



一种基于容器编排技术的资源运维系统的设计与实现

张勋, 张呈宇, 魏进武

(中国联合网络通信有限公司研究院大数据研究中心, 北京 100176)

摘要: 容器运行、容器编排、框架编排是构建容器运维系统的核心功能。容器资源运维系统的工作原理是运用 Mesos 进行资源调度、运用 Kubernetes 进行容器编排、运用 Marathon 技术进行框架管理。这 3 种技术在容器运维中相互关联、相互衔接, 使得系统内部机制正常运转。深入阐述了这 3 项技术是如何应用到实际系统构建中的, 给出了设计的组合架构及改进技术难点, 以解决中国联通现有业务应用容器化之后资源调度不平衡的问题、现有物理集群弹性扩/缩容的问题以及实际业务部署统一运营的问题, 从而实现容器运维管理系统的建设。

关键词: 资源运维系统; 容器调度; 弹性扩/缩容

中图分类号: TN923

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2018110

Design and implementation of a resource operation and maintenance system based on container arrangement technology

ZHANG Xun, ZHANG Chengyu, WEI Jinwu

Big Data Center of China Unicom Research Institute, Beijing 100176, China

Abstract: Kubernetes container layout tools, Docker operation technology and Mesos framework choreography technology selection were used to solve China Unicom's existing problems of business application container's unbalance of resource scheduling. The problem of existing physical cluster elastic scalability capacity was solved. Through cluster and the technique of Mesos, the original big data deploy faster and the scalability problems were solved. Finally, container operation and management system was realized.

Key words: resource operation system, container scheduling, elastic scalability capacity

1 引言

目前中国联合网络通信公司(以下简称联通)采用的以 VMware 虚拟机为主的虚拟化技术, 并不能满足现有通信领域, 实际业务应用使用容器技术进行虚拟化扩/缩容的要求; 并且联通现有大数据集群, 是以 Hadoop 为主的传统固定集群模

式, 扩/缩容需要依靠人工添加实际物理机的模式, 重新安装集群, 人员、硬件成本开销较大。而现有开源技术中, Kubernetes 可以很好地解决使用 Docker 虚拟化技术之后容器的生命周期问题, 以及实际业务应用容器化技术之后快速扩/缩容的问题。而 Mesos 可以解决 Hadoop 集群自动化扩/缩容的问题。因此中国联通研究院建设 CU-DCOS 容

器运维系统,结合现有开源技术解决上述两项生产环境的固有问题。研究开源技术与联通现有环境集成的容器资源运维系统的实现。

2 现有容器应用运维的问题

中国联通目前的IT系统能够满足生产供给和部署,但在实际运维中却十分依赖第三方厂商的

协助,仍旧是每个业务一个团队的“作坊”模式。每年的人力维护成本十分巨大,且效率较低。随着中国联通承载的实际业务越来越多,不同业务应用团队之间主机资源不平衡程度加剧,扩/缩容能力较低,严重影响了生产环境的稳定性和顽健性,如图1和图2所示。目前的各个业务系统运营存在资源运行不均衡、时段运行不均衡等重大

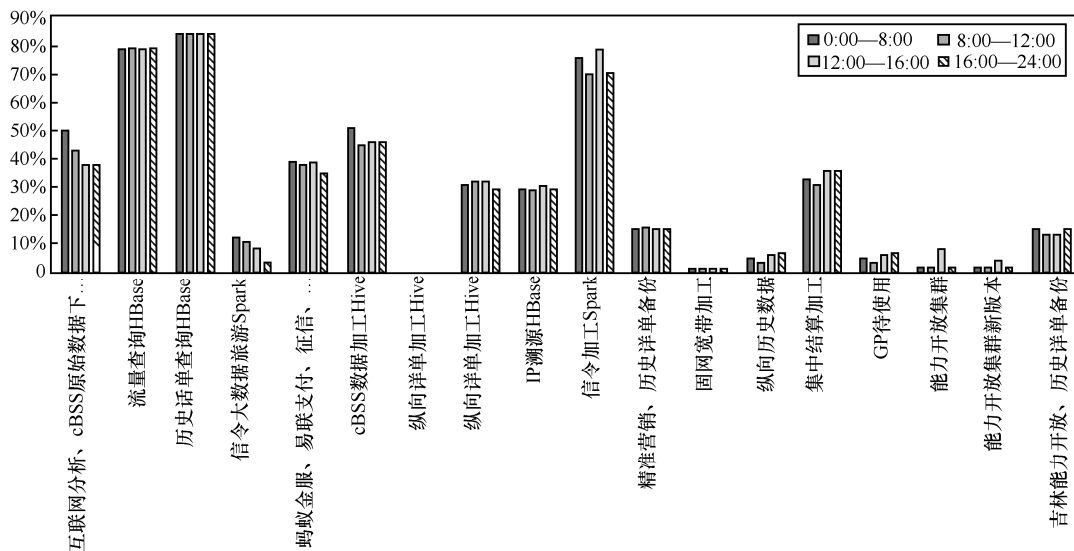


图1 不同作业的内存分时段使用率统计

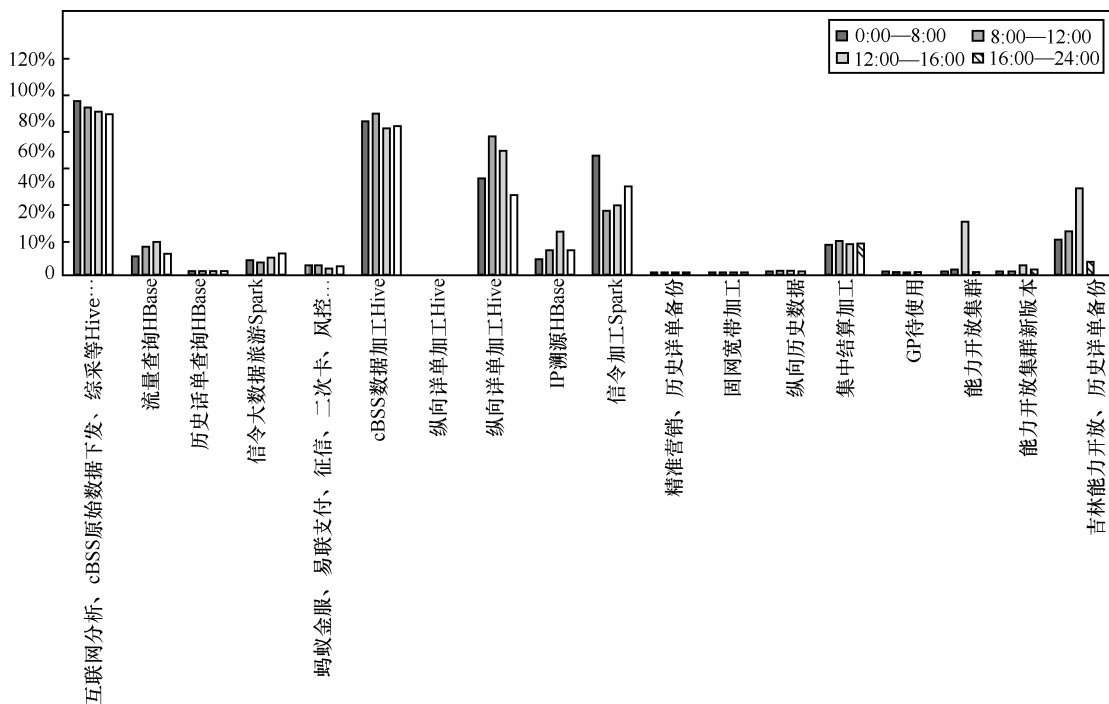


图2 不同作业的CPU分段使用率统计



问题,主要体现在以下几点:

- 集群分布在多个机房,业务增长过快导致频繁的数据迁移,针对性的平台性能优化不足;
- 因 GP 故障、生产性能不足等问题,自 2015 年至今经历多次数据迁移,存储资源不足及必要的数据留存周期造成无法针对性做进一步优化;
- HBase 库在数据量增大后,读的概率下降,查询效率会逐步降低,需要持续优化保障,其他业务需求增多,与核心生产服务资源征用情况相比较明显,直接影响到 cBSS 原始明细下发、对外实时查询等服务;
- 单个业务团队运维人员投入急剧增加,尤其需要增加实际业务自动化运维监控功能,落地自动化运维监控平台,减少人员运维成本;
- 如图 1、图 2 所示,如果能将不同主机的主机资源共享,将大大提高主机资源的使用率。生产系统中,每一个计算集群都部署在专门的物理机中,计算任务完成后,则会闲置,而其他的计算集群也无法使用该主机资源。原有的技术结构存在的问题是缺乏纯物理机资源的粗、细颗粒度的调度能力。

本文研究填补了中国联通没有统一的容器化资源运维平台的空白,让实际业务应用在无状态化、容器化之后,统一管理业务应用的生命周期。打通现有运维模式,统一运维所有的业务应用资源,使现有的开源技术有机地和联通传统业务相结合。

3 国内研究现状

网状网作为整个一级业务支撑系统的核心系统,是中国移动内外部信息传输交换、服务管控、数据处理、业务支撑、运营开放为一体的综合信息交换枢纽。目前承载 200 多个平台的接入,支撑业务达到 2 000 多个,系统承载业务具有容量

大、实时性强、波动剧烈、增长迅速、重要性强、客户影响大、无状态业务居多等特点,非常适合做 PaaS 平台的试点。业务支撑中心和网状网项目技术团队经过大量的研讨,创新性地提出了应用进程单元(application process unit, APU)的概念,把资源和应用有效地结合在一起,解决未来系统发展和管理的瓶颈。而且通过深入的技术研究和实践探索,在 Docker 基础上通过增强接口和管理功能,实现了 APU 概念的落地。结合 Kubernetes 作为集群管理平台,搭建了能够承载网状网系统的 PaaS 平台试点,实现了整个平台的容器化改造和集群的部署、管理和监控。

4 容器运维系统的设计

4.1 功能架构及其设计思路

一般地,较设计需求来说,容器运维系统是面向生产系统各应用提供大数据微服务化能力管理、调度和开放化运营的管理框架,并面向不同的业务部门和信息域的需求,实现多租户管理的能力开放。因此在设计容器运维系统的功能时,需要综合考量。核心的必要功能罗列如下。

(1) 容器管理

需包括整个容器的生命周期管理,即容器服务的部署创建、容器服务配置的修改管理、容器服务的销毁、容器服务的应用监控。为适应不同的用户,还应具备服务访问的权限^[1]。如管理员可以访问所有的容器实例,而单个用户只能访问自己创建或者同组之间赋予权限的容器实例。各个分组之间应是完全物理隔离或逻辑隔离的^[2]。

(2) 框架管理

也需包括整个框架的生命周期管理,即框架服务的部署与创建、框架服务配置的修改管理、框架服务的销毁、框架服务的监控。需要与容器有所区分的是,框架相对容器的生命周期较长,且一般都是相对复杂的应用,交付方式也不单是

网址端口如此简单明确。很多业务场景下，如Hadoop的能力，其创建不应由用户自行创建，更多的业务场景则是用户来发起工单、申请能力、管理员开通、交付使用。所以框架的生命周期管理应由管理员掌控^[3]。

(3) 集群管理

相对地，也需要包括集群的全生命周期管理，如集群的创建、集群节点数量等服务配置的修改管理、集群的销毁与监控^[4]。

其次需要包括完整的工单系统，包括用户和管理员这两个基本权限角色，同时为了管理更有序，还需要对用户进行分组，实现组间、组内多租户更细化地分配资源。作为工单系统，角色的注册和权限管理也是必不可少的。另外还需要对已有的软件资源（物理机、框架、镜像）设计相对应的管理功能。

- 镜像仓库：镜像的添加或删除。
- 框架仓库：框架的添加或删除。

- 物理机资源：资源纳管，将物理机纳入策略设置^[5]。

除了以上的核心功能，还需根据实际业务情况设计针对性的进阶功能，如针对快速迭代的DevOps管理、容器存储的持久化、一键部署所需的应用包功能。经过功能梳理后，容器运维系统的功能如图3所示。

4.2 容器运维系统的技术路线

经过前期的技术选型通过集成已有的开源软件技术可以完成以下功能的搭建。

- DCOS：集群生命周期管理、框架仓库、物理机资源纳管。
- Kubernetes：容器生命周期管理、租户管理。
- Mesos：资源调度。
- Marathon：框架生命周期管理。
- Harbor：镜像库管理。
- Nginx&Flannel：服务注册、负载均衡、访问鉴权。

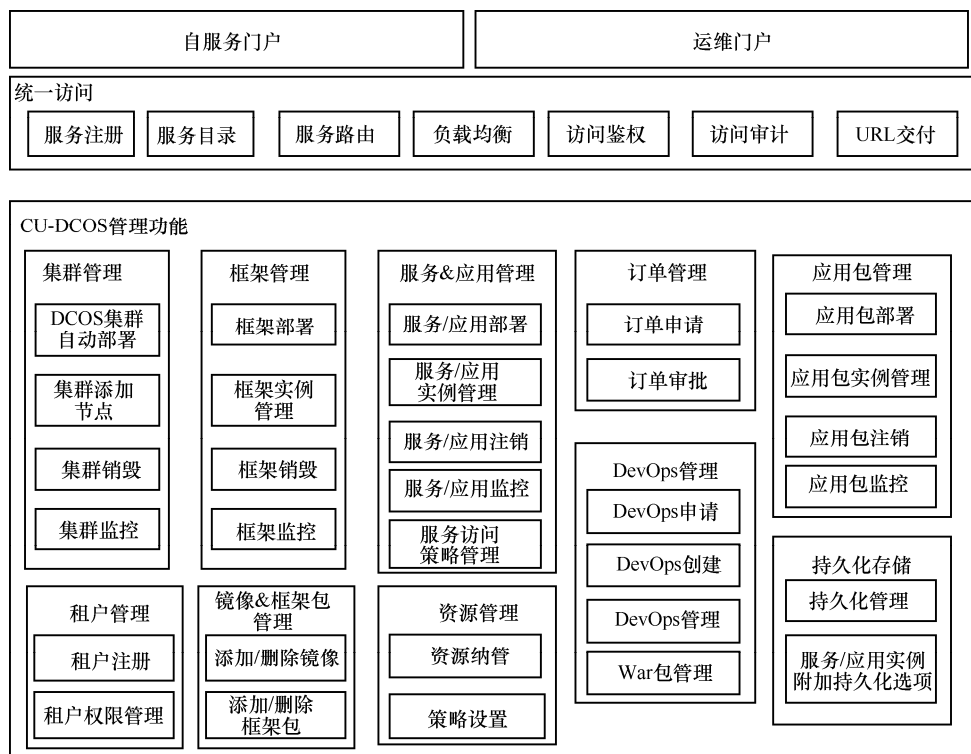


图3 容器运维系统功能



- Helm: 应用分组管理。
- Ceph: 容器持久化存储。
- Jenkins: DevOps 迭代开发。
- Heapster&&EFK: 监控容器运行情况并输出监控日志。

容器资源运维系统使用 DCOS+Marathon+Kubernetes 的架构模式(如图 4 所示),完成集群、框架、容器三大维度生命周期的管理。容器资源运维系统从技术架构上分为接入层、服务层、框架调度层和前台的管理系统^[6]。

- 接入层: 负责服务发现和服务路由,把每个容器业务应用串联起来。
- 服务层: 应用 Helm 技术,把每个用户自定义的容器配置保存起来打包,以便二次部署;基于 Jenkins 技术,在线编译程序,快速构建验证与开发环境,实现应用的容器化封装。
- 框架调度层: 应用 Heapster,实现容器的

健康度实时监控;EFK 完成系统日志层面的搜索、输出、展示;Kubernetes 完成容器的生命周期管理;Marathon 完成框架的生命周期管理;Mesos 提供这两个模块的资源调度。

- 前台管理: 用 Java 编写前台的客户端,调用现有技术的 API,实现所有技术的调用。

5 关键问题技术 Mesos

5.1 Mesos 的资源调度改进设计

容器资源运维系统针对主流的 Hadoop 系列大数据应用,采用两级调度策略,由 Mesos 发起资源分配,实际的应用框架消费资源,合理化使用物理机资源,如图 5 所示,做到资源空闲时,大数据应用集群抢占系统资源;资源紧张时,大数据应用自动释放资源,并且不影响实际业务使用。这其中包括改进的 Mesos 资源配置机制,如图 6 所示。

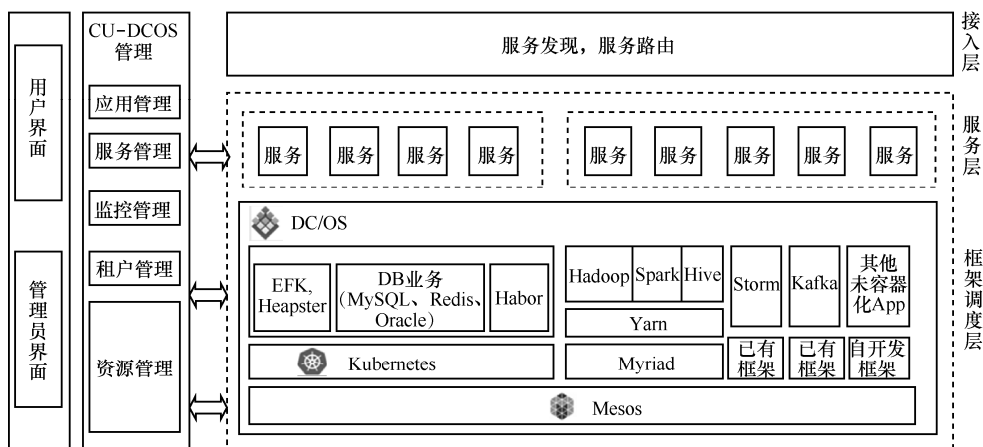


图 4 系统技术架构

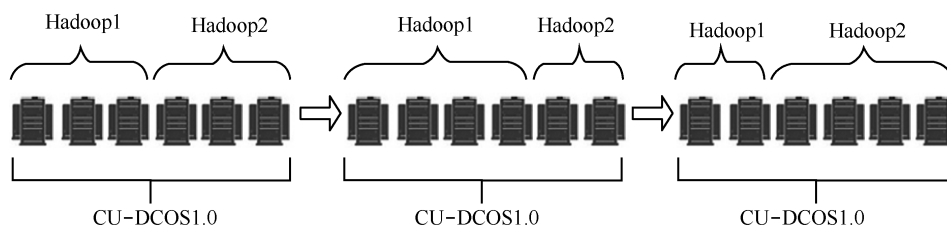


图 5 Hadoop 集群资源调度实例

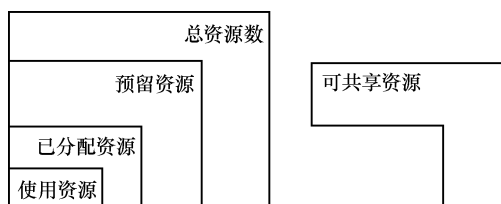


图6 Mesos 资源机制

- 预留资源：由特定框架或角色静态或动态声明的资源，包括一些已分配的资源。
- 已分配资源：已提供给框架或由框架接受并作为执行程序/任务提供的资源。
- 可撤销资源：可以由分配器或代理在任何时间/短期通知回收的资源以及这些资源上的容器将被销毁。可撤销资源可以由分配器和代理使用超额预订生成。在代理使用超额预订的上下文中，“可撤销”的含义还包括容器可能经历节流的东西（例如较低的 CPU 份额、较少的网络优先级等）^[7]。

5.2 Mesos 的任务机制改进设计

- 正常任务：在不可撤销资源上运行的任务，与 AWS 保留实例相同。
- 可撤销任务：低优先级和易碎的任务，如 AWS spot 实例。
- Lender 框架：一个运行正常任务并保留资源的框架，这个框架可能没有充分利用那些保留的资源，这就像目前存在的任何框架。
- Tenant 框架：一种运行可撤销任务并利用 Lender 框架中的资源的框架的新范例。这种类型的框架可以处理不稳定/笨拙的资源和更高的任务故障率。租户框架需要具有启用“REVOCABLE_RESOURCES”的能力。

如图 7、图 8 所示，Lender 框架决定使用其保留的资源时，使用来自分配器所提供的资源来启动任务。如果任务不适合当前可用的资源，则代理首先通过杀死正在使用可撤销的不可节制的资源的执行器来驱逐一些任务^[8]。这可以由预订超额预订和动态预留之间的交互触发。在框架或

操作者移除一些保留的资源之后，同时由“被移除的保留资源”所生成的“可撤销非节流资源”正由任务/执行器使用，则可撤销的不可节制资源总量也将在代理上减少^[9]。

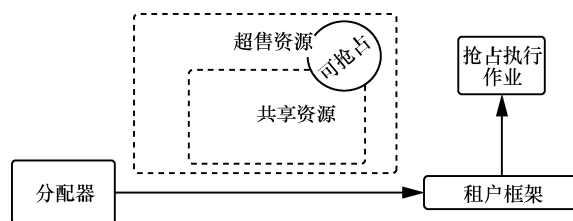


图7 驱逐保留资源任务

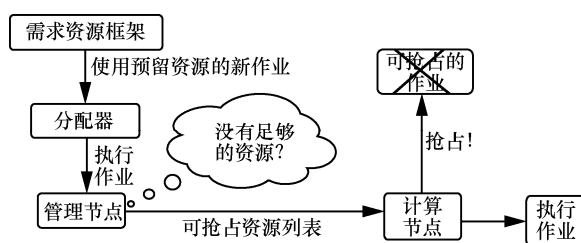


图8 驱逐正常资源任务

6 结果验证

本文以 CentOS 7 为验证环境，验证了主机 10.1.24.108、10.1.24.39、10.1.24.137、10.1.24.234 这 4 台 x86 机架服务器，集成 Docker (1.11)、Hadoop2.7、Kubernetes1.5、Open DCOS 1.05、Mesos 1.0.0 等开源软件设计实现容器运维系统。

(1) 资源调度验证

由于设计方案侧重大数据的资源调度。为了对比容器运维系统与 Open DCOS 两个调度大数据资源的方法，采用实际验证法对两个方法进行了验证。结果如下。

首先，验证了两个系统对于 Myriad 框架对细粒度资源的支持。目的是验证 Myriad 框架对执行任务时，按照细粒度模式申请资源，任务运行时资源的利用情况。

实际验证结果如下。

Open DCOS：细粒度执行时间为 165.476 s，



固定模块执行时间为 147.648 s。

CU-DCOS 容器运维系统：细粒度执行时间为 164.693 s，固定模块执行时间为 144.976 s。

通过验证发现，在资源模板配置零的情况下，计算任务的时间比固定资源配置要长一些，且两种方法都支持 Myriad 框架对细粒度资源进行调度。

其次，验证了两个系统对于资源预留功能方面的功能实现。目的是验证 Myriad 的基础资源预留功能以及在这个基础上的附加功能，包括动态资源预留、资源抢占等。

在 Mesos-slave 机器上配置资源选项做配置，为 CPU、内存、角色等进行资源静态预留。在 Myriad 框架上，调用容器运维系统新增的 RESTful

接口发起动态预留信息，将资源等配置资源到固定机器上。

实际验证结果如下。

Open DCOS 和容器运维系统均支持动态、静态资源预留，Open DCOS 对资源抢占功能不支持，但容器运维系统对于资源抢占功能支持。集群资源抢占效果如图 9 所示，图 9 中集群 1 有 2 台主机，集群 2 有 2 台主机。当集群 1 压力较大时，会抢占集群 2 较为空闲的主机资源参与计算，如图 10 所示。

(2) Mesos 运行模式验证

验证结果发现应用框架方便实用，并且可以根据需要自行开发。容器化需要对应用进行无状

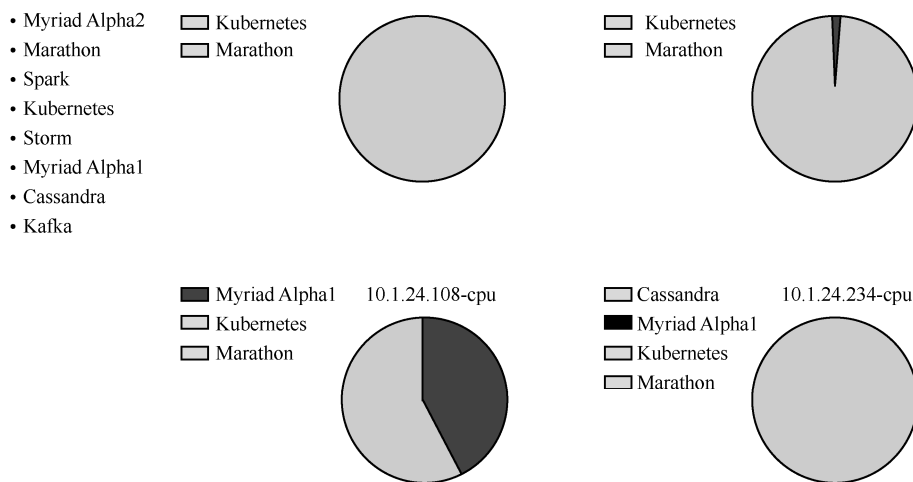


图 9 容器运维系统资源监控

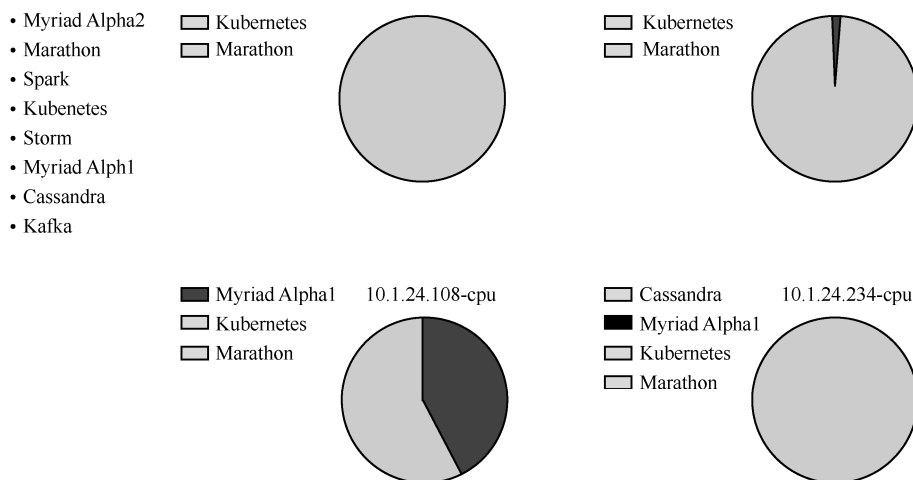


图 10 集群 1 压力较大时容器运维系统资源监控

态改造工作。改造工作主要针对以下 3 个有状态内容：会话方面使用 Redis 缓存或 Memcached；日志使用日志服务器 Graylog；非日志数据文件通过外挂存储的方式移至持久化存储。

（3）基于 Mesos 的未容器化的应用解决方案

验证结果发现部分业务暂时难以容器化改造的应用，特点为：有状态、体量大、自身封闭、难以微服务化，且无响应应用框架，如核心应用、Oracle 等。

基于 Kubernetes 的大颗粒度 Docker 容器：使用大颗粒度容器承载整个应用。这样便于部署和管理，可以提高资源利用率，持续集成（DevOps），且 Docker 占用资源极少。

使用 Marathon 支持的 Mesos-容器。Mesos-容器比 Docker 更轻，只针对进程用 CGroup 做 CPU、内存的隔离，不对文件系统、网络等进行限制。不需要使用镜像启动，直接运行命令行启动该容器。

（4）技术路线对比验证

Kubernetes 相较于 Marathon（见表 1）有更好的潜力，在容器化调度方面，支持证书，支持容器的持久化卷等额外的扩展功能，且在多集群和容灾方面都比 Marathon 有更好的安全性保证。总体来说，Kubernetes 更适合容器的编排和调度。在 Mesos 方面需要使用开源技术和自行研发结合方式实现，其中对资源的统一管理和调度采用开

源的 Mesos 来支撑，但缺乏框架的自动安装部署和门户等附加功能，需要在稳定的 Mesos 版本上深度研发。

7 结束语

容器运维系统旨在通过新一代的云计算架构——容器技术，解决联通面临的实际问题。在解决问题过程中，依赖于开源软件通过自主架构、自主方案设计，部分核心能力自主研发；形成了面向生产环境下的 IT 系统，实现资源的一体化调度、大数据微服务化管理能力；完成 IT 资源的集中管理的新一代的私有云平台系统。一方面验证了以容器为基础的 PaaS 平台从模式、到技术的可行性，另一方面在行业内首次实现了面向大数据、物理资源弹性调度、多租户管理的“资源+数据+能力”的平台架构。

参考文献：

- [1] BAIER J. Getting started with Kubernetes[M]. Birmingham: Packt Publishing, 2015.
- [2] BERNSTEIN D. Containers and cloud: from LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2015, 1(3): 81-84.
- [3] HINDMAN B, KONWINSKI A, ZAHARIA M, et al. Mesos: a platform for fine-grained resource sharing in the data center[C]// USENIX Conference on Networked Systems Design & Implementation, April 2-5, 2010, Lombard, IL, USA. New York: ACM Press, 2010: 429-483.
- [4] 吴龙辉. Kubernetes 第 1 版实战[M]. 北京: 电子工业出版社,

表 1 技术对比验证

	Kubernetes	Marathon
功能		
灰度发布	已经支持（rolling-upgrade）	部分支持（还需要手动操作）
容器编排	支持 POD 方式	不支持 POD 方式
容器安全	支持现有的容器安全机制比如 SELinux 等	仅支持 Linux User 这种初步的安全机制
证书管理	提供对 Docker Hub 等证书的管理机制	不支持
持久化卷管理	支持（PV,PVC）	部分支持，相应的调度策略还在开发中
多机群（容灾）	支持（Kubernetes）	不支持
生态	Mesos、Docker	Mesos、Docker
匹配度	强，k9s 在容器调度方面提供的功能更为完善	强，Marathon 近些年随着 DC/OS 的发展，功能也在向 k8s 逐渐拉齐，另外 Marathon 可以通过进程调度的方式直接部署一些非 Docker 化的应用
前景	开源软件，k8s 主要是谷歌系，另外社区和企业里 HPE 等大力在推广，有广泛的商业应用	开源软件，Marathon 主要是 Mesosphere 在大力推广，有广泛的商业应用



2015: 13-211.

WU L H. Kubernetes application 1st ed[M]. Beijing: Electronic Industry Press, 2015: 13-211.

- [5] SIERRA K, BATES B. O'Reilly: head first java(中文版)[M]. O'Reilly Taiwan 译. 北京: 中国电力出版社, 2007.

SIERRA K, BATES B. O'Reilly: head first Java[M]. Translated by O'Reilly Taiwan. Beijing: China Electric Power Press, 2007.

- [6] KAKADIA D. Mesos 大数据资源调度与大规模容器运行最佳实践[M]. 崔婧雯, 刘梦馨, 译. 北京: 电子工业出版社, 2015.

KAKADIA D. Applications on Mesos[M]. Translated by CUI J W, LIU M X. Beijing: Electronic Industry Press, 2015.

- [7] 曹旭, 曹瑞彤. 基于大数据分析的网络异常检测方法[J]. 电信科学, 2014, 30(6): 152-156.

CAO X, CAO R T. Network anomaly prediction method based on big data [J]. Telecommunications Science, 2014, 30(6): 152-156.

- [8] 漆晨曦. 立足小数据基础的电信企业大数据分析应用发展策略[J]. 电信科学, 2014, 30(10): 15-20.

QI C X. Analysis and application development strategy of telecom enterprise big data based on small data [J]. Telecommunications Science, 2014, 30(10): 15-20.

- [9] 韩晶, 张智江, 王健全, 等. 面向统一运营的电信运营商大数据战略[J]. 电信科学, 2014, 30(11): 154-158.

HAN J, ZHANG Z J, WANG J Q, et al. The unified-operation-oriented big data strategy for telecom operators[J]. Telecommunications Science, 2014, 30(11): 154-158.

[作者简介]



张勋 (1990-), 男, 中国联合网络通信有限公司研究院大数据研究中心软件开发工程师, 主要研究方向为开源技术、容器技术等。



张呈宇 (1988-), 男, 中国联合网络通信有限公司研究院大数据实验室大数据平台技术研发组组长, 主要从事大数据、云计算 IaaS、容器技术等研发工作。



魏进武 (1978-), 男, 博士, 现就职于中国联合网络通信有限公司研究院大数据研究中心, 主要负责大数据架构设计、规划、实施方案, 统一云化研究等的研究工作。