

# 基于 Kubernetes 的列控系统测试容器云平台设计

马勤政,徐中伟,梅 萌  
(同济大学 电子与信息工程学院,上海 201804)

**摘 要:**为降低列控系统测试软件的研发运维成本,提升列控系统测试平台的测试效率与自动化水平,提出了基于 Kubernetes 集群技术的容器云平台的功能与架构设计,同时对容器云平台上的研发与测试流程进行了说明与规范。根据列控系统测试平台的功能特征,应用微服务的思想,将平台拆分为若干逻辑子集群,子集群之间通过 Kubernetes 集群内部的网络服务进行通信,平台的各类功能在其对应的逻辑子集群中以容器微服务的形式各自独立执行。同时,在云平台中引入自动化测试脚本与测试案例自动生成的技术,以提高自动化测试的覆盖范围。这一平台设计能够降低列控系统测试平台各个功能之间的耦合程度,提升平台的可靠性与可扩展性,在满足现有的列控系统信号设备测试需求的基础上,为大量测试任务以及未来各类新型列控信号设备的测试任务提供支持。

**关键词:**列控系统测试;云计算;Kubernetes;微服务;容器云平台;自动化测试

中图分类号:TP399

文献标识码:A

文章编号:1673-629X(2021)06-0052-07

doi:10.3969/j.issn.1673-629X.2021.06.010

## Design of Container Cloud Platform for Test of Train Control System Based on Kubernetes

MA Qin-zheng, XU Zhong-wei, MEI Meng

(School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China)

**Abstract:** In order to decrease the cost of the development and operation of the software for train control system test and improve the automation level of the test platform of train control system, an architecture of container cloud platform based on the Kubernetes cluster technology is proposed. At the same time, the procedure of development and test on the container cloud platform is also defined. Applying the idea of micro service, the platform is divided into several logical sub clusters, connected by inner network service provided by Kubernetes. At the same time, the automatic generation of automated test scripts and test cases is introduced into the cloud platform to improve the coverage of automated testing. This platform design can reduce the degree of coupling between the various functions of the train control system test platform, improve the reliability and scalability of the platform, and provide support for a large number of test tasks and the test tasks of various new train control signal equipment in the future on the basis of meeting the existing test requirements of the train control system signal equipment.

**Key words:** test of train control system; cloud computing; Kubernetes; microservice; container cloud platform; test automation

### 0 引 言

随着国家铁路与轨道交通领域的蓬勃发展,高速铁路与城市轨道交通中基于通信的列控系统技术不断推陈出新,从而使得相应的测试需求也不断增加。在传统的列控系统测试平台的软件开发过程中,针对每一个不同的测试任务,均需要开发不同的信号设备仿真对象,对特定的业务逻辑与安全通信功能进行封装,再令各仿真对象独立运行,组成测试所需的仿真环境。仿真对象软件的各类功能均集成在一个应用程序之

中,使得各个功能组件之间具有较高的耦合性,难以扩展与更新,使得列控系统测试软件的开发与运维的时间与维护成本难以降低,也难以对测试平台中软件的故障问题进行快速定位与修复。

通过建立容器云平台,将列控系统测试软件依照功能拆分为若干服务组件,各个服务之间仅通过通信接口进行耦合,在集群中的一个或多个计算节点上作为进程独立运行,组件之间通过网络协议进行通信,通过各类型服务之间的组合对列控系统测试软件进行开

收稿日期:2020-08-09

修回日期:2020-12-11

基金项目:国家自然科学基金委高铁联合基金(U1734211)

作者简介:马勤政(1995-),男,硕士,研究方向为基于通信的列车控制系统;徐中伟,博士,教授,研究方向为基于通信的列车控制系统、安全苛求系统,可信计算、软件构件化开发及应用等。

发,能够大幅降低列控系统测试软件的内部耦合度,提升开发与维护效率。相较于虚拟机,容器技术作为一种轻量级的虚拟化手段,将服务与其所需的运行环境打包为镜像文件,便于部署与修改,具有更高的灵活性,同时对 I/O、CPU 与内存等资源的需求均较低。近年来国内外对于容器及其编排技术均有较多研究成果。为提升研发效率、降低运维与操作成本,该文提出了基于 Kubernetes 容器编排系统的列控系统测试容器云平台的设计。

## 1 Docker 与 Kubernetes 技术概述

Docker 是一种提供轻量级虚拟化解决方案的开源的引擎,建立在 Linux 操作系统的 Cgroup、Namespace 以及 rootfs 等容器技术之上。Docker 能够将服务应用程序及其所需的文件系统按层级打包为一个容器镜像,容器镜像不依赖操作系统类型,仅要求操作系统具有相同的内核,这使得 Docker 镜像能够在任何支持 Docker 的设备中进行迁移。同时,Docker 镜像相对于其他的虚拟化方案具有更快的创建、启动与停止速度,同时占用更少的操作系统资源,从而降低了应用服务的开发、部署与迁移成本<sup>[1]</sup>。

Kubernetes 是一种用于在多机器云环境,即容器云集群中自动管理、部署与扩展容器应用的开源的容器编排系统。Kubernetes 运行在 Docker 容器之上,由 Google 公司的 Borg 系统发展而来<sup>[2]</sup>,能够按照用户设定的规则自动化地对集群进行管理,为集群的运行提供了稳定性与效率的保障。

## 2 平台设计

### 2.1 系统架构设计

列控系统测试平台使用计算机软件与硬件资源对列控系统中的一个或多个通信实体进行模拟,各个模拟对象组成了完整的测试环境。测试环境能够与被测对象进行信息交互,测试人员通过在测试环境中下达命令以模拟测试大纲要求的运行环境,并观察被测对象的行为是否与预期一致。各个仿真对象在业务逻辑上具有较大差异,但在本质上均能够被抽象为应用层、安全层与通信层三个逻辑模块的组合。其中应用层负责依照测试大纲生成需要发送至被测对象的原始消息或解析接收到的被测对象发送的消息,安全层负责在待发送的原始消息中加入安全相关的数据或验证收到消息的安全性,而通信层负责测试平台与被测对象之间消息包的发送与接收。

铁路安全通信系统的结构抽象如图 1 所示<sup>[3]</sup>。

在铁路通信信号领域,两个通信实体之间的消息传输都遵循一定的层级式的通信协议。原始消息通过

逐层封装,形成最终的消息报文。以 RSSP-I 协议与 RSSP-II 协议为例,两类通信协议中的每一层都是对某种功能需求的抽象,对下层的输出进行进一步封装,供上层使用,且仅对相邻的上下层负责。依照这种低耦合的关系,能够将列控系统测试容器云平台的计算机集群划分为应用服务子集群、安全服务子集群以及通信服务子集群三个逻辑子集群。基于上述设计,在列控系统测试软件的开发过程中,可以将待开发的仿真对象按功能拆分为不同的应用层、通信层以及安全层服务,将每个服务打包为若干 Docker 容器镜像,并部署在同一个 Kubernetes Pod 中,同时相同类的 Kubernetes Pod 均应部署在列控系统测试容器云平台上的对应类型的逻辑子集群之中。

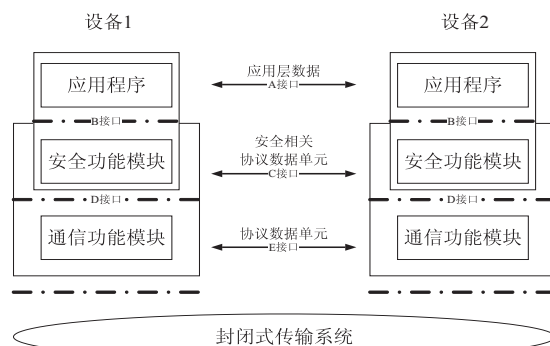


图1 铁路安全通信系统的结构抽象

在不同的列控系统测试任务中,应用层服务往往会依据任务的特性进行重新设计,安全层服务则依赖于少量稳定的协议,具有较高的可复用能力。每一个应用层通常会需要一个或多个通信层服务的支持。通信层服务一般可进一步分为软件协议栈与硬件设备两部分。这三类服务能够通过 Kubernetes 集群提供的内部的网络通信接口相互进行信息交互,从而实现对测试环境所需的通信实体的仿真。在测试任务执行过程中,应用层服务与通信层服务均需要同列控系统测试容器云平台外部进行通信。依据被测对象类型的不同,通信层服务与被测对象进行信息交互需要依赖不同的硬件通信设备,因此通信层逻辑子集群需要将通信层服务调度到具备特定网络接口资源的计算机节点上,并且允许服务实例直接使用宿主计算机节点上的网络接口。为了向测试平台下发命令或观测列控系统测试的过程与结果,测试人员需要在集群以外编写可视化界面程序,通过 Kubernetes 提供的外部访问机制访问应用层服务。

在列控系统测试过程中,除业务逻辑错误导致的通信信号数据包错误外,还可能发生潜在的各类故障,从而导致平台运行异常。为应对此类故障,列控系统测试云平台还需具有监控服务子集群,负责实时监控平台的运行状态,通过访问 Kubernetes 的控制平面以

及集群中每个工作节点上的 Kubelet 代理程序提供的应用程序接口 (API), 实时获取每个工作节点上的容器以及计算、储存与网络资源的健康状况, 对监控信息

记录进行保存, 通过对外提供 API 的方式为平台运维人员提供服务。列控系统测试容器云平台的架构如图 2 所示。

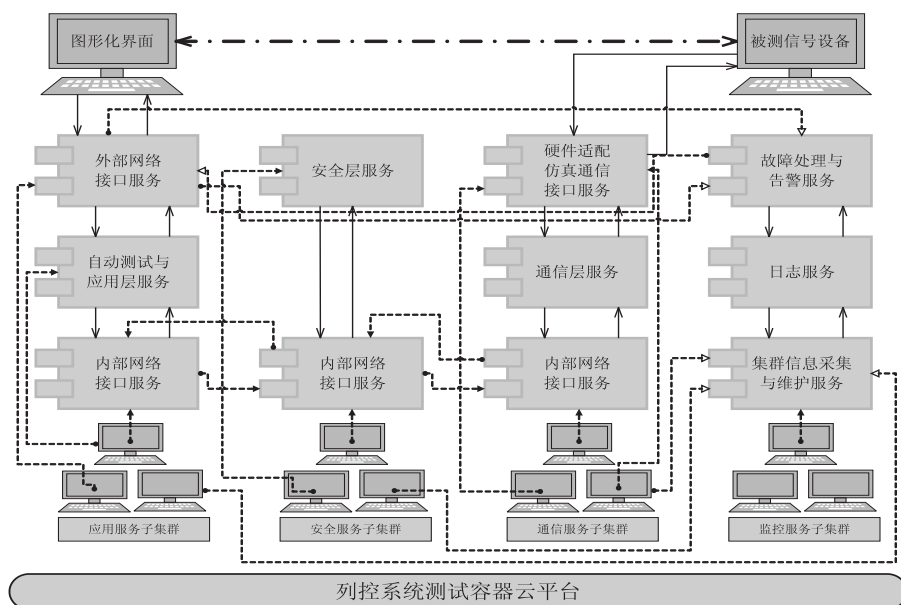


图 2 列控系统测试容器云平台的架构

## 2.2 平台功能设计

作为列控系统功能测试的基础设施, 列控系统测试容器云平台一方面需要能够简化列控系统测试软件的开发与部署流程, 提高研发效率, 另一方面应能够提升测试流程的自动化程度与稳定性。同时, 列控系统是一类安全苛求系统, 为了满足平台中的仿真对象与真实信号设备的一致性, 平台需要保证各类服务在部署和运行过程中的可靠性与稳定性。基于上述业务需求, 对列控系统测试云平台的功能做出如下设计。

### (a) 服务的部署与发现。

列控系统测试容器云平台应对 Kubernetes 提供的 API 进行再次封装, 为列控系统测试软件的开发者提供一套简明完备的命令行界面 (CLI), 通过 CLI 命令中的参数支持各层级的应用在其中的部署。开发者能够通过列控系统测试容器云平台的 CLI 将应用程序的容器镜像部署在集群当中, 或是对已经部署在集群上的服务进行下线操作。依据抽象分离的原则, 云平台应成为一个黑盒 (black box), 能够在平台内部自行完成应用的部署或下线, 无需开发者了解部署与下线过程的细节, 从而降低开发者的成本。

在服务的部署过程中, 对于无需与外部应用或设备进行通信的应用 (通常为安全层服务), 列控系统测试容器云平台能够通过 Kubernetes 的 DNS 发现服务使之能够与其集群内部的上下层服务进行通信<sup>[4]</sup>; 对于需要与外部设备或应用通信的应用 (应用层、通信层或其他负责监控与维护稳定性的服务), 云平台允许开发者在 CLI 中指定服务所需要使用的网络地址, 使

用 Kubernetes 提供的 Service 或 Ingress 机制将服务暴露在所指定的网络地址之上<sup>[5]</sup>。

在对铁路信号设备的测试中, 通信过程往往需要建立在不同的物理信道上, 如运行在以太网上的 TCP/UDP 通信、串口通信、控制器局域网 (CAN) 通信或使用特殊的通信网卡来模拟特定的网络环境, 如 GSM-R、LTE-R 等<sup>[6]</sup>。不同的物理信道要求需要列控系统测试容器云平台中具有具备各类通信硬件条件的计算机节点, 因此 Kubernetes 集群在建立时需要为具备不同硬件设备的节点做出特定的标识 (label)。在通过 CLI 部署通信层服务时, 需要指定所需要部署在其上的节点应具有硬件资源种类, Kubernetes 的节点选择控制器便能够按照开发者要求将通信层服务调度到最佳的符合要求的节点上。

### (b) 测试自动化。

在列控系统的测试任务中, 通常需要通过多种服务的若干实例进行组合实现对多种仿真对象的模拟, 从而能够依照测试大纲中的测试用例对被测对象进行测试。以 CTCS-3 级的 RBC 测试为例 (见图 3), 为了对 RBC 对象进行测试, 需要实现 CBI、TSRS、相邻 RBC 以及车载 ATP 设备的仿真对象<sup>[7]</sup>。这四类对象的应用层各不相同, 安全层均依赖 RSSP\_II 通信协议, 而在通信功能层面又存在通信协议与通信物理接口的差异。

在列控系统测试容器云平台中, 通过对各层级服务的组合, 能够快速实现仿真对象的实现, 提升了代码的利用率, 从而提升了测试工作的效率。

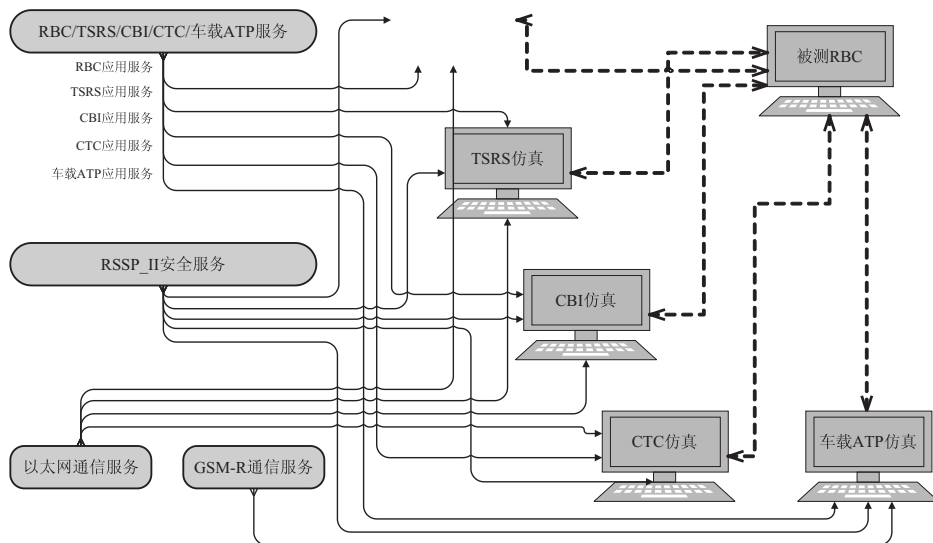


图3 CTCS-3 级列控系统 RBC 测试平台架构

为了对真实运行环境中可能出现的变化进行仿真,需要不同的仿真对象进行有序的协同操作。测试用例是对多种仿真对象有序行为以及被测对象的预期行为的描述。每个测试用例的流程均可表达为一个有向无环图(DAG),这个有向无环图是全连通的,具有树的性质,从根节点出发能够访问到图的全部节点。描述各个测试用例的树能够构成一个森林,具有时序关系的树之间具有父子关系。测试任务从树林的根处开始,并行执行不具备时序关系的测试用例,在执行每个测试用例时,对描述任务的图进行深度优先搜索,当图的每条边均被访问时,测试用例即执行完毕。全部的测试用例执行完毕后,测试过程即完成执行。在传统的测试过程中,由于各仿真对象均为集中式的独立应用程序,难以通过计算机指令协同调度各应用程序。因此测试人员需要依照测试用例的指示,手动对仿真

对象下达相应的命令。这一过程繁琐且容易出错。为了提高测试的自动化程度,列控系统测试容器云平台需要提供一种用于描述测试案例中有向无环图的领域专用语言(DSL)<sup>[8]</sup>。这一领域专用语言能够将测试案例描述为计算机可识别的过程,同时能够对各个测试案例进行优化,依照测试的相关需求对它们的执行顺序进行调整,使得整个测试的速度与效率得到提高<sup>[9]</sup>。通过计算机将从测试大纲转写而来的 DSL 脚本与各仿真对象的应用层服务分别作为容器部署在同一个 Kubernetes Pod 中,共享相同的 Linux Namespace,脚本与各服务即可在云端协同运作<sup>[10]</sup>。执行该脚本,脚本即可依照测试大纲的要求通过进程间通信(IPC)向各应用层服务顺序下达命令并接收反馈,从而实现对各仿真对象的协调。

容器云平台的自动化测试流程如图4所示。

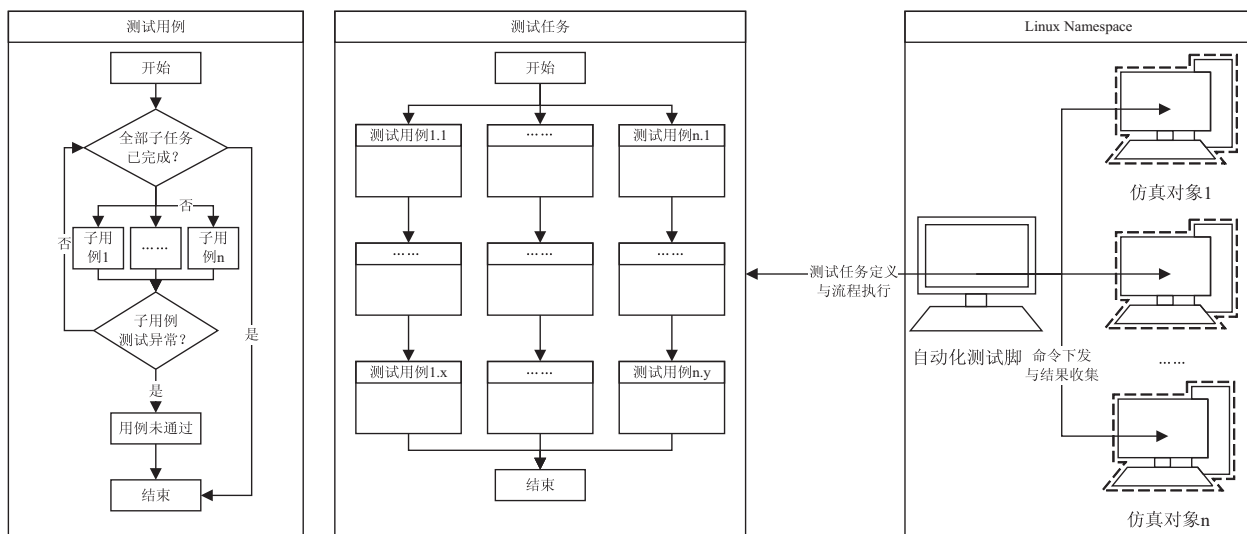


图4 容器云平台的自动化测试流程

随着列控系统测试场景的范围不断扩大,在一些测试任务中,需要通过较多的测试案例来对列控系统

的真实运行场景进行仿真,在一些测试场景中还需要对一些测试数据进行改变以充分模拟可能的运行状

况,仅通过真实的测试数据与测试场景进行测试已不能满足测试的需要。因此,列控系统测试容器云平台中还需要具备通用测试案例与测试数据的自动生成的功能,能够通过对测试场景的建模,使用启发式或半启发式的算法根据数据的不同特征生成所需的测试案例<sup>[11]</sup>。同时,近年来机器学习技术在自动化测试领域中的应用与研究不断深入,能够为测试提供更充分的、覆盖范围更加完备的数据<sup>[12]</sup>。列控系统测试容器云平台也应将人工智能技术引入列控系统的测试过程中,进一步提升列控系统测试的效率与自动化程度。

#### (c) 服务的可靠性保障。

在一个完整的测试任务中需要测试平台中的各个服务之间协同配合。在列控系统测试过程中,通信层为仿真对象提供与外部被测对象交互信息的接口,通常是无状态的。安全层服务按照其中安全协议中是否规定了上下文关系,可以是有状态或无状态的。而应用层服务通常用来执行仿真对象的业务逻辑,是有状态服务。通信层服务作为整个测试平台最基础的服务,同一类服务在同时可能需要支持多个测试任务同时进行。在多个任务同时执行的场景下,为了满足测试任务的实时性以及可靠性要求,就需要对热点通信层服务进行动态扩容,在集群中部署多个服务实例。而在少量任务同时执行的场景下,过多的通信层服务实例会占用集群较多的计算资源,此时需要平台自动对服务进行缩容。借助通信层服务的无状态特性与 Kubernetes 提供的弹性伸缩机制<sup>[13]</sup>,平台运维人员既能够在发现通信层服务缩扩容需求时,手动调整服务实例个数,也能够是在集群层面指定自动动态伸缩策略,委托 Kubernetes 的控制平面代为执行。

为了维护列车信号系统的可靠性,现行的绝大多数铁路信号协议均应用了冗余技术,在安全层或通信层上使用多机冗余,同时维护一个主系通信信道与多个备系通信信道的正常运行,一旦发生故障,可以立即进行主备系切换,保证铁路信号设备之间的通信稳定。而在列控系统测试容器云平台中,应用层服务与部分安全层服务作为有状态服务,一旦出现故障,难以仅通过替换服务的方式进行恢复。同时,云平台中还可能存在着分布式入侵、系统配置错误与网络故障等可能性<sup>[14]</sup>,因此,列控系统测试容器云平台需要具备一定的故障恢复能力。列控系统测试容器云平台能够通过 Kubernetes 的持久化储存机制,将服务的状态实时保存在节点当中,当某一服务发生故障需要进行恢复时,平台能够自动部署新的服务实例,这一实例能够读取之前实例故障前的状态,并将其作为依据对自身状态进行设置,从而使得云平台中的应用实例有了一定的冗余备份能力。进一步提升测试平台的可靠性。

#### (d) 平台故障监控与告警。

当前国内外已经提出了多种成熟的云平台监控方案<sup>[15]</sup>,通过与 Kubernetes 的 API Server 交互获取平台中各节点的性能指标,并通过各节点上的 Kubelet 提供的监控接口获取节点上的容器监控信息。监控平台获取到的信息均会被储存在数据库中,平台提供一个对外的图形化界面,用户可通过图形化界面操作监控平台,生成并查看所需的监控报告。

对于列控系统测试容器云平台,除计算资源与储存资源容量过低可能发生的故障外,还包括通信层服务所依赖的通信硬件资源的故障、安全层或应用层服务运行过程中未预期到的潜在故障等。故障监控平台在监测到常规故障发生的可能性(如计算、储存资源即将耗尽或某个服务的容器长期异常无法自动恢复等)后,会在平台提供的可视化界面上进行告警,等待平台运维人员检查处理。当故障发生后,监控平台会暂停故障涉及到的测试任务的运行,并且通过告警机制告知相关运维人员故障的相关信息,包括故障点的位置、故障现象以及可能的解决方案等。运维人员在结合监控平台的日志信息与现场情况对故障进行排查与修复后,能够根据故障的严重程度命令尚未完成的受影响的相关任务继续或重新进行。

### 3 容器云平台下的研发与作业流程

#### 3.1 研发与部署流程

通过对列控系统测试软件的各部分进行解耦合,将列控系统测试的各个功能拆分为不同层次的服务后,对逻辑复杂的安全层与涉及硬件层面的通信层服务的开发可以由较为专业的开发者完成,这些服务一旦开发部署完成,就能够保持稳定,为应用层服务提供支持。开发者在绝大多数只需要进行不同测试任务中所需的仿真对象(应用层与可视化界面)的开发,从而简化了列控系统测试软件的研发与部署流程。

在进行通信层服务的开发时,开发者首先需要查询集群中是否具有具备服务所需的硬件条件,且相关的计算、存储与网络资源满足条件的工作节点。若当前集群中尚不具备上述节点,那么需要向集群中添加新的满足条件的工作节点,或者在现有的工作节点机器上加装所需的硬件。服务应用开发完成后,开发者需将服务打包为 Docker 镜像,并提供 Kubernetes 部署服务所需的模板文件。模板文件中应该允许服务使用宿主工作节点的网络资源,同时允许对服务与其上层服务的内部通信接口以及服务与被测对象之间的外部通信接口进行配置。

在安全层服务的开发过程中需要针对不同的铁路信号安全协议进行实现。由于安全层服务的有状态特

性,安全层服务除安全相关业务本身的 Docker 镜像外,还需一个用于实时记录服务运行过程中各个状态的 Docker 镜像。这两个镜像容器将运行在同一个 Kubernetes Pod 中。在安全层服务部署所需的模板文件中,除了可配置服务与其上下层服务的内部通信接口外,还需要对持久化储存的声明,要求 Kubernetes 集群提供一定大小的持久化储存空间,这一空间能够被该 Kubernetes Pod 中的全部容器共享,从而使安全层服务主容器能够将相关状态信息共享至用于记录的容器。

应用层服务是对通信对象的仿真。作为测试任务中的逻辑实体,各个应用层服务应具备相互通信的接口、与外部可视化界面进行通信的接口、与日志服务通信的接口以及与自动化测试脚本之间的通信接口。在一次测试任务中,往往需要同时对多个通信对象进行仿真,因此对应用层服务的部署通常是对应用层服务

组的部署。这个服务组中除了包含各个应用层服务的 Docker 镜像外,还需要一个具备日志记录功能的 Docker 镜像。为此,应用层服务组的部署也要求 Kubernetes 集群提供一定大小的用于存放日志的持久化储存空间。此外,测试任务中如果存在进行自动化测试的需求,开发人员还需要编写相应的 DSL 脚本,并将其打包为一个 Docker 镜像加入应用层服务组之中。日志镜像与自动化测试镜像也需要具备与外部可视化程序进行通信的接口。上述镜像均应作为容器部署在同一个 Kubernetes Pod 中。在部署应用层服务所需的模板文件中,需要能够对服务与其下层服务的内部通信接口、服务与可视化界面通信的外部接口以及服务组中应用之间相互通信的接口进行配置。同时,对于每一个应用层服务,文件中均应能够制定该服务所需的安全层与通信层服务。列控系统测试容器云平台整体的开发与部署流程如图 5 所示。

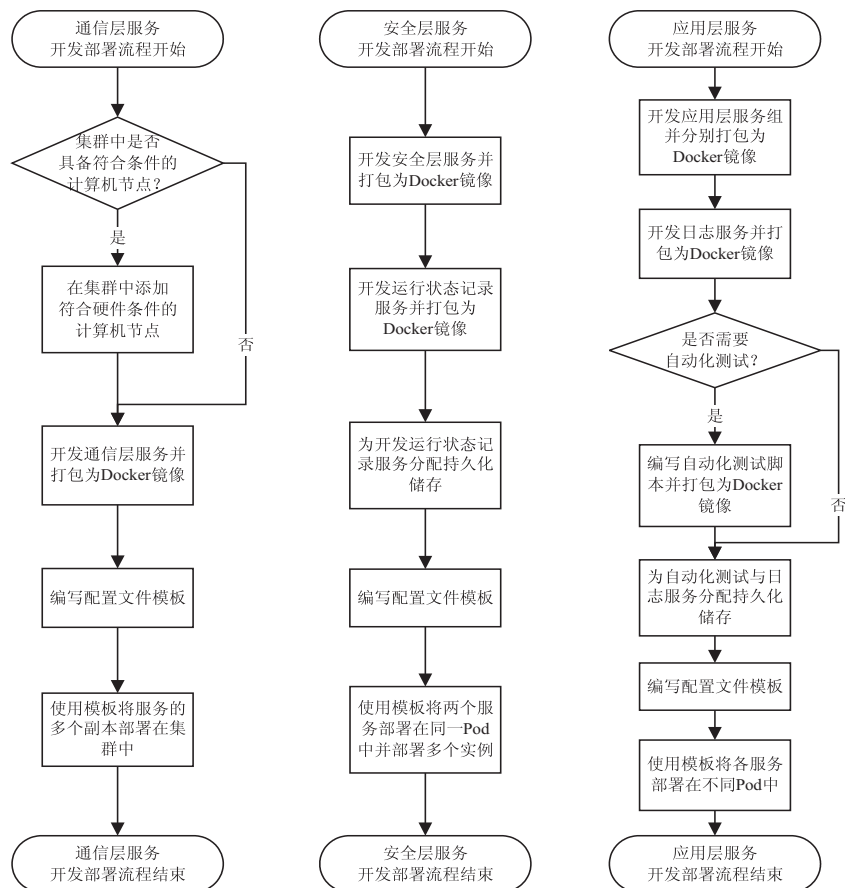


图5 列控系统测试容器云平台的研发部署流程

### 3.2 列控系统测试流程

在列控系统信号设备的测试任务中,测试人员首先需要依照上述部署流程,选择相应所需的镜像对各层服务进行部署。通过 Kubernetes 的就绪探针(readiness probe)机制能够观察各服务的就绪情况。在各层服务启动完毕后,测试人员即可在集群外部启动可视化界面,与各应用层服务、日志服务与自动化测

试服务进行连接。若测试任务具备自动化条件,测试人员可通过图形化界面执行自动化脚本;反之,测试人员需通过手动操作图形化界面依照测试大纲完成测试。在测试完成后,测试人员能够通过图形化界面获取测试结果的日志,并根据日志进行分析,完成测试任务。列控系统测试容器云平台上的测试任务流程如图 6 所示。

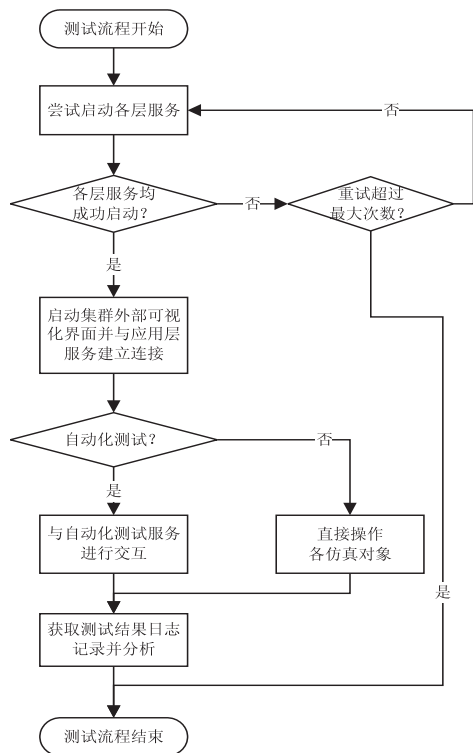


图6 列控系统测试容器云平台上的测试任务流程

#### 4 平台性能分析

列控系统测试容器云平台将列控系统测试软件的功能拆分为若干微服务,结合了 Docker 容器技术以及 Kubernetes 的容器编排能力,相比于传统的列控系统计算机测试方案,在开发效率、可靠性、自动化程度等方面均有较大幅度提升。列控系统测试容器云平台与传统方案的性能对比如表 1 所示。

表 1 列控系统测试容器云平台与传统测试方案的性能指标对比

性能指标	传统集中式方案	列控系统测试容器云平台
各功能间耦合度	代码层面的耦合	服务层面的耦合
开发效率	针对完整仿真对象进行开发,效率低	针对特定服务进行开发,效率高
自动化程度	对应用程序进行协调,自动化程度较低	对服务进行协调,自动化程度较高
故障定位与修复	故障定位在整个测试平台,修复难度大	故障定位在特定服务,修复难度低
多测试任务	需要开发多个软件,难以管理	通过服务之间的组合支持多个测试任务
可靠性	仅具备安全协议可靠性	增加了平台层面可靠性
可扩展性	需要对应用程序整体进行修改,可扩展性较弱	添加新服务或修改已有服务,可扩展性较强
跨平台移植	测试方案整体集成,难以跨平台移植	服务部署稳定,仅需移植可视化界面

#### 5 结束语

该文提出了一种基于 Docker 与 Kubernetes 集群的列控系统测试云平台,将列控系统测试软件的各类

功能作为服务打包为 Docker 镜像,并在 Kubernetes 集群中部署,通过 DSL 编写的自动化测试脚本与外部的可视化界面进行测试的方案。这一方案降低了列控系统测试中各个组件之间的耦合程度、提升了开发效率与自动化程度,同时也能够更好地支持多个测试任务的同时进行,推动列控系统测试工作向着规模化与完全自动化的方向发展。

#### 参考文献:

- [1] 刘梅,高岑,田月,等.基于 Docker Swarm 集群的调度策略优化算法[J].计算机系统应用,2018,27(9):199-204.
- [2] VERMA A, PEDROSA L, KORUPOLU M, et al. Large-scale cluster management at Google with Borg[C]//Proceedings of the tenth European conference on computer systems (EuroSys '15). Bordeaux, France: ACM, 2015:1-17.
- [3] 铁道部运输局. RSSP-I 铁路信号安全通信协议(V1.0)[S]. 北京:铁道部运输局,2010.
- [4] 刘渊,乔巍.云环境下基于 Kubernetes 集群系统的容器网络研究与优化[J].信息安全,2020,20(3):36-44.
- [5] MARKO L. Kubernetes in action[M]. Beijing: Publishing House of Electronics Industry, 2018.
- [6] 李苏雯,李鹏,张利飞,等. LTE-R 与 GSM-R 终端设备技术要求及测试方法对比[J]. 铁道技术监督, 2020, 48(6):27-32.
- [7] 张曙光. CTCS-3 级列控系统总体技术方案[M]. 北京:中国铁道出版社,2008.
- [8] 闫峰.基于 DSL 的自动化任务执行和管理工具的设计与实现[D]. 长春:吉林大学,2012.
- [9] SI X. Research on the generation method of CBTC test sequence[J]. IOP Conference Series Earth and Environmental, 2020, 440:042030.
- [10] BELGUIDOUM M, HIBA S H. A DSL for elastic component-based cloud application[J]. International Journal of High Performance Computing and Networking, 2019, 15(1/2): 58.
- [11] ANAND S, BURKE K, CHEN T Y, et al. An orchestrated survey of methodologies for automated software test case generation[J]. Journal of Systems and Software, 2013, 86(8):1978-2001.
- [12] ZHANG J M, HARMAN M, MA L, et al. Machine learning testing: survey, landscapes and horizons[J]. arXiv: 1906.10742, 2020.
- [13] 陈雁,黄嘉鑫.基于 Kubernetes 应用的弹性伸缩策略[J].计算机系统应用,2019,28(10):213-218.
- [14] 赵经纬.云平台故障检测与恢复能力测评体系研究[D]. 西安:西安电子科技大学,2019.
- [15] 刘焱.云平台下服务分布式监控系统的研究与实现[D]. 成都:电子科技大学,2020.