POP:一个基于微服务架构的在线编程系统

胡 星 王泽瑞 李 烁 杨 楠 张知凡 王 巧 王千祥
(北京大学信息科学技术学院高可信软件技术教育部重点实验室 北京 100871)

摘 要 随着云计算的发展,基于云端的编程模式越来越受到开发者的青睐。在线编程系统与PaaS平台相结合,可以大大简化应用开发过程,为开发者提供便利。Docker的出现推动了PaaS平台的迅猛发展,Docker的种种特性给予了在线IDE更加理想的开发部署应用的环境。POP(Public Online Programming)是一种利用Docker技术实现的基于微服务架构的公共在线编程系统。POP通过对Docker资源的合理调度管理,使得在线编程系统在部署、调试和运行各类应用时能够更加节省资源和时间。

关键词 云计算,公共在线编程系统,微服务,Docker
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.002

POP:Micro-service Based Online Programming System

HU Xing WANG Ze-rui LI Shuo YANG Nan ZHANG Zhi-fan WANG Qiao WANG Qian-xiang
(Key Lab of High Confidence Software Technologies, ME, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract With the development of cloud computing, more and more developers prefer programming based on cloud. Combined with the PaaS platform, online programming system will greatly simplify the application development, providing great convenience for developers. The emergence of Docker promoted the rapid development of PaaS. All the features of Docker are fit for online IDE to install and configure completely. This paper introduced POP (Public Online Programming) which based on Micro-service using Docker. In this paper, monolithic architecture app was broken into several services. Each service was running in an independent Docker container. Each component evolved on its own in Micro-service architecture. It reduces the evolvement risks. Through the Docker management and scheduling, POP can allocate Dockers for different types of applications to deploy, debug and run as soon as possible and minimize resources.

Keywords Cloud computing, Public online programming, Micro-service, Docker

1 引言

随着云计算的发展,越来越多的本地应用被迁移到云服务平台尤其是PaaS[1](Platform as a Service)。PaaS为开发人员提供了高层的运行环境,其只要将自己的Web应用上传到PaaS上,就可以将应用部署在云中展现给终端用户。在PaaS平台的支持下,Web应用的部署与运行变得更加敏捷,并能够对需求的变更做出快速响应。

软件开发系统作为软件的一个种类,自然也可以迁移到云服务器上,以支持编程人员通过浏览器来开发软件。在线编程系统[2]即是这样一种基于云的编程系统[3],开发者可以通过浏览器进行应用软件的编辑、编译、调试甚至部署。

在线编程系统可以为PaaS平台应用的开发者提供进一步的便利[4]。应用的编写、调试与部署过程全都可以在云端进行,有效地简化了应用的开发与运维过程:开发者在PaaS平台创建应用后,可以直接进入在线编程系统进行应用开发,并通过在线编程系统完成应用的部署与发布操作。在国外,Cloud9和Codenvy等在线编程系统都不同程度地支持将应用发布到PaaS平台上。

本文利用Docker[5]技术设计并实现了一个基于微服务架构的公共在线编程系统——POP。POP的构件部署在一组容器中,构件之间通过服务化的方式进行交互,通过对容器的调度管理有效地提高了容器的部署效率,并方便了系统中各个构件的独立演化。

2 相关工作

Docker的影响力不仅得到了行业内许多大企业(包括Amazon, Canonical, CenturyLink, Google, IBM, Microsoft[6], New Relic, Pivotal, Red Hat和VMware)的支持,同时也为许多初创公司提供了茁壮成长的平台。越来越多的合作围绕Docker开展,直接或间接地推动了Docker生态环境的快速发展。Amazon在云计算领域一直处于领导地位,作为AWS(Amazon Web Services)产品的Elastic Beanstalk和Amazon EC2 Container Service都已经开始支持Docker[7]。

到稿日期:2015-11-30 返修日期:2016-02-28 本文受863高技术项目(2013AA01A213),国家自然科学基金创新群体项目(61121063)资助。
胡 星(1993-),女,博士生,主要研究领域为软件运行监控;王泽瑞(1989-),男,硕士生,主要研究领域为运行环境管理;李 烁(1990-),男,硕士生,主要研究领域为软件工程;杨 楠(1991-),女,硕士生,主要研究领域为软件工程;张知凡(1993-),男,硕士生,主要研究领域为软件工程;王 巧(1994-),女,主要研究领域为软件工程;王千祥(1970-),男,博士,教授,CCF高级会员,主要研究领域为软件工程、中间件。

除此之外，在线编程系统也在开始尝试利用 Docker 来提升自己的性能。Cloud9 IDE[8] 为开发者提供了可以在云计算环境中测试、部署开发者编写代码的在线 IDE。2014 年 7 月，Cloud9 IDE 推出新版本，新版本通过 Docker 运行在 Linux 容器中，每个工作区都是一个完全独立的虚拟机，运行时也不需要考虑配置问题。Codenvy[9] 作为一个云 IDE，与 Cloud9 一样都支持多种语言的开发、编译与发布等。这些云 IDE 使得应用程序在 IDE 中可以往不同 PaaS 上发布：Amazon(Elastic Beanstalk)，Heroku，GAE 等。Cloud9 和 Codenvy 这些项目均对常用的 PaaS 平台给予了支持。

微服务架构在近几年也受到广泛关注，微服务具有模块化、分布式、独立部署等特性。国内外有许多利用微服务的成功案例，例如国外的 Amazon，Netflix 和国内的阿里。Docker 是目前最具代表性的容器技术，许多微服务的技术方案都采用了 Docker 技术。利用 Docker 技术能够很好地实现微服务架构，独立的微服务借助 Docker 运行在容器中。

3　POP 概述

公共在线编程系统即 POP 是一款支持多语言编辑、编译、运行、部署、面向真实项目开发的在线编程工具，能使开发者方便且快速地在线编写、调试、部署应用程序。POP 不仅支持关键字高亮、自动缩进、编译和运行的基本功能，还拥有在线调试及上传到 PaaS 平台等高级功能。

3.1　POP 功能概述

POP 支持 Java，PHP，Python 和 Node. js 4 种应用的开发和部署运行。用户可以通过 POP 在线编辑、编译和部署开发的应用。不仅如此，用户还可以使用 GIT 等版本控制工具，将代码上传到 PaaS 平台上，以及在 Paas 平台上自动更新和下载代码。

3.2　微服务架构

微服务指采用一组服务来实现一个应用，各个服务独立运行于不同的进程中，不同的服务通过轻量级的交互机制(例如 http)进行通信[10]。微服务架构相对于传统的单体式应用开发更有利于应用的持续性开发。对于微服务，模块的扩展对整个系统是独立的，这在很大程度上降低了系统的耦合性。

微服务架构便于 POP 各个模块的独立开发，保持了系统整体的稳定性。不仅如此，POP 各构件之间还实现了各自的演化，POP 系统每个构件的更改、测试和部署都独立于其他构件，从而降低了传统结构在演化时面临的系列风险，例如系统构造错误、部署错误等。

POP 利用 Docker 容器技术来实现其微服务架构，大大简化了容器化微服务的创建、集成、部署、运维等过程。POP 的每一个功能都以微服务的形式完成，每一个微服务实例都对应着一个 Docker 容器。每个容器都提供了构件运行的完整环境，不需要依赖外部的任何东西，从而创建出了高效的分布式系统。采用微服务架构，POP 的每个功能模块通过容器相互隔离，独立演化，当一个功能服务出现故障时，其他功能服务不会受到影响，从而提高了系统的稳定性。

POP 的微服务主要包括：用户入口服务、各种语言代码编辑服务、总控管理服务、运行部署服务等。各个服务之间各自开发，降低了系统的耦合性。

POP 的微服务结构如图 1 所示。

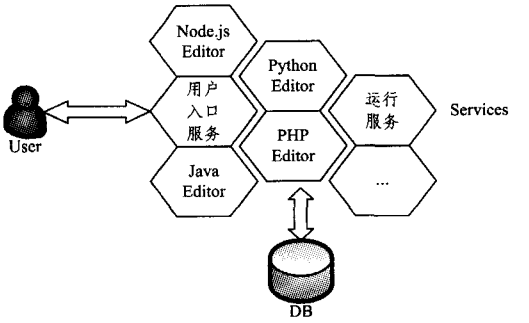

图 1　POP 的微服务结构

4　系统设计

4.1　POP 整体设计

POP 由固定容器、弹性容器、数据库以及静态缺陷分析、代码推荐等共性服务构成。固定容器包含入口容器、总控容器、4 个编辑容器(Java，PHP，Python，Node. JS)和包含 4 种运行环境的运行容器：Java(包括 Java debug)，PHP，Python，Node. JS。POP 的整体设计如图 2 所示。

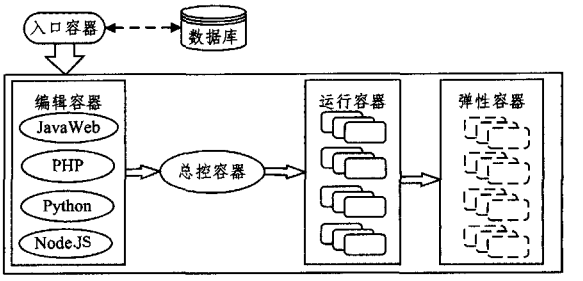

图 2　POP 的整体设计

1)入口容器是 POP 系统的总入口，负责管理用户、项目等，并包含 Maven 库。

2)编辑容器负责程序员对代码的编辑等操作，并在不同子目录下放置不同程序员的不同项目代码。POP 在编辑容器中为每种类型的应用提供用于体验的示例代码，便于新手学习。

3)总控容器负责对运行容器进行调度管理。用户要部署应用时，编辑容器会通知总控容器，由总控容器统一为编辑容器要部署的应用分配运行容器。

4)运行容器中配置了运行应用所需要的环境，用于应用的编译、运行与调试，由不同的 image 创建。本文在每个运行容器中均安装了 4 种运行环境，不同的运行容器在同一时间的运行环境不同，由总控容器负责统一调度，进行运行环境的切换。这里将 Java 应用的调试环境作为一种特殊的运行环境来进行配置，从而达到运行容器的统一管理。

5)弹性容器的功能与运行容器相同，主要用于调试、运行任务变化时保证服务质量，并节约资源。不同类型容器的弹性策略不同。

6)数据库记录系统的各类信息：各个程序员的登录、退出时间，当前使用的开发环境，当前开发的项目，容器资源使用情况等。

4.2　POP 的基本功能

POP 具有常规 IDE 编辑代码、编译、调试等功能，页面包括工具栏、项目管理面板、代码编辑区、大纲面板、控制台等。除此之外，它还提供了应用的部署运行功能，同时还支持对

PaaS 平台的相关操作。

1)代码编辑功能

POP 实现了代码编辑、语法高亮、自动换行等功能，支持快捷键的操作。当代码编辑完成，用户点击保存时，浏览器端将向服务器端发送一个请求，其中包含了保存文件的路径与修改后的文件内容。服务器端收到请求后，修改对应文件的内容，完成文件保存。

2)项目管理功能

包含的操作有：对文件的存、取，对文件内容的修改；对项目、文件夹或文件的创建、删除、重命名。项目管理功能的操作通过服务器端的文件操作进行。用户在浏览器端执行某项操作后，相应的请求被发送到服务器端，服务器端根据请求的内容执行相应操作。之后，浏览器端通过更新项目管理面板，将更新后的项目文件列表显示给用户。

3)部署运行功能

用户将代码编辑完成后，点击“Build”进行编译，编译通过后点击“Run”将应用部署到系统为应用分配的运行容器上，点击系统返回的网址即可看到应用的运行结果。

4)代码托管功能

POP 提供在 PaaS 平台上导入应用和提交应用的功能。利用在线编程的优势，POP 将版本控制工具的功能进行了封装，从而提供了用户代码的自动下载、更新和提交功能。相比于本地开发，POP 能够简化用户编程以外的操作，使得开发过程更加方便。

5 关键技术与实现

5.1 POP 容器中的交互机制

POP 利用 redis 来进行通信。redis 的通信机制类似于设计模式的观察者，发布者可以发布某种类型的消息，消息的类型以 Channel 的方式来呈现，订阅者将以订阅事件的方式来接收消息。本文将 redis 这种通信机制用于容器之间的通信，各个容器之间的交互如图 3 所示。

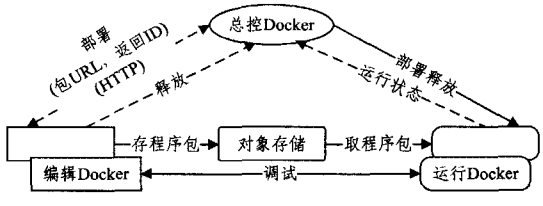

图 3 POP 容器交互图

1)总控容器通过 redis 发送部署、释放容器的消息，并运行容器通过订阅该通道得到消息，并执行相应的操作。

2)本文利用对象存储来转发需要部署的程序包，编辑容器将用户编辑的代码打包存储，运行容器根据对象存储地址来获取相应应用的代码进行部署。

3)运行容器接收到代码包后安装、启动目标程序。如果调试项目，则运行容器同时启动一个线程，用于与编辑容器进行通讯。该线程通过 redis 将项目调试时产生的信息发送至编辑容器，并接收来自编辑容器的调试控制命令，执行相应命令，完成调试过程。

4)编辑容器在部署完程序包重新编辑时，发出释放命令，以减少运行容器的占用。

5)本文定义了两个 channel，分别为“control”，“report”。“control”用于总控容器向运行容器发送命令(如部署应用、释放资源等)，运行容器通过订阅“control”可以获得总控容器发出的命令，从而执行相应的操作；同时运行容器也可以通过“report”向总控容器发送消息，总控容器通过订阅“report”就可以获得运行容器相应的状态变化。

5.2 运行容器的管理机制

在 POP 的设计中，每个运行容器都有 4 种应用运行时所需要的运行环境，所以应用部署时运行容器的选择、运行容器的释放都需要进行管理[11]。

5.2.1 运行容器初始管理机制

考虑到 Java，PHP，Java debug 3 种类型的应用使用较为频繁，并且开启服务器需要一定的时间，本文提出了在容器充足的条件下为这 3 种应用预留服务器，即在初始状态下，Java，Java debug，PHP 这 3 种应用的服务器为开启状态，以便应用能够尽早部署，节省服务器的开启时间。参数定义如表 1 所列。

表 1 运行容器初始化参数表

runner_type	state	代表含义
Java	Ready	运行 Java 应用的服务器已经开启，没有应用部署
Java debug	Ready	调试 Java 应用的服务器已经开启，没有应用部署
PHP	Ready	运行 PHP 应用的服务器已经开启，没有应用部署
None	Ready	当前容器没有开启服务器，也没有应用部署

5.2.2 部署应用运行容器调度管理机制

基于 5.2.1 节，因为每个容器在初始时的参数不同，所以在为每个应用分配容器时有不同的策略，不同的应用在每个容器中的部署也有所差别。针对不同容器结合应用的类型，本文提出以下调度策略。

1)当要部署 Java，Java debug，PHP 类型的应用时，首先查找相应服务器开启但没有部署应用的容器进行应用的部署，之后再查找 runner_type 为 None、state 为 Ready 的容器并开启此类应用的服务器，以备下次部署。

2)若 1)查找失败，则查找 runner_type 为 None、state 为 Ready 的容器，开启相应的服务器，部署应用。

3)若 2)查找失败，则查找 runner_type 不为 None、state 为 Ready 的容器，先关闭已经开启的服务器，然后开启将要部署的应用类型的服务器，部署应用。

4)若 3)查找失败，则表示当前没有空闲的容器，返回等待资源的消息。

5)当部署 Node.js，Python 类型的应用时，首先查找 runner_type 为 None、state 为 Ready 的容器，部署应用。

6)若 5)查找失败，则查找 runner_type 不为 None、state 为 Ready 的容器，关闭开启的服务器，部署应用。

7)若 6)查找失败，则表示当前没有空闲的容器，返回等待资源的消息。

调度策略在确定了要部署应用的容器后，首先判断容器的状态，以防容器重复部署应用；部署应用前准备应用所需的运行环境，如开启相应运行环境的服务器；部署应用时，从对象存储中获取应用代码之后部署到容器中；最后将容器的运行结果链接返回给用户。运行容器部署应用的算法流程如图 4 所示。

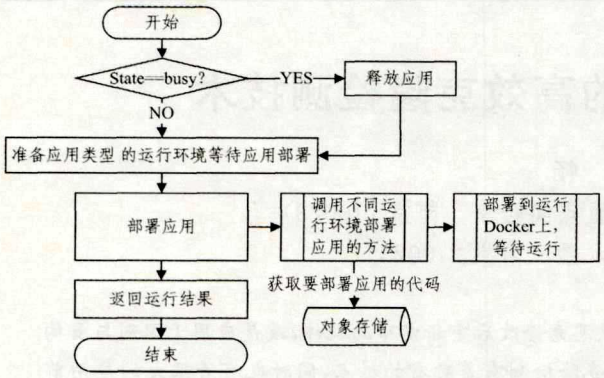

图 4　运行容器安装应用流程

5.2.3　运行容器资源回收机制

由于容器资源数量远远少于需要部署的应用的数量，为了使容器资源能够得到有效的利用，本文提出了用阈值来衡量某个运行容器是否需要被释放。当前系统时间与上次访问时间的差如果大于该阈值，则表示该容器长时间未被访问，需要被释放以避免容器资源的浪费。其中的访问包括：部署应用、调试应用，每次访问过后更新最后使用容器的时间。

最后的访问时间由运行容器自己维护的信息和访问日志 access. log 的最近访问时间确定。由于每个运行容器有 4 种运行环境，不同的环境其 access. log 不同，因此要考虑不同环境日志的读取方式并且要忽略服务器自己维护的状态信息，从而筛选出开发人员在部署应用后真正的访问时间。

在释放资源时，不同的运行容器依其运行环境的不同有不同的释放策略。对于运行着 Java，Java debug，PHP 应用的容器来说，只需释放其应用，而不用关闭其服务器；而运行着 python，node. js 应用的容器不仅要释放应用，还要关闭其服务器。运行容器释放资源的算法流程如图 5 所示。

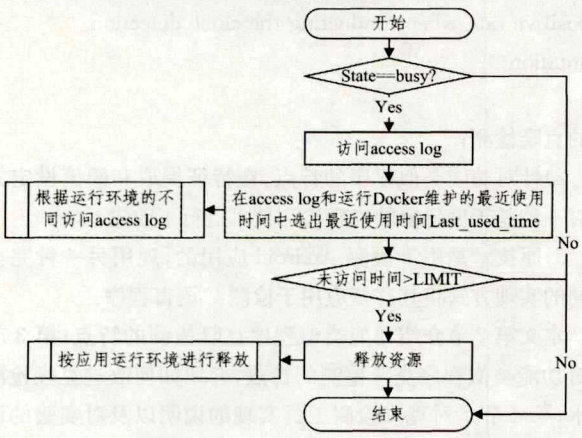

图 5　运行容器资源释放流程

5.3　系统实现

POP 原型系统运行于 BAE（Baidu App Engine）之上（pop2015. duapp. com），并利用了 BAE 提供的容器、通信机制以及百度对象存储。POP 的入口界面及 Java 语言的编辑界面分别如图 6、图 7 所示。系统已经在教育部多个教指委联合主办、BAE 提供运行平台的 2015 年全国计算机设计大赛中得到了初步应用。参赛选手可以方便地在 BAE 上创建应用，然后进入 POP 开发应用，并直接将应用部署到 BAE 上。

图 6　POP 的入口页面

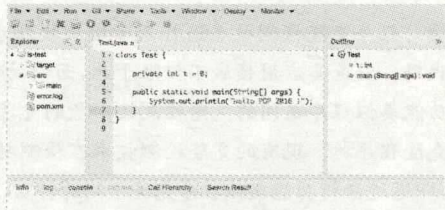

图 7　Java 语言的编辑页面

结束语　随着云计算技术的发展，越来越多的本地应用被迁移到互联网上，通过在线服务的方式提供给用户。通过将本地开发环境迁移到云端，在线编程系统使得开发者可以通过浏览器进行应用软件的开发，并可以将编写好的应用直接上传至 PaaS 平台并发布。本文利用微服务技术设计并实现了基于容器的在线编程系统。微服务架构保证了系统开发演化的稳定性。利用容器来实现在线编程系统的编辑、编译、运行、部署的流程，并且在一个容器中安装多种运行环境，通过运行环境的切换实现对容器的有效调度。

参 考 文 献

[1] PaaS[EB/OL]. (2016-5-5)[2016-5-13]. https://en. wikipedia. org/wiki/Platform_as_a_service.

[2] JENKINS J, BRANNOCK E, HEINZ A, et al. JavaWIDE: innovation in an online IDE: tutorial presentation[J]. Journal of Computing Sciences in Colleges, 2010, 26(2): 248-250.

[3] WU L, LIANG G, KUI S, et al. CEclipse: An Online IDE for Programing in the Cloud[C]// Services. DBLP, 2011: 45-52.

[4] ZENG S Q, XU J B. The Improvement of PaaS Platform[C]// 2010 First International Conference on Networking and Distributed Computing (ICNDC). IEEE, 2010: 156-159.

[5] FINK J. Docker: a Software as a Service, Operating System-Level Virtualization Framework[J]. Code4lib Journal, 2014, 25: 3-5.

[6] SURKSUM K V. Microsoft announces support for Docker container virtualization for next version of Windows Server[J]. Red, 2017, 2016.

[7] KERNER S M. Amazon Embraces Docker Virtualization[J]. Eweek, 2014.

[8] Cloud9[EB/OL]. [2016-5-13]. https://c9. io.

[9] Codeny[EB/OL]. [2016-5-13]. https://codenvy. com.

[10] LEWIS J, FOWLER M. Microservices[EB/OL]. (2014-3-25)[2016-5-13]. http://martinfowler. com/articles/microservices. html.

[11] XU X, YU H Q. A Container-based Resource Management Game. Model for Cloud Computing [J]. Journal of East China University of Science and Technology (Natural Science Edition), 2015, 41(1): 89-96. (in Chinese)
徐昕，虞慧群. 基于容器的云资源管理博弈模型[J]. 华东理工大学学报(自然科学版)，2015，41(1)：89-96.