文章编号:1671-4598(2018)05-0252-05 DOI:10.16526/j.cnki.11—4762/tp.2018.05.063 中图分类号:TP31 文献标识码:A

基于容器的具有版权保护特性的协同开发架构

叶培根，毛建华
（上海大学 通信与信息工程学院，上海 201210）

摘要：文章探究了通过使用容器构建文件 Dockefile 将多种应用集成到同一容器的自动化协同开发方法，基于海云分形架构，海端各应用程序的开发者只关注于自身应用的容器化方法，无法获得其他应用的代码以及算法等关键涉密信息，在云端各方的 Dockerfile 自动化集成为同一文件并构建成为第三方可信云平台内的容器镜像；云端的自动化集成有效的避免了不同领域不同归属不同部门协作中知识产权内容的泄露问题；并且，通过 cadvisor 等流量监控工具，可以获得容器内实际通过流量，计算各应用处理流量比例，合理分配收益，从技术方面避免合作成果的利益分配不均问题；第三方可信云平台由政府、科研机构或第三方相关专业机构等具有社会公信力的实体机构进行运营，从而保证云端数据的安全；通过海端、云端间的协同，实现基于容器的具有版权保护特性的增量式协同开发架构。

关键词：容器；版权保护；协同开发

A container based collaborative development framework with copyright protection

Ye Peigen, Mao Jianhua
(School of communication and information engineering, Shanghai University, Shanghai 201210, China)

Abstract: This paper explores the method that variety of applications integrated into the same container automated collaborative development framework by using the container configuration file Dockerfile, Haiyun fractal architecture based on method of sea container ends of the application developers only pay attention to their own applications, unable to get the secret key code of the other application and algorithm of information in Dockerfile automation cloud parties the integration for the same file which becomes a trusted third party cloud platform container mirror inside. Automatic integration of cloud effectively avoids the leakage of intellectual property rights in different areas, different ownership and different departments. And through the cadvisor and other flow monitoring tools, we can get the actual flow through the container, calculate the proportion of each application processing flow, reasonable distribution of revenue, and avoid the uneven distribution of benefits of cooperation results from the technical aspects. The third party trusted cloud platform is operated by the government, scientific research institutions or third party related professional agencies and other entities with social credibility, so as to ensure the security of cloud data. Through the collaboration between the sea side and the cloud, a container based incremental collaborative development framework with copyright protection characteristics is implemented.

Keywords: container; copyright protection; collaborative development

0 引言

目前，各科研院所、企业、高校等机构掌握着大量的科研成果，其中部分成果以专利、论文的形式予以公开，但仍有较大部分的科研、产业成果存在于研发团队中。出于版权保护、利益等相关考虑，科研成果间的协同创新目前主要基于双方互相信任的基础上，协同门槛高，利益分配模糊。究其原因，不仅有政策、技术问题，也有体制问题。如何通过技术手段，降低协同创新门槛，让各方能够放心的提供己方成果，不受版权侵犯的困扰；在共享协同成果的同时，合理分配各自知识产权所创造的利益，使其版权、利益都能得到保护是解决问题的关键。随着容器技术的蓬勃发展，集装箱式的部署方式为软件应用的协同开发创新实践提供了新的思路。

随着云计算、移动计算、物联网、大数据和社交网络的发展，面向智能应用的生态环境有利于创新涌现，有利于实现全面透彻的感知、宽带泛在的互联和智能分析，但也对智能应用的开发和创新模式带来了新的挑战[1]。而端云间协同的网络架构能够集成海端数据和云端计算的优势，逐渐成为创新的技术驱动力。

云计算是随着网络传输带宽不断提高而发展出的一种新的计算形式，区别于现有的在终端计算的方式，云计算将计算压力承载在网络的另一端，即云端，完成计算任务后再将结果返回给终端。云计算模式具有大规模、虚拟化、高可用性、高可扩展性、按需服务，成本低廉等特点。目前各大企业和科研机构正不断推进着云模式与现有技术的结合，云计算技术也成为业界技术创新实践的活跃试验场。

收稿日期:2017-11-19； **修回日期**:2017-12-18。
基金项目:中科院战略先导项目(XDA06010800)。
作者简介:叶培根(1992-),男,河南省濮阳市人,硕士研究生,主要从事软工数据智能方向的研究。
毛建华(1990-),男,副教授,硕士研究生导师,主要从事地理信息系统、智慧城市、激光雷达遥感方向的研究。

容器是通过一种虚拟化技术来隔离运行在主机上不同进程，从而达到进程之间、进程和宿主操作系统相互隔离、互不影响的技术，可以提供轻量级的虚拟化，以便隔离进程和资源[2]。容器能够用来实现资源隔离，提高资源利用率。其轻量化的特性，使其能够实现虚拟机的功能下，各项开销都远低于虚拟机的资源利用。容器的运行效率基本接近于物理机，容器的操作可达到秒级甚至毫秒级，容器的一次构建，多次复用的灵活性极大满足了传统应用快速移云的需求。容器虚拟化技术的日益流行以及其诸多特性，逐渐成为业界和学术界研究和实践的热点。在实践方面，容器促进了微服务架构的发展，使得传统的大型软件更易于平缓的转换为高内聚、低耦合的微服务架构。同时，容器以及相关技术的日益成熟，促使了容器云平台的诞生，为 PaaS 层（platform as Service）的实践提供了一条新的技术途径。在科研方面，根据文献检索引擎 Web of Science 上的统计结果显示，图 1，容器以及虚拟化相关研究在近几年愈加活跃，以论文为代表的科研成果数量从 2014 年开始显著提升，是当前学术界日益关注的研究热点。容器与其他技术相比具有诸多优势，除了轻量化、弹性可伸缩的特性外，其镜像分层机制和自动化构建的特性为本文提出基于容器的具有版权保护特性的协同开发架构提供了技术基础。

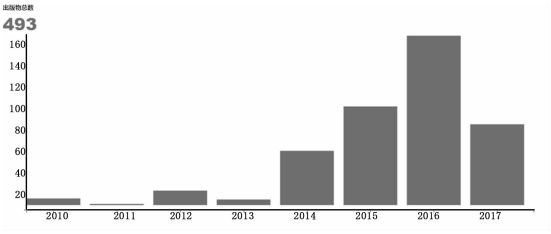

图 1　Web of science 中容器与虚拟化文献统计

1　国内外相关研究进展

容器本质上是通过虚拟化操作系统的方式来管理代码以及应用程序，容器技术的发展从分配存储到协调网络都对现有架构提供了新的思路，众多研究者都在试图用这样一种轻量化、一次构建多次复用的容器技术对现有框架进行创新研究。众多研究者关于容器架构的研究，对我们提出具有版权保护功能的增量式开发架构具有重要的借鉴意义。

Xinjie Guan 设计了一种面向应用程序的 docker 容器，基于 AODC 资源分配框架，最小化数据中心的应用部署成本，并能够根据云端应用的工作负载自动伸缩[3]。Xxx 综合考虑 docker 各种应用的特征，云数据中心的可用性对 AODC 进行建模，并提出了一种用于数据中心的可伸缩性算法。

Abdulrahman Azab 提出 socker 概念，一种用于封装运行在 Slurm 上的 docker 容器的技术。不同于其他 docker 支持的用于 HPC（High－performance computing）的容器平台，socker 使用底层的 Docker 引擎[4]。经过测试，socker 不仅安全，并且除了 Docker 引擎自身所带来的开销外并没有引入更多的资源消耗。

当本地存在 docker 镜像时，此节点启动 docker 容器速度很快。但是如果该节点没有镜像时，从注册中心下载镜像会增加启动时间。Senthil Nathan 提出了一种对容器镜像进行协作管理的系统 CoMICon[5]。CoMICon 的特点有：1、能够在一组节点中启用协作注册中心。2、能够存储、删除部分镜像层。3、有助于注册中心之间转移镜像。4、当启动一个新的容器时，能够从不同的注册中心下载镜像。经过测试，CoMICon 平均能够增加可用镜像数量 3 倍以上，减少应用配置时间 28%

由于容器和虚拟机不同的镜像格式和不同的镜像管理进程阻碍了混合分布式云架构的采用。German Molto 提出了一种基于开源工具和标准的工作流，能够在混合分布式基础设施上引入一致的应用程序，这些应用需要同时部署在 VM 和 Docker 容器中。利用和扩展 TOSCA 标准来描述应用程序需求，并采用 DevOps 实践，使得用于在不同平台上执行应用程序所需工件的一致性创建[6]。

为了能够在云端运行应用，我们需要确定一组能够与容器相适应的虚拟机，并且还要考虑到需求的多样性。为此，在考虑到容器需求和虚拟机资源的异质的情况下，Matteo Nardelli 提出了一种用于容器部署的虚拟机弹性配置的公式（EVCD）[7]。并且以论证多个 QoS 度量指标灵活性为目的评估了 EVCD 公式。

为了解决微服务体系结构中服务发现的问题，Joe Stubbs 提出了一种基于 Serf 节点的自服务开源解决方案。Serf 节点是由一个或多个 Docker 容器组成的镜像。新镜像部署在 Serf 节点集群内，并在集群内进行自我宣传，提供服务发现机制、监视和自我修复，最终生成没有主节点，同构且完整的图[8]。

复现已有科研成果的能力成为学术界日益重要的课题。在计算机科学技术中，由于公开的代码和数据中有许多没有正式说明的假设、依赖和配置，使得科研成果的复现性大大降低。Jürgen Cito 通过借助 Docker 容器克服了上述问题，提高了软件及 web 工程中科研成果的复现性[9]。

与基于管理程序的虚拟化方法相比，容器的使用提供了更快的启动时间，减少了计算机资源的消耗。但目前缺少一种在设计阶段针对容器进行可部署性验证的工具。Fawaz Paraiso 提出了一种对 Docker 容器进行建模的方法，可以保证容器的可部署性和更好的管理[10]。

随着越来越多的用户希望在动态、异构环境中部署容器，跨越多个云和数据中心的容器部署能力和有效管理变得至关重要。Moustafa Abdelbaky 提出了一个 C－Ports 原型框架，该框架利用约束规划模型选择资源，利用 CometCloud 分配和释放资源。该原型已被有效地用于部署和管理由五个云和两个集群组成的动态环境中的容器[11]。

2　架构研究

2.1　Docker 介绍

Docker 是一个开源项目，是目前非常流行的容器平台，能够使代码在虚拟化的容器中自动运行。开发者使用 docker 技术可以避免协作开发过程中应用只能运行在自己机器上的困境。Docker 将软件以特定格式打包，并将其运行在共享的操作系统中。容器技术不是虚拟机，Docker 在打包应用时并不打包操作系统，只打包所需依赖包和软件应用所需设置。这种机制保证了软件运行的高效，轻量级，自包含系统，不依赖底层平台等特性。Docker 能够运行并管理多个独立的运行在容

器内的应用已获得更好的计算密度。企业通过使用 Docker 建立的敏捷软件交付生产线，能够实现新版本更快更安全更兼容的交付客户。

2.2 整体架构设计

容器镜像分层机制能够使容器自动化的将一个或多个应用打包封装为镜像，一个镜像可包含一个或多个应用。多个镜像可以共用一个父镜像，镜像和父镜像呈现树状层叠关系。每层都是链条层里的一个镜像，各个层叠到一起才形成一个完整的镜像。一个层可以看成一个目录，包含一部分差分数据，镜像部署为容器时，根据层叠关系下载依赖镜像组合成一个容器的运行时和根文件系统。利用容器镜像分层机制，我们设想参与协同开发的各方应用能够以一个或多个镜像层的方式共同组成一个最终完整的大镜像。协同开发的各方首先基于共同的基础镜像以构建镜像文件的方式将自己的应用容器化，其次将创建镜像过程中撰写的镜像构建文件（本文使用的是 Dockerfile）和相关配置文件都上传至第三方的可信平台，由于镜像构建文件具有严格的书写规范，因此能够将各方自动化合并为大镜像的镜像构建文件，合并工作不需要人工参与，避免了合并过程中各方知识产权的泄露。需要注意的是，第三方可信平台需要具有加密功能，只能允许用户上传下载自己的相关文件，不能操作其他文件，避免泄露。此第三方可信平台具体实现细节不在本文中过多描述。通过容器镜像分层机制和自动化的构建功能，我们最终可以得到集成了各方应用的大镜像，其构建过程中，参与协同开发的各方均无机会接触到对方的相关文件，从技术上保证了协同过程中的版权保护，并且实现了基于对方应用快速增量式开发的目的。

2.2.1 Dockerfile 的合并规范

用户 A、B 基于同一公共基础镜像（本文使用 centos7.1511 版本），用户 A 按照合并格式提供 A 服务的 Dockerfile 镜像构建文件，用户 B 按照同样的格式提供 B 服务的 Dockerfile 镜像构建文件。由于 A、B 用户的 Dockerfile 具有同样格式的书写规范，两份 Dockerfile 可通过解析代码的方法将其重新组合为一份 Dockerfile 文件。合并各式如图 2。

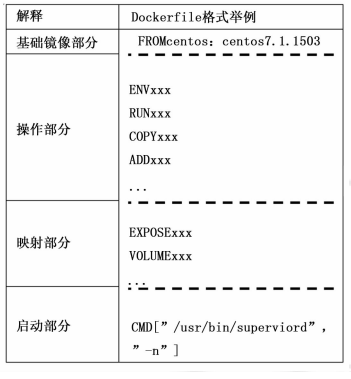

图 2 Dockerfile 合并各式

1）基础镜像部分：

一个有效的 Dockerfile 必须以 FROM 作为第一条命令，所用公共基础镜像需是经过验证的健壮镜像，一般可以很方便的从 DockerHub 拉取，也可使用自定义的基础镜像。多用户进行协同开发时需使用同一基础镜像，多用户之间使用共同的基础镜像是不同应用间进行合并集成的前提。

2）操作部分：

每个用户将应用编写为 Dockerfile 的过程中，都要以公共基础镜像为开始进行操作，常见的操作有：下载安装应用所需依赖安装包、拷贝相关文件进入容器、安装过程运行命令、设置环境变量等。每一部分的操作都要尽可能的不对其他用户产生影响，例如：在设置环境变量进行操作完成后，要将环境变量的值设置为原来的初始值，使得其他用户在编写 Dockerfile 时所获得的环境变量即为初始值，避免混乱。操作要尽量减少该应用部分的体积，例如：下载安装完成相关依赖包后要立即删除安装包，否则合并后的镜像构建为容器时会使体积过于庞大，不利于容器快速重建，也会影响镜像从镜像库拉取至本地的时间。

3）映射部分：

此部分主要是将各应用的端口暴露以供外部访问、数据卷映射至宿主机进行存储。需要注意的是各应用间需要映射的端口、目录需事先双方商定，避免使用常用端口发生冲突的情况。容器内的相互调用不需要将端口对外映射。

4）启动部分：

此部分命令不需用户撰写，由自动合并程序在 Dockerfile 文件最后添加即可。命令相对较为固定，即容器启动后自动运行 supervisor 进程控制系统，随后由 supervisor 启动容器内各应用。

2.2.2 合并环境下的文件构成

表 1 合并环境下的文件构成

Dockerfile	读取指令自动构建镜像
进程控制程序配置文件	用于启动容器内各应用
其他	各应用相关文件

在云端自动化集成各方应用的环境下存在有以下三种文件（表 1）：1）用于读取各协同方指令并自动化构建镜像的 Dockerfile 文件。2）进程控制程序配置文件；由于各方应用存在着进程启动上的先后顺序，某些应用依赖于其他应用所提供的数据、运行环境等，在此文件内设置各应用的启动、就绪、阻塞、销毁以及相关持续时间等。3）各方应用所需要用到的相关文件。

2.2.3 协同开发过程概述

在确定完成需求和集成的目标后，参与协同开发的用户互相商定各自的应用所需要暴露的端口，以及数据卷映射在宿主机的目录，保证后续的集成不会造成端口、目录的冲突。随后，按照合并规范，通过自动化的合并程序生成集合了多应用的镜像构建文件，产生容器镜像。将镜像文件运行可得到运行态的 Docker 容器，其中包括了进程控制程序、各协作用户应用程序。容器在启动时首先启动进程控制程序，随后各应用程序的启动交由进程控制系统接管。这里需要注意的是，由于各应用的启动具有先后性（例如：应首先启动数据库应用，再启动上层应用），各应用的启动脚本应注意通过休眠等方式实现按序启动。当容器启动后可通过 Cadvisor 工具监控容器流量以及性能，实时掌握容器状态，计算流量，合理分配容器内各

应用收益。当确认合并后的容器能够正常运行后，可将镜像上传至私有镜像库，实现一次构建，分布式多次复用。协同增量式开发流程如图 3。

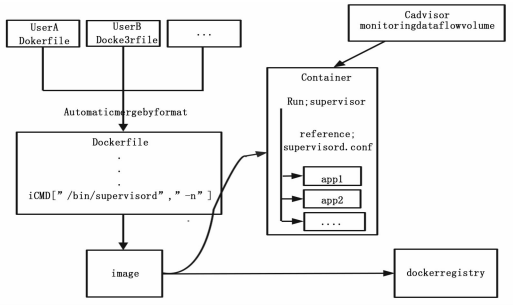

图 3　协同增量式开发架构图

3　增量式协同开发 Demo 系统设计与实现

3.1　系统组成

本 Demo 系统共有三部分构成：1）Web 应用服务程序 Tomcat，2）门户网站应用程序 app. war，3）存在于 mysql 数据库中的第三方数据。

3.2　协同开发场景

用户 A 持有 web 应用服务程序 Tomcat，用户 B 持有第三方数据，可以提供使用但需要客户按使用量付费，并且不希望客户一次付费之后永久获得所有数据。用户 C 持有门户网站应用程序，在有数据的情况下可直接运行起来即可对外提供服务。用户 A、B、C 各自拥有一部分功能，本实验设计目的是将隶属三个不同用户的功能型应用快速协同开发成为一个新的应用，并且协作过程中以容器的方法保证各方技术、数据等核心知识不会泄露，在完成增量式开发的过程中提供版权保护功能。

3.3　相关配置商定

用户 A、B、C 经过商定后可确定 tomcat 应用程序所在目录，mysql 数据库和数据存放目录，门户网站应用程序所在目录，tomcat 映射端口，mysql 数据库映射端口。

3.4　Demo 系统协同开发流程

用户 A、B、C 将相关配置、文件、数据上传至第三方可信平台，分别按照图? Dockerfile 合并各式撰写各自应用的 DockerFile，经由自动合并程序合并成为集成了三方应用的 Dockerfile，本次 Demo 系统应用集成后的 Dockerfile 如图 4 所示。

```
FROM        centos:centos7.1.1503                                    ①
MAINTAINER Payton Ye

# Install Mariadb
ENV DATA_DIR /var/lib/mysql
RUN yum install -y mariadb mariadb-server && \
    yum install -y java-1.7.0-openjdk.x86_64 && \                    ②
    yum clean all
ADD mysqld_charset.cnf /etc/my.cnf.d/
COPY scripts /scripts
RUN chmod +x /scripts/start
RUN chmod +x /scripts/firstrun_maria

# Install tomcat
COPY apache-tomcat-7.0.55 /usr/lib/tools/                            ③
COPY tomcat.sh /usr/lib/tools/tomcat.sh
RUN chmod +x /usr/lib/tools/tomcat.sh

# Install app.war                                                    ④
COPY app.war /usr/lib/tools/apache-tomcat-7.0.55/webapps/

# Install supervisor
RUN yum install -y python-setuptools && \
    easy_install supervisor && \                                     ⑤
    yum clean all
COPY supervisord.conf /etc/supervisord.conf
RUN mkdir /var/log/supervisor

EXPOSE 3306 8081                                                     ⑥
VOLUME ["/var/lib/mysql"]

CMD ["/usr/bin/supervisord", "-n"]                                   ⑦
```

图 4　Demo 系统 Dockerfile

图 4 中共有 7 部分组成，各部分解释如下。

1）基础镜像部分。

centos：centos7. 1. 1503 为此次集成三方应用所基于的基础镜像，MAINTAINER 代表此 DockerFile 的维护者信息。

2）用户 A 持有的 Mysql 数据对应 Dockerfile 的代码信息。

其过程依次为设置环境变量，基于公共原始镜像安装 mariadb 数据库，拷贝相关配置文件以及数据至镜像内，为相关脚本添加执行权限。

3）用户 B 持有的 Tomcat 应用服务程序对应的 Dockerfile 的代码信息。

其过程依次为：拷贝相关应用文件、配置文件至镜像，赋予应用启动脚本执行权限。

4）用户 C 持有的门户网站应用对应的 Dockerfile 的代码信息。

其内容为将 Java 应用的 War 包拷贝至镜像中。

5）安装 supervisor 进程管理程序。

此步骤可固化在基础镜像中默认使用 supervisor 作为进程管理工具，若要选择其他进程管理程序控制各有应用的启动，则替换此部分的代码。

6）映射部分。

对外暴露数据库接口（可不暴露，则只能容器内应用访问数据），对外暴露映射目录。

7）启动部分。

不需用户撰写，由自动合并程序在 Dockerfile 文件最后添加。命令相对较为固定，容器启动后自动运行 supervisor 进程控制系统，随后由 supervisor 启动容器内各应用。

使用合并后的 Dockerfile 执行构建镜像的命令，产生容器镜像，容器内由 supervisor 启动三方应用，各应用之间互相调用实现协同。同时启动 Cadvisor 容器，能够监视应用的资源情况、应用的流量情况。具体流程如图 5 所示。

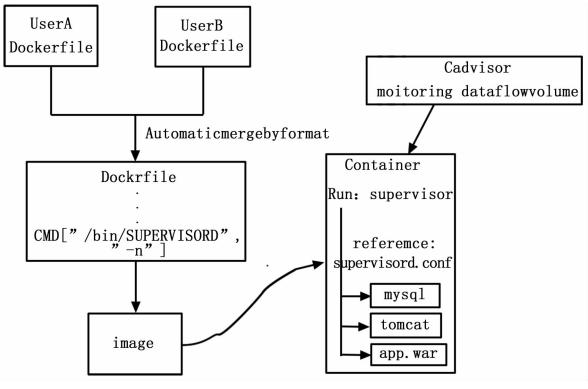

图 5　Demo 系统增量式协同开发流程图

实际构建出来的容器启动后，访问所在服务器地址：8081，即可访问门户网站。服务器性能越高，从启动到访问所等待时间越短。

4　结论

本文提出一种基于海云分型架构，并具有版权保护特性的自动化协同开发方法。在海端开发者遵照 Dockefile 格式将应用容器化，在云端，各方 Dockerfile 自动化集成为同一文件并

构建成为第三方可信云平台内的容器。海端各应用程序的开发者只关注于自身应用的容器化方法，无法获得云端其他应用的代码以及算法等关键涉密信息，从而在根本上解决了不同领域、不同部门的海端用户进行协作过程中知识产权成果的安全问题。而第三方可信云平台由政府、科研机构或第三方相关专业机构等具有社会公信力的实体机构进行运营，从而保证云端数据的安全。通过海端、云端间的协同，实现基于容器的具有版权保护特性的协同开发架构。并且，通过 cadvisor 等流量监控工具，可以获得容器内实际通过流量，调用次数，使用时间等参数，计算各应用处理流量比例，实现多种计费方式，保证各方知识成果得到合理利益分配。

未来，海云分型架构可根据用户需求演化为多种协同方式，如：1）云端海化：第三方可信云平台可集成海端容器化工作场景为 SaaS 服务供开发者使用，使容器化本地应用的工作在云端完成，本地不再需要容器开发环境，降低海端开发成本，同时避免了本地知识泄露以及数据丢失等不安全情况的发生。形成的知识成果可直接以资源的方式供他人调用，获得收益。2）海端云化：针对机构内部协作率高，对外协作需认可的需求场景，可部署私有可信云平台，供内部知识成果的交流创新，满足内部频繁的协作开发，所有数据、流量完全在私网中通信，海端云化后的工作场景能够进一步提高协作中的速度以及安全性。在与外部单位进行协作中，本地云平台以海端的身份与第三方可信云平台进行通信，经过本地认证机构为相关技术成果赋予对外访问权限，实现海云协同开发。

参考文献：

[1] 宁德军，封松林，萧海东，等. 下一代智能应用开发模式研究 [J]. 网络新媒体技术，2016，5 (1)：1－10.

[2] 汪 恺，张功萱，周秀敏. 基于容器虚拟化技术研究 [J]. 计算机技术与发展，2015 (8)：138－141.

[3] Guan X，Wan X，Choi B Y，et al. Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers [J]. IEEE Communications Letters，2017，21 (3)：504－507.

[4] Azab A. Enabling Docker Containers for High－Performance and Many－Task Computing [A]. IEEE International Conference on Cloud Engineering [C]. IEEE，2017：279－285.

[5] Nathan S，Ghosh R，Mukherjee T，et al. CoMICon：A Co－Operative Management System for Docker Container Images [A]. IEEE International Conference on Cloud Engineering [C]. IEEE，2017.

[6] Moltó G，Caballer M，Pérez A，et al. Coherent Application Delivery on Hybrid Distributed Computing Infrastructures of Virtual Machines and Docker Containers [A]. Euromicro International Conference on Parallel，Distributed and Network－Based Processing [C]. IEEE，2017：486－490.

[7] Nardelli M. Elastic Allocation of Docker Containers in Cloud Environments [C]. Zeus Workshop. 2017.

[8] Stubbs J，Moreira W，Dooley R. Distributed Systems of Microservices Using Docker and Serfnode [A]. International Workshop on Science Gateways [C]. IEEE，2015：34－39.

[9] Cito J，Gall H C. Using docker containers to improve reproducibility in software engineering research [M]. Web Engineering. Springer International Publishing，2016.

[10] Paraiso F，Challita S，Al－Dhuraibi Y，et al. Model－Driven Management of Docker Containers [A]. IEEE，International Conference on Cloud Computing [C]. IEEE，2017：718－725.

[11] Abdelbaky M，Diazmontes J，Parashar M，et al. Docker Containers across Multiple Clouds and Data Centers [A]. Ieee/acm，International Conference on Utility and Cloud Computing [C]. IEEE，2016.

(上接第 251 页)

本装置选取 STC89C52 单片机作为主芯片，设计信号采集电路、放大电路、滤波电路、整形电路、显示电路、报警电路。通过红外光电传感器采集驾驶员脉搏频率特征信号，可显示当前脉搏跳动速度，体现该装置的即时性；通过所测数据与设定值的对比来判定驾驶员是否处于睡意状态，并对处于睡意状态下的驾驶员进行预警，同时针对个体差异性，可自由调节预警数值范围，体现了该装置的多效性与兼容性。调试结果显示本装置识别准确率高，数值可靠，能够有效的检测驾驶员的睡意状态并实现预警功能；成本低廉，操作简单，可用于车载，为检测驾驶员睡意技术的研究提出了新的思路，具有广泛的市场应用前景。

参考文献：

[1] 李忠东．疲劳驾驶成为“隐形杀手”[J]．交通与运输，2012 (2)：29－30.

[2] Mao Z，Yan X P，Wu C Z. Driving Fatigue Identification Method Based on Physiological Signals [C]．International Conference of Chinese Transportation Professionals Congress. 2008：341－352.

[3] Sommer D，Golz M，Trutschel U，et al. Assessing Driver’s Hypovigilance from Biosignals [M]．4th European Conference of the International Federation for Medical and Biological Engineering. 2008：152－155.

[4] Jiao K，Li Z，Chen M，et al. Effect of different vibration frequencies on heart rate variability and driving fatigue in healthy drivers [J]．International Archives of Occupational and Environmental Health，2004，77 (3)：205－212.

[5] Anumas S，Kim S C. Driver fatigue monitoring system using video face images & physiological information [C]．Biomedical Engineering International Conference. IEEE，2011：125－130.

[6] Zhang Z，Zhang J S. Driver Fatigue Detection Based Intelligent Vehicle Control [C]．International Conference on Pattern Recognition. IEEE，2006：1262－1265.

[7] 张爱华，杨 华，孔令杰，基于生理信息的视频显示终端精神疲劳状态研究 [J]．生物医学工程，2011，28 (5)：1025－1029.

[8] 陈雪峰，周宽久．基于脉搏波分析的驾驶员疲劳判定研究 [A]．科学仪器前沿技术及应用学术研讨会 [C]．2006.

[9] 韩海川．基于人体生理信号的疲劳驾驶识别方法的研究 [D]．天津：天津职业技术师范大学，2016.

[10] 顾征远．面向驾驶员的疲劳监测系统研究 [D]．杭州：浙江大学，2013.

[11] 金伟正．单线数字温度传感器的原理与应用 [J]．电子技术应用，2000 (6)：42－43.