的运维事务不进一步封装，对个人或者团队来说，都意味着很高的学习成本和事务执行成本。在传统的 IT 运维组织中，我们能看到彼此事务之间的割裂非常明显，比如说网络、机房、服务器、应用部署等，都是在不同的团队完成，彼此工作独立进行。在敏捷和精益运维驱动之下，必须要求有一个集成平台来把这些事务流调度起来，否则无法提高事务执行的效率和质量，真正地把运维**交付功能变成了交付服务的模式**

对于如何封装这些事务或者活动，从 DevOps 提倡的“自动化一切”（Auto everything）可以找到些答案，其核心的自动化主线就是面向用户的敏捷持续交付。我把持续交付又分成两类场景：**一种是持续交付基础设施，一个是持续应用交付（持续构建、持续测试、持续部署、持续反馈）**，他们有点近似 IAAS 和 PAAS 的关系。

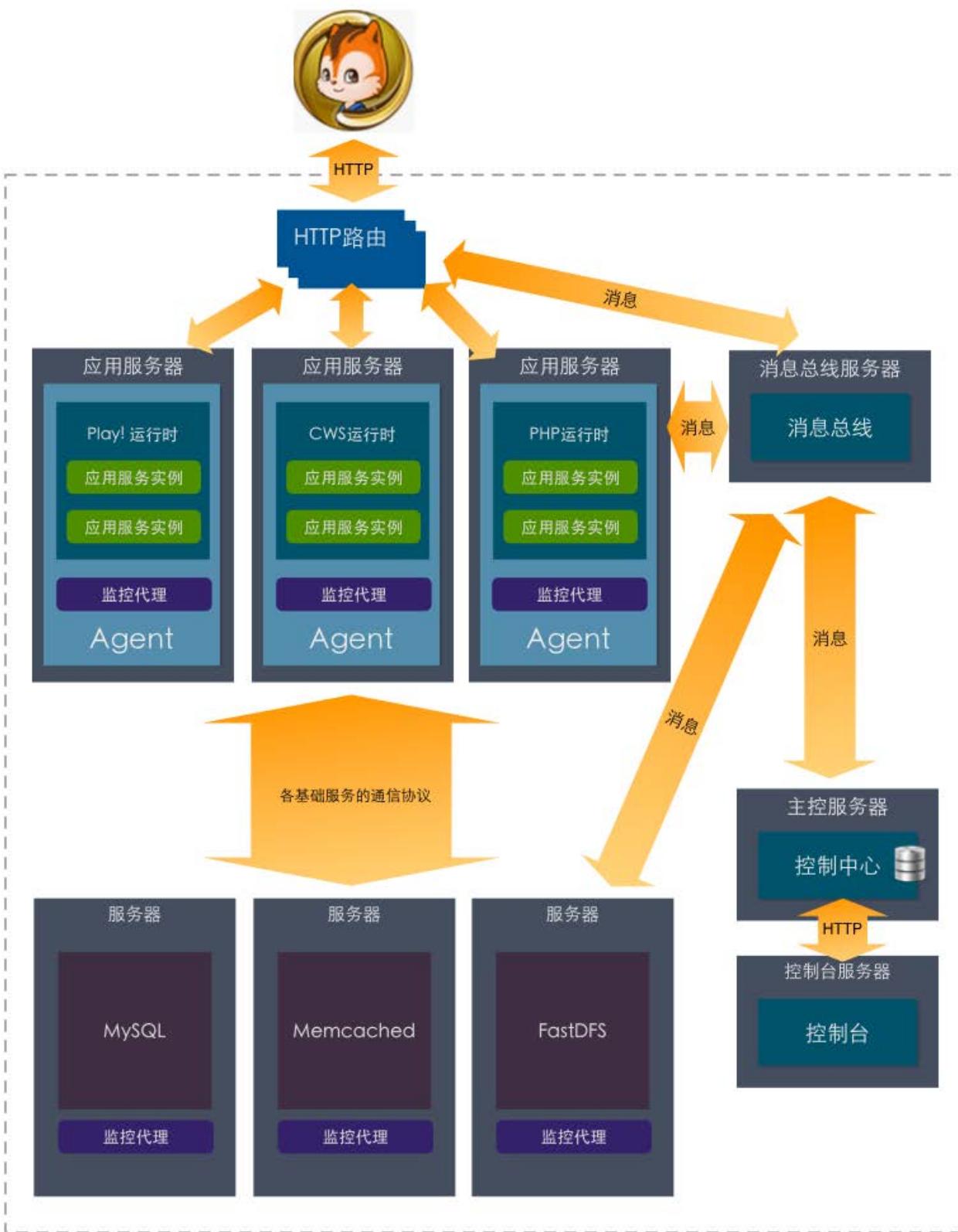
持续交付基础设施在公有云 IAAS 平台中得到很好的解决，利用**软件定义计算、存储、网络**等技术来实现对上层应用所需资源的快速交付。在私有 IT 环境中，当前有大量客户采用虚拟

机方案或者私有云方案来解决交付难和慢的问题。最新的轻量级虚拟化技术 Docker 更是热点，根本的原因是把应用的交付在镜像级别完成，从而让应用交付更加快速。

持续交付软件从代码产生的那一刻就开始进行管理，到编译、到测试、到灰度环境验收再到正式环境部署，并且希望这条主线完全自动化。**面向程序包的持续集成非常简单**，现在有很多的开源解决方案来实现，如 Jenkins、Go 等，但有一种情况需要特别注意，就是**程序包的配置管理问题**，这个也往往是影响部署的重要因素。所以我们很多时候使用开源平台只是为了构建程序包，后续包及其其中的配置管理以及实例化部署，特别是大规模集群部署，都是由单独的**持续部署**平台来解决，而非之前的持续集成工具（虽然它们也支持发布），但持续部署平台需要有和持续集成平台无缝对接的能力。



基于软件包的交付解决之后，我们希望交付的粒度更大，**如何实现全应用（从应用的前端接入到后端存储）的交付**，此时便有了 PAAS 平台和基于应用架构的可视化部署服务两种方案。这两种实现思路有很大的不同，我们知道完整的 PAAS 平台提供了对底层公共服务的向上 API 统一抽象，比如说数据库服务、存储服务、Cache 服务。PAAS 平台最经典的实现应该是 Cloud Foudry 了，国内很多 PAAS 平台基本上都是参考 CF 来实现的。阿里 UC 也有一个类似的 PAAS 平台，示意图如下。



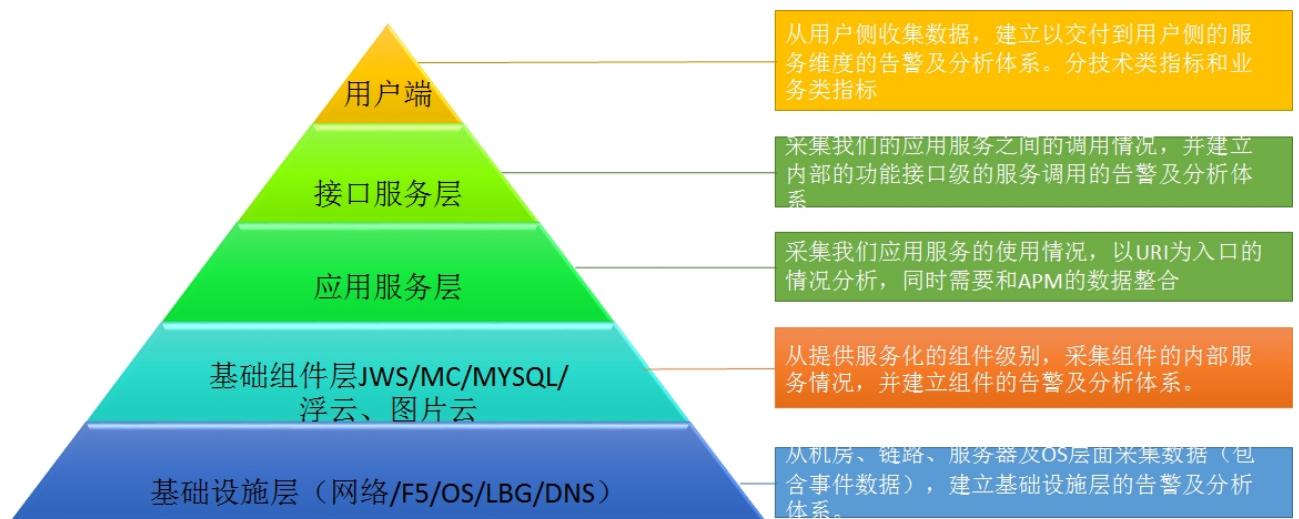
而在现实的情况下，很少公司有能力把 MySQL、MC、Fastdfs 封装公共服务供上层应用直接调用，意味着对研发程序有着一定的要求，是否还有一种更轻量的无约束自动化方式呢？我们可以把运维的全应用部署转变下思路，此时把应用架构中的各个部分拆解成对象组件(包含属性和状态)，比如说机房、OS、应用包等，全应用部署就是这些对象的编排，类似可视化 IDE 编程环境。

综上所述，运维的自动化最终要实现可视化，复杂的运维工作流必须通过可视化来表达，可视化后的自动化才能让所有人理解一致、执行一致、结果一致。

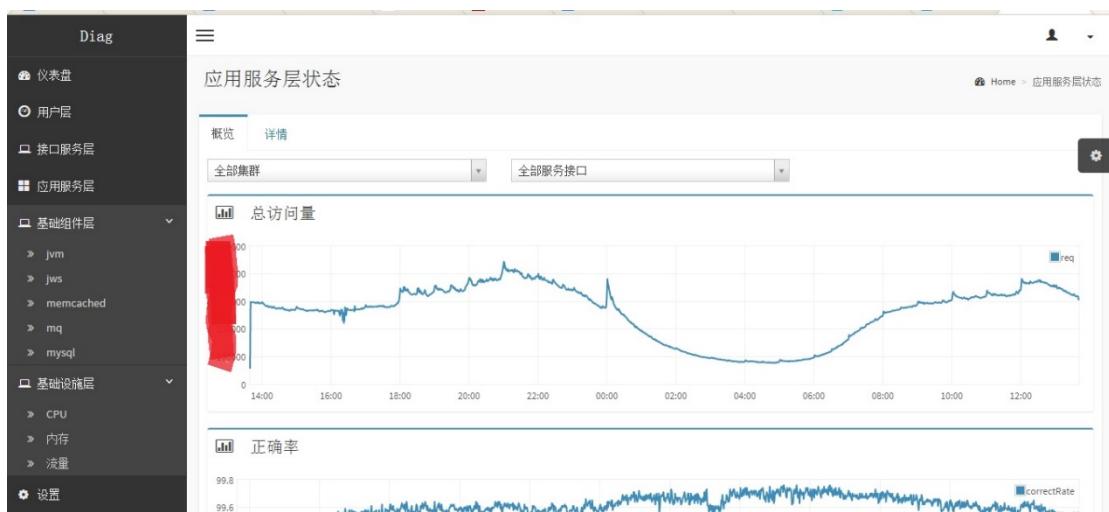
第二部分，可视化服务度量

“除了上帝，一切人都必须用数据说话”，这是运维人员必须恪守的信条。我写过一篇完整的数据驱动运维的文章[“关于数据驱动运维的几点认识”](#)，里面系统地介绍了数据化运维的目的、数据的来源以及如何构建数据体系，等等。

最近也在进行一个数据实践，就是建立面向应用的端到端数据分析体系，该体系对数据有个标准化的分层归类，从基础设施、上层组件、到应用服务、到接口、再到用户侧，基于应用的拓扑架构，收集各类指标，统一到一个分析平台中展现，如下图所示。



基于这套分层化的数据体系标准，我们也有对应的系统实现，如下图所示。



当形成标准的数据采集、分析和展现体系之后，可以向其他应用不断去复制这套方案，大家只需要遵循一套数据标准即可，最后数据的采集、分析、展现和告警都是标准化完成。这套

数据体系建设完成之后，可以在运维的故障定位、服务优化、架构改进、运维规划等各方面找到应用场景。

此时有人会有疑问如何面向应用把这些数据整合关联起来？我们当前是**基于配置文件的静态视图和基于接口调用而生成的动态视图来集成**。动态调用视图生成会复杂一点，可以让线上的接口调用统一由名字服务中心来接管调度，抽样对接口调用进行染色，从而生成动态的访问关系。

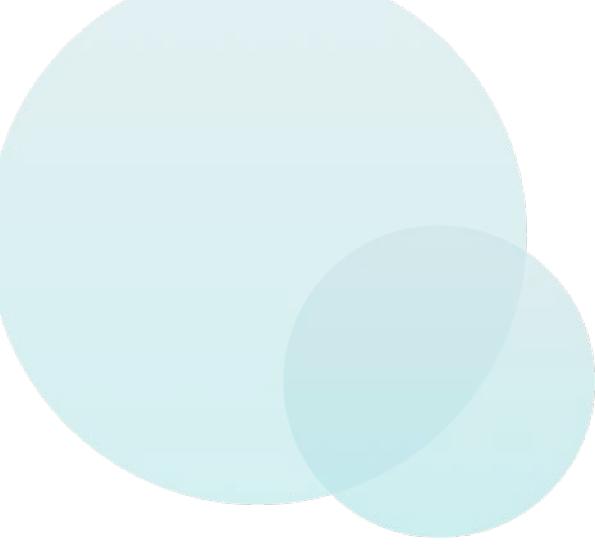
以上视图能快速发现和定位规模故障，但对于单个用户的故障指标上则应对乏力。此时分布式 Trace 服务的作用就显现出来了，可以借鉴 Twitter 的 Zippkin 和 Google 的 Dapper 的实现思路。当前我们就结合自身的业务架构特点，实现了一个统一的服务调度框架和名字服务中心，在业务代码无侵入的情况下，可以把业务调度链的染色数据上报和关联，实现对于单个问题的快速定位。

数据的可视化能力非常重要，需要在面向整体和面向某个业务流上都有实现。**它首先体现出你对运维的理解是什么样的**，从可视化 Dashboard 上可以看到最直接的运维经验；**其次基于可视化之上的数据共享，让大家对数据的理解达成一致**；**最后利用一致化的可视化数据发挥运维的驱动能力，驱动 DevOps，数据的核心价值就在于此**。

因此**可视化的能力建立了运维的能力**，可视化的程度越高，运维的能力越高。那么你现在到底可视化了哪些运维服务，并能进行度量呢？

作者简介

王津银，07 年进入腾讯公司接触运维，先后在 YY 和 UC 参与不同业务形态的运维，对运维有一些理解。极力倡导互联网价值运维理念，即面向用户的价值是由自动化平台交付传递，同时由数据化来提炼和衡量。微信公众号：互联网运维杂谈。



私有云之殇： 公有云这座大山(上)

编者按：本文系 InfoQ 中文站向 ZStack 项目创始人张鑫的约稿。2015 年 4 月正式对外开源的 ZStack 项目宣称要解决在 OpenStack 中得不到解决的问题，并明确将项目目标指向目前前景似乎越来越不明朗的私有云市场。作为 ZStack 项目的架构师、CloudStack 的前开发人员，张鑫对于私有云——或者说 on-premise 的企业 IT——到底遇到了什么问题、此类服务到底应该怎么做，是如何构想的？本文将分享他对这个话题的思考。

写在前面：由于企业私有云市场迟迟未打开，近两年来已有多家 IaaS 企业被廉价收购甚至倒闭，业界已经开始出现一种质疑私有云是伪命题的声音。在此，作者想借 ZStack 发布的机会，梳理一下私有云的过去和现状，并展望一下它的未来。

数据中心自动化的前世今生

从系统管理的角度来看，我们可以把企业 IT 市场分为三个阶段：前虚拟化纪元，虚拟化纪元，IaaS 纪元（基础设施即服务纪元）。

前虚拟化纪元是指虚拟化技术出现以前的阶段，可以追溯到计算机出现到 1998 年 VMWare 的创立。在这个阶段，人们管理机器的方式主要是人工部署架构（包括安装物理机，网络部署等），配合分布式软件管理。这个时期由于没有虚拟化技术，一台物理机只能运行一个操作系统，网络拓扑也必须事先设计然后硬连线，整体上效率低下，不灵活。由于一旦部署就难以改变，底层架构往往会制约上层业务；软件方面也没有统一标准，多是由系统管理员自己开发的脚本来实现部分自动化。

1998 年 VMWare 成立并发布首款操作系统虚拟化产品后，市场进入了企业级虚拟化纪元。由于虚拟化技术允许同一物理机上运行多个操作系统，并且操作系统之间可以用虚拟化软件提供的虚拟网络连接，其灵活性让 IT 架构的部署极大简化。系统管理员在安装物理机后，网络拓扑可以使用扁平的二级网络，即所有物理机在同一个二级网络广播域（broadcast domain），然后在虚拟网络级别实现隔离。这阶段虽然有虚拟化软件的帮助，系统管理员仍然需要在虚拟机级别手动部署网络或存储。比如需要手动为同一个虚拟网络内的虚拟机配置 DHCP/DNS 服务器，需要在不同的虚拟网络之间手动配置路由等。

从 Amazon 在 2006 年发布 EC2 公有云，市场开始真正进入 IaaS 纪元。跟虚拟化纪元不同

的是，IaaS 软件借助虚拟化技术不仅提供池化资源，还将计算、存储、网络资源按服务的方式提供给用户，实现按需索取。同样是部署网络的例子，在 IaaS 纪元，管理员不再需要手动配置 DHCP/DNS 服务器、网络路由，而是向 IaaS 软件描述需求，由 IaaS 软件自动去配置部署。随着软件定义网络(SDN)、软件定义存储(SDS)等技术的出现和应用，管理员甚至不再需要手动安装虚拟软件到物理服务器（借助 IPMI/PXE），不需要手动配置交换机，只需要把硬件安装在机架并连线，其余所有工作都交给软件处理。自此数据中心管理开始进入软件定义的全自动化时代。

私有云的需求是真实存在的

作为 CloudStack 的前开发人员（作者于 2010 年加入 Cloud.com，后被 Citrix 收购），作者曾多次作为 CloudStack 的代表参加了多家国际知名公司内部私有云项目的设计和策划，第一线地接触到客户的需求和痛点。在整个过程中，作者最深切的感受是：私有云市场的需求是真实存在的，但现有产品与客户需求的差距也是巨大的。除去加在私有云上的各种噱头，客户的核心需求是管理数据中心或企业 IT 架构中不断增加的硬件、实现自动化、减少底层架构的部署时间，从而加速企业在上层业务上的创新。用通俗的话说就是：

1. 机器越来越多，人管不过来，得让软件去管；
2. 老板急着上线业务，IT 部门不能拖后腿说光架设机器部署网络就要一个月。

这个刚需是非常巨大的。调研机构 TBR 在 2014 年发布的报告中估算当年私有云市场的总份额在 410 亿美元上下，并预测 2018 年私有云市场份额将达到 680 亿美元。

企业 IT 不适合简单复制公有云模式

IaaS 技术出现以来，其最广泛最成功的应用当属公有云。一大批公有云厂商的出现加速了行业的发展和创新，催生了大量相关产业，同时也在一定程度上帮助很多客户从底层架构的依赖中解脱出来，实现了业务创新。例如很多成功的 SaaS 公司就是完全运行在公有云之中。公有云的宣传口号是“让云计算像水和电一样”，这是非常高明的文案，让人潜意识中以为有了公有云为我们提供按需供应的计算资源，就不需要自己建立和维护 IT 架构或数据中心，毕竟没人会去自己建自来水厂和发电站。但实际上公有云远没有达到自来水管道和供电网络那般完善和稳定，在数据安全和成本方面也并不占优势。我认为公有云之所以成功，是因为它去除了基础设施里众多的差异性，抽象出能够满足大部分客户需要的模型，例如业界事实标准的 Amazon EC2 模型。

但这个模型并不适合所有场景。例如像 Facebook 这样的公司，交换机间就需要 40G 的独享带宽，存储上还需要专门公司提供的闪存（Flash）；而保险公司可能又希望数据库跑在物理机上，并且物理机网络要能够跟前端虚拟机的网络形成 VPC。公有云是不能够在一个多租户的环境内去满足各个客户的差异性需求的。所以，公有云虽然是到目前为止 IaaS 领域发展最

好也是最成功的例子，但要断言它必将统治世界还为时过早。如果哪天 Intel 和 AMD 的服务器芯片都被公有云厂商买走了，那么一个新时代必将开始；在此之前，我们仍然需要面对私有云市场的需求。

在过去几年跟不同客户打交道的过程中，作者深刻感觉到数据中心自动化以及企业 IT 架构云化的需求必须划分为两类。

一类是服务提供商的需求（service provider）。这类企业搭建私有云的目的是为了支撑上层业务的运行，例如电商公司的私有云、游戏公司的私有云。这类私有云对技术的要求跟公有云类似，因为他们的业务模式决定了基础架构需要有快速动态扩展的能力，存储需要是分布式以支撑数据的爆炸式增长等。这类公司通常有很强的 DevOps 能力，同时也面临着传统 IT 架构不能解决的问题，所以他们有能力部署不成熟的 IaaS 软件并追逐新技术。

另一类需求是传统企业，他们的 IT 架构主要停留在虚拟化纪元，多采用 VMWare 的产品。我们称这类需求为传统的企业虚拟化需求（enterprise virtualization）。云计算的三层结构中，面对终端用户的是 SaaS，而 IaaS、PaaS 都是为之服务的。三者关系存在一种上层应用决定下层架构，下层架构又反过来制约或促进上层应用的关系。传统企业的特点在于他们仍然运行着大量传统应用，这些应用很多都是上个世纪的产品，它们并非为云环境所设计，简单的把它们搬迁到公有云模式中会导致很多技术上的困难。

在公有云模式中有两个重要的特点：一个是资源池化，一个是应用程序需要能容忍基础架构的失败。

资源池化是公有云去差异化的一个重要设计，它主张租户共享资源池，资源获取的过程不可定制。但传统应用在很多场合是需要能够进行差异化定制的。例如对于 IO 密集的传统应用，用户可能期望虚拟机的根磁盘（装有操作系统的磁盘）放在 NFS 上，数据磁盘通过专门的存储网络放在高速的 IP SAN 上面。这就要求对创建虚拟机的过程可以定制，用户可以选择不同的磁盘分配策略和不同的网络。又比如 Citrix 的远程桌面 XenDesktop，它需要多租户即独享私有网络又共享扁平网络的拓扑，这就要求类似于 Amazon VPC + Classic EC2 的网络模型。当企业业务由众多传统应用构成时，这种差异性的定制需求会进一步加强。公有云提供商也认识到了这种需求，并在有限范围内修改以满足客户。例如 Amazon VPC 发布最初是不支持静态 IP(static IP)的，后来在用户的强烈要求下才加入，但它的 Classic EC2 是不可能在多租户环境下支持这种功能的。去差异化是公有云的成功法宝，它无法也不会为了迎合传统程序而实现各种定制功能。

在公有云中运行良好的应用都是能够在应用层面容忍一定程度的基础架构失败（比如虚拟机崩溃，网络失联等）的。公有云厂商一直在反复向客户强调基础架构是会失效的，也提供多种手段帮助用户设计 failover 的应用，而这恰恰是传统应用的死穴，因为它们很多是单机应用，对下层架构的稳定性要求很高。所以大量传统企业因为稳定性而仍然依赖 IOE（IBM、Oracle、EMC）的架构。企业客户在选择 IaaS 产品时，往往要求 IaaS 能够在架构层面提供

容错手段，比如虚拟机级别的高可靠、网络层面的高可靠和冗余。这种由传统应用带来的需求不是云友好（cloud-friendly）和云感知（cloud-aware）的，公有云模式是不会对他们做出过多考虑的。

这也是为什么对于服务提供商来说，私有云可以很容易照搬公有云模式，但在面对传统企业客户的需求时，简单复制公有云模式是不够的——因为我们面对的是一个强大的、不可更改的传统应用生态。

参考文章

《Why OpenStack is different from other open source projects》

《Keep OpenStack Weird》

《Why vendors can't sell OpenStack to enterprises》

作者简介

张鑫，2006 年加入 Intel 上海开源技术中心（OTC），从事开源虚拟机项目 XEN 的开发，为社区共享了多个功能，例如 XEN 中 E100 网卡模拟器，XEN/IA64 虚拟 BIOS 对 Windows 的支持等。同时也共享了大量 bug 修复的补丁。2010 年赴硅谷加入 Cloud.com（后被 Citrix 收购），从事 CloudStack 的开发工作，其间多次作为 CloudStack 代表参与客户私有云项目的设计和部署。在从 Citrix 退出后，和搭档一起创立 ZStack。



生态圈新闻

云生态专刊

Suse 推出基于 Ceph 的企业及存储服务

近日，开源软件提供商 Suse 正式发布 SUSE Enterprise Storage。该产品基于 Ceph 的 Firefly 版本，是 Ceph 的商业支持版本，具有缓存和分层、自动精简配置、写时复制和纠删码等功能。SUSE Enterprise Storage 之所以选择 Ceph，原因在于它是最流行的 OpenStack 分布式存储解决方案。

Suse Enterprise Storage 既作为 Suse OpenStack Cloud 的一个可选项提供，也可以作为一个独立的产品使用，其价格为每 PB 每月 0.1 美分。SUSE Enterprise Storage 这款纯软件产品的出现有望打破存储市场上硬件厂商占统治地位的局面。

Engine Yard 收购开源 PaaS 平台 Deis

4 月 15 日，Engine Yard 宣布收购 OpDemand 公司，具体交易金额不详。Engine Yard 表示在收购后，OpDemand 的所有员工都会加入 Engine Yard，并继续参与开源项目，同样，Deis 也仍会继续开源。合并后的团队将重点关注 Deis 在调度、编排以及应用监控方面的问题，并致力于帮助客户在生产环境中使用 Deis。

OpDemand 是开源的 PaaS 平台 Deis 的母公司，Deis 是一个受 Heroku 启发的基于 Docker 和 CoreOS 的轻量级 PaaS 平台，旨在让部署和管理服务器上的应用变得轻松容易。

英特尔、华为联合举办中国首届 OpenStack Hackathon

2015 年 4 月 13-15 日，由英特尔与华为联合举办的国内首届 OpenStack Hackathon 在上海举行。来自英特尔的 9 位工程师、来自华为的 6 位工程师与来自海云捷讯的 2 位工程师在三天的合作编码中，集中解决 Nova、Neutron 与 Oslo/Messaging 组件中的 29 个 bug，将形成的方案与代码交付到社区上游的核心开发者，以改进即将发布的下一个大版本——OpenStack Kilo。

活动组织者希望邀请更多 OpenStack 从业者与社区开发者们加入下一次 Hackathon，并计划邀请相关组件的核心开发者到场进行现场代码审核。活动将作为 Intel 和华为的传统，每年将举办两次。下一次活动的时间地点确定后会通过 OpenStack 社区群组发布通告。

Docker 招聘中国区主管

5 月 1 日，Docker 发布了自家的容器网络管理项目 libnetwork，目标是定义一个容器网络模型（CNM），并为应用程序提供一致的编程接口以及网络抽象。libnetwork 使用 Go 语言编写，目前仍在全力开发中，并没有达到使用标准。

另外，从官方邮件中得知，Docker 公司正准备进军中国。目前他们正在招聘中国区的主管，以运营中国社区，并进一步扩展 Docker 的影响力。新的主管直接汇报给 CFO。具体信息读者可以参看具体的 JD。

其他业界新闻速递

2015 年一季度 ,OpenStack 社区活跃贡献者每月平均有 556 名 , 这一数字创造了 OpenStack 的历史新高。与之前一个季度相比 , 核心和普通代码贡献者的数目分别上涨了 %10 和 %20 。

2015 年愚人节当天 ,OpenStack 创始者之一的 Nebula 宣布因市场原因而关闭。对于 Nebula 的倒闭 , UnitedStack 公司创始人兼 CEO 程辉在接受采访时说 , “Nebula 也给社区提了个醒 , 单纯的交钥匙工程行不通。 UnitedStack 目前提供的服务是持续运维和更新 , 帮助用户解决 “ 如何将 OpenStack 应用在商用环境中 ” 的问题 , 打通 “OpenStack 商用的最后一公里 ” 。这也是 UnitedStack 的价值所在。”

无独有偶 ,HP 云计算部门负责人 Bill Hilf 在接受纽约时报采访时的言论也引起了小小的波澜 , 记者将 Bill 的话理解为 HP 将放弃公有云业务。中国市场方面 , 随着私有云解决方案在的逐步扩大 ,HP 中国再次扩大其 Helion 产品组合 , 加入了一个新的基于 OpenStack 和 Cloud Foundry 的私有云产品。

亚马逊在今年一季度财报中公开了 “ 亚马逊网络服务 ” (AmazonWebServices , 下简称 “ AWS ”) 部门的业绩数据 , 这是自该部门于 2006 年成立以来首次公布财报数据。当季 , 亚马逊 AWS 云计算业务营收 15.6 亿美元 , 同比增长 49%; 运营利润 2.65 亿美元 , 同比增长 2000 万美元 , 运营利润率约 17% 。除了已经入华的 Windows Azure 和 AWS ,IBM

大力推广的 BlueMix 下半年有望在中国落地。

海外公司中 , 作为 OpenStack 的早期支持者 Mirantis 宣布加入 Cloud Foundry 基金会 , 而 CloudStack 的主要玩家 Citrix 则宣布加入 OpenStack 基金会。

在国内市场 , 环信 CTO 马晓宇在接受采访时表示 , 企业 SaaS 服务正在成为主流趋势。从用户数量的增长速度和活跃度来看 , 最热门的是图片类的应用 ; 一些特定的垂直领域应用用户粘性也比较大。

1天为你的APP
加入移动IM功能

即时通讯云领导者

环信提供更稳定可靠的即时通讯云服务，并承诺全面开源与深度安全 不用自建服务器，不用自己开发，加几行代码轻松拥有IM功能



单聊功能

支持发送语音，图片，表情，文字，位置，名片，附件



群聊功能

支持500人到2000人大群
拥有完善的群组权限管理



移动客服

7x24 客服在线，实时沟通



实时音视频

基于IP网络的点对点实时语音
适应低带宽要求



Web IM

跨平台基于浏览器的IM客户端
保证高质量用户体验



官方微信

想了解更多环信即时通讯云功能
赶快扫一扫



有云

世界更精彩

www.unitedstack.com

这是一个天翻地覆的时代
每一个梦想都有闪光的机会
我们,是云时代的创新者
两年的奋斗时光
回首
我们欣慰已完美踏出第一步
今天的**UnitedStack**日新月异
一个全新的我,
期待一个全新的你



UnitedStack 有云
openstack [cloud] services



促进软件开发领域知识与创新的传播



中文 | 英文 | 日文 | 葡文 |