文章编号:1671-4598(2021)12-0220-06 DOI:10.16526/j.cnki.11—4762/tp.2021.12.040 中图分类号:V1 文献标识码:A

基于 Thrift—Eureka 改进的微服务技术研究

赵媛心，王 毅，纪祖嚣，赵俊翔，刘 萍
（北京临近空间飞行器系统工程研究所，北京 100076）

摘要：首先介绍了航天型号软件开发框架遇到的问题，分析了航天型号垂直应用框架和微服务框架的优缺点，着重介绍了目前互联网领域流行的微服务框架 Thrift—Eureka[1-2] 和 Dubbo；接着描述了互联网领域的微服务框架适配航天型号软件领域遇到的问题；然后基于微服务框架思想和 Dubbo 微服务框架，提出了一种适用于航天型号软件领域的微服务框架；该服务框架有两种使用模式，一种适用于包含数据库的型号软件系统；另一种适用于不包含数据库的型号软件系统；针对有数据库的型号软件系统，利用数据库系统来实现微服务框架的注册中心功能；针对没有数据库的型号软件系统，利用广播协议、本地日志来实现微服务框架注册中心功能。

关键词：垂直应用框架；微服务框架；Thrift—Eureka；Dubbo

Research on Improved Microservice Technology Based on Thrift—Eureka

ZHAO Yuanxin, WANG Yi, JI Zubi, ZHAO Junxiang, LIU Ping
(Beijing Institute of Nearspace Vehicle's Systems Engineering, Beijing 100076, China)

Abstract: This article first introduces the problems encountered by using the aerospace model software development framework. analyzed the advantages and disadvantages of vertical application framework and microservice framework, focused on the current popular microservice frameworks Thrift—Eureka and Dubbo in the Internet field. Then based on the idea of microservice framework and Dubbo microservice framework, a microservice framework suitable for aerospace model software field is proposed. The service framework has two usage modes, One is suitable for a model software system that contains a database, and one is suitable for a model software system that does not contain a database.

Keywords: vertical application framework; microservice framework; Thrift—Eureka; Dubbo

0 引言

航天领域是国家安全基石的重要组成部分，是多学科综合、多专业耦合、多领域覆盖的系统工程，其中的型号软件具有技术水平高、复杂度高、覆盖范围广的特点。同时，组织也在发挥型号软件中积攒的技术和管理优势，逐步扩大产品化、市场化软件的研制和推广。随着航天控制软件规模越来越大，涉及的单位越来越多，分工也越来越细化。在这种场景下所有功能在一个软件中编写实现已经不现实，多个系统、多个软件配合使用已经成为航天控制领域软件的主要模式。随之产生多系统、多软件协同调用问题越来越突出，归纳起来有以下两点：

1）调用方如何发现被调用方[7-8]；

2）被调用方启用或者关闭如何通知调用方[9-10]。

目前常用的解决方法有以下两种：

1）静态配置。在数据库或者文件中手动维护被调用方软件地址列表。如果被调用方启用或者关闭，则手动更新数据库或者文件列表。调用方每次调用服务之前，首先读取数据库或者文件，然后再调用服务。

2）引入互联网领域的微服务框架。分布式微服务框架[1-2]优点在于：

（1）一个服务对应一项业务能力，做到单一职责[11-12]；

（2）服务实现自动注册和发现；

（3）服务之间相互独立，可以实施不同的优化方案[13]。

当前互联网领域比较流行的微服务框架[3]有 Thrift—Eureka[1-2] 和 Dubbo[20]，这些微服务框架的核心原理和组件基本一致，组件包括客户端[4]（调用方软件）、服务端（被调用方软件）和注册中心[5-6]，注册中心的主要作用就是维护服务端列表，通过健康检查[14-15]和消息通知[16]及时维护服务列表，目前比较常用的注册中心是 Zookeeper[17]。

但是这两种方式在适配航天控制领域软件都遇到了困难。静态配置方式遇到的主要困难是每次软件启用、关闭都需要手动更新数据库或者文件，这就给运维人员提出了很高的要求，尤其是调用关系复杂的情况下，如何及时正确地配置调用关系就成为了系统瓶颈。

互联网领域软件与航天控制领域软件的一个重要区别是物理网络隔离。Thrift—Eureka 框架、Dubbo 比较适用于互联网领域，而不太适合航天控制领域，原因就在于这些框架依赖服务很多，部署复杂。通常航天控制软件需要在多个区域部署试用，而且各个区域的网络是物理隔离无法远程部署服务。如果需要汇报演示软件，航天控制软件自身部署需要 1 h，但是微服务框架部署可能需要 1 d。所以就需要结合互

收稿日期:2021-10-26； **修回日期**:2021-11-03。

作者简介:赵媛心(1989-)，女，安徽阜阳人，硕士，工程师，主要从事航天软件研制方向的研究。

引用格式:赵媛心，王 毅，纪祖嚣，等.基于 Thrift—Eureka 改进的微服务技术研究[J].计算机测量与控制，2021，29(12)：220-225.

联网领域微服务框架思想开发一套适用于航天控制软件领域的微服务框架，其需要具备以下特点：1）易用性高；2）功能丰富；3）依赖项少，部署简单。

1　Thrift－Eureka 微服务架构及原理

1.1　Thrift－Eureka 框架介绍

Facebook 作为全球 10 大巨头互联网公司之一，其内部包含多套软件系统。并且各个软件系统使用的开发语言互不相同，部署环境互不相同。为了打通不同系统之间的信息壁垒，串联各个信息孤岛，构建互联互通的软件系统，其开发了 Thrift 框架。在内部稳定运行一段时间后，Facebook 于 2007 年将 Thrift 框架作为一个开源项目提交给了 Apache 基金会。由于创建 Thrift 框架的原由就是解决跨平台通信问题，所以 Thrift 天生支持多种语言通信。既支持流行了很多年的开发语言 C＋＋、JAVA、C＃等，也支持当前正在流行的 Erlang、Python、Ruby 等。Thrift 框架虽然在程序语言支持方面表现优异，但是它的静态编译技术也常常被人诟病。Thrift 框架需要先定义数据结构，然后使用 Thrift 工具将数据结构转换成 IDL 文件。这就意味着，如果某个数据结构发生了变化，必须重新编译生成 IDL 文件，但是这个弱项并没有阻碍 Thrift 成为流行的微服务通信框架。

完整的 Thrift 框架包括一个客户端和一个服务端，首先用户需要自定义数据结构，然后使用 Thrift 工具自动生成客户端和服务端代码，并且自动生成的代码中包含接口协议字段，以便客户端与服务端进行通信。在通信过程中 Thrift 会将数据信息转换成高效的二进制字节来提高传输效率。服务器端会利用底层多线程、阻塞或者非阻塞协议来接收数据。具体采用哪种协议基于操作系统不同而不同。

通过上面介绍可以发现 Thrift 框架是客户端与服务端直接通信的，这就会导致以下问题。

1）服务的端口或者 IP 发生变化，调用方需要手动修改 IP 或端口；

2）新启服务无法实现动态发现；

3）服务之间调用关系错综复杂，难以维护。

针对以上缺点互联网领域通常将 Thrift 与 Eureka 配合使用。Eureka 是 Netflix 开发的服务发现框架，本身是一个基于 REST 的服务，以达到负载均衡和中间层服务故障转移的目的。

Eureka 是 Netflix 开发的服务发现框架，其有两部分构成，即服务端和客户端。服务端主要提供用户服务注册功能与更新功能，即用户服务启动之后，会将自身服务信息注册到 Eureka 的服务端中。每隔 30 s 用户服务节点会发一次心跳到 Eureka 的服务端中，如果 Eureka 服务端连续多个心跳周期（一般为 3 个周期）都没有收到某个服务节点的心跳，那么 Eureka 服务端就认为该服务节点不可用，然后将其剔除。通过这样的机制 Eureka 的服务端就可以总览所有服务节点信息，并且快速识别可用的服务列表。Eureka 客户端主要提供服务订阅功能和与服务端的连接封装管理功能。即利用 Eureka 客户端上层应用可以使用服务名称与其他应用进行通信，而不用通过配置 IP 与其他应用进行通信。Eureka 客户端还内置了轮询算法来实现负载均衡功能。Thrift－Eureka 微服务框架如图 1 所示。

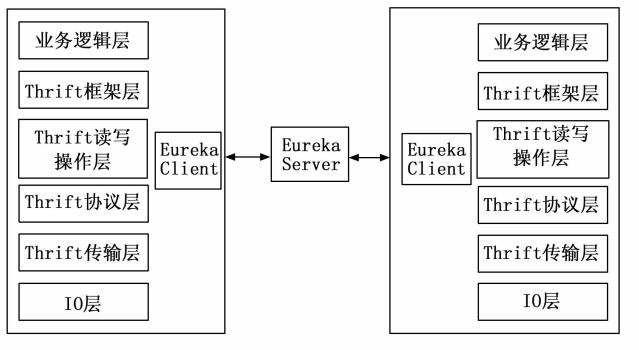

图 1　Thrift－Eureka 微服务框架图

由图 2 所示，Thrift－Eureka 框架的调用逻辑分为以下几步：

1）在注册中心中配置服务端名称，在客户端中配置其需要订阅的服务名称；

2）服务启动之后，服务端将本地 IP 和服务信息发送给注册中心；

3）注册中心将服务端 IP 和服务发送给客户端；

4）客户端基于注册中心发来的消息，调用服务端的接口。

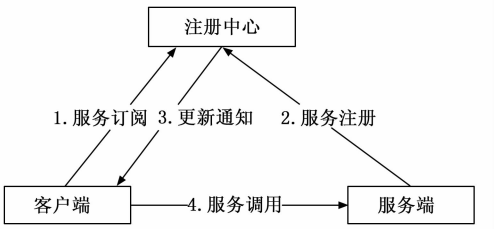

图 2　Thrift－Eureka 框架调用图

1.2　Dubbo 框架介绍

Dubbo 最早诞生于阿里巴巴，随后加入 Apache 软件基金会，项目从设计之初就是为了解决企业的服务化问题，因此充分考虑了大规模集群场景下的服务开发与治理问题，如易用性、性能、流量管理、集群可伸缩性等。在 Dubbo 开源的将近 10 年时间内，Dubbo 几乎成为了国内微服务框架的首选框架，尤其受到大规模互联网、IT 企业的认可，可以说作为开源服务框架，Dubbo 在支持微服务集群方面有着非常大的规模与非常久的实践经验积累，是最具有企业规模化微服务实践话语权的框架之一。采用 Dubbo 的企业涵盖互联网、传统 IT、金融、生产制造业多个领域，一些典型用户包括阿里巴巴、携程、工商银行、中国人寿、海尔、金蝶等。

Dubbo 作为一款微服务开发框架，它提供了远程服务调用与微服务治理两大关键能力。利用 Dubbo 提供的丰富

服务治理能力，可以实现诸如服务发现、负载均衡、流量调度等服务治理诉求。同时 Dubbo 是高度可扩展的，用户几乎可以在任意功能点去定制自己的实现，以改变框架的默认行为来满足自己的业务需求，Dubbo 框架如图 3 所示。

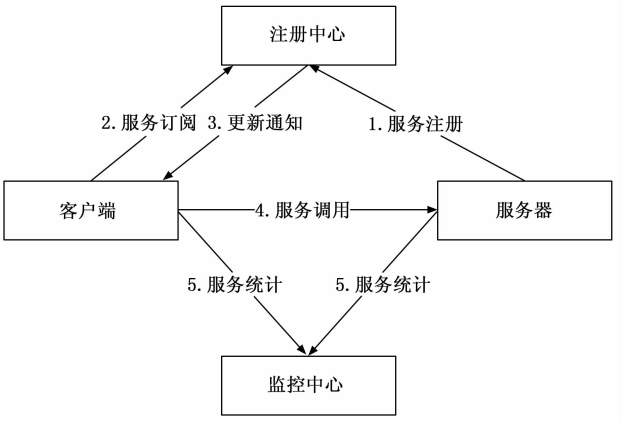

图 3 Dubbo 框架调用图

自开源以来，Dubbo 就被一众大规模互联网、IT 公司选型，经过多年企业实践积累了大量经验。因此，Dubbo 在解决业务落地与规模化实践方面有着无可比拟的优势：

1）易用性高，如 Java 版本的面向接口代理特性能实现本地透明调用；

2）功能丰富，基于原生库或轻量扩展即可实现绝大多数的微服务治理能力；

3）高性能的跨进程通信协议；

4）地址发现、流量治理层面，轻松支持百万规模集群实例。

Dubbo 提供了从服务定义、服务发现、服务通信到流量管控等几乎所有的服务治理能力，并且尝试从使用上对用户屏蔽底层细节，以提供更好的易用性。定义服务在 Dubbo 中非常简单与直观，可以选择使用与某种语言绑定的方式（如 Java 中可直接定义 Interface），也可以使用 Protobuf IDL 语言中立的方式。无论选择哪种方式，站在服务消费方的视角，都可以通过 Dubbo 提供的透明代理直接编码。

点对点的服务通信是 Dubbo 提供的一项基本能力，Dubbo 以远程服务调用的方式将请求数据（Request）发送给后端服务，并接收服务端返回的计算结果（Response）。远程服务调用对用户来说是完全透明的，使用者无需关心请求是如何发出去的、发到了哪里，每次调用只需要拿到正确的调用结果就行。同步的 Request－Response 是默认的通信模型，它最简单但却不能覆盖所有的场景，因此，Dubbo 提供更丰富的通信模型：

1）客户端异步请求（Client Side Asynchronous Request－Response）；

2）服务端异步执行（Server Side Asynchronous Request－Response）；

3）客户端请求流（Request Streaming）；

4）服务端响应流（Response Streaming）；

5）双向流式通信（Bidirectional Streaming）。

Dubbo 的服务发现机制，让微服务组件之间可以独立演进并任意部署，客户端可以在无需感知对端部署位置与 IP 地址的情况下完成通信。Dubbo 提供的是 Client－Based 的服务发现机制，使用者可以使用 Zookeeper 作为服务发现组件。

Dubbo 提供了包括负载均衡、流量路由、请求超时、流量降级、重试等策略，基于这些基础能力可以轻松的实现更多场景化的路由方案，包括金丝雀发布、A/B 测试、权重路由、同区域优先等。Dubbo 强大的服务治理能力不仅体现在核心框架上，还包括其优秀的扩展能力以及周边配套设施的支持。

1.3 Thrift－Eureka 框架与 Dubbo 框架对比

Thrift－Eureka 需要使用 API（接口定义语言）定义一个文件，然后通过 Thrift 来生成对应的各种语言的代码，服务的提供者和消费者都需要把这个代码引入进去，服务端负责接口的实现，消费者使用 API 的存根去直接调用提供者。目前 Thrift－Eureka 支持的语言比较多，比如常见的 C＋＋、Java、Python、Ruby、C＃、PHP 等。Thrift 属于 C/S 模式，它通过代码生成工具将接口定义的文件生成服务器端和客户端的代码，当然，服务端和客户端可以是不同语言的，从而实现了客户端和服务端之间跨语言的支持。传输的序列化也支持很多种，比如：二进制模式、压缩模式、JSON 模式以及 Debug 模式，可以在开发的过程中使用 Debug 模式和 JSON 模式来方便的看到传输的数据，在正式环境使用二进制模式或压缩模式来提升性能。在通讯模式上也支持很多种，它支持阻塞的 I/O 和 NIO，还有专门用来传输文件的传输方式。在线程模型上，它也支持很多种，比如简单的单线程模式、线程池模式、多线程使用非阻塞 I/O 模式。这样可以在不同的业务场景中使用更适合的技术来实现微服务调用。Thrift 并没有服务治理相关的功能，消费者只能使用服务提供者 IP 和端口号来进行访问，而 Eureka 具有服务注册与发现功能，需要将 Thrift 和 Eureka 配合使用，来作为一个完整的微服务框架。

Dubbo 的消费者和提供者以及注册中心之间使用的是长连接，服务的消费者和提供者会在内存中累计服务调用的次数，并且定时将调用的信息发送到监控中心；消费者和提供者之间通信采用的是非阻塞 I/O（即 NIO）；Dubbo 的序列化使用的是阿里修改过的 hession 序列化方式，Dubbo 是基于 Java 来进行开发的，所以也支持 java 的客户端和服务端。Dubbo 具备完善的服务注册、服务订阅、通知以及监控功能。

总体而言，Dubbo 作为一个完善的微服务框架，具备丰富的工具。但是它的缺点在于只支持 Java 语言。Thrift－Eureka 搭配使用也能作为一个完善的微服务框架，其具备的工具集合满足航天型号软件使用需求，而且其支持的语言非常丰富。综合比较下来，系统选择改造 Thrift－Eureka 微服务框架来适配航天型号软件。

2　Thrift－Eureka 改进的微服务架构

结合引言中介绍的航天型号软件使用场景，即经常需要在不同场地进行试用，并且各个场地之间网络不通。如果型号软件部署需要 1 小时，但是微服务框架部署需要 1 天，那就是本末倒置了。针对这种情况，微服务框架改造的重点在于不增加使用难度的情况下，缩短部署时间。

2.1　含有数据库的航天型号系统

结合上文中提到的改造目标，如果能够将注册中心和监控中心的功能进行转移，并且将客户端软件和服务端软件做成依赖包的形式嵌入型号软件中，那么就达到了在不增加使用难度的情况下，减少部署时间的目标。目前 Thrift－Eureka 的客户端软件和服务端软件可以作为依赖包嵌入型号软件。再分析一下注册中心和监控中心的功能，注册中心的功能主要是服务注册、服务更新通知和服务发现，监控中心的主要功能是客户端指标收集和服务端指标收集。

1）服务注册：服务提供方将自身服务地址写入注册中心；

2）服务更新通知：当服务提供方 IP 或者端口发生变化后，及时通知客户端更新服务列表；

3）服务发现：客户端读取注册中心中的服务列表；

4）客户端指标收集：客户端将调用成功次数、失败次数及耗时情况写入监控中心；

5）服务端指标收集：服务端将调用成功次数、失败次数及耗时情况写入监控中心。

在原有数据库的基础上增加两张数据库表，就可以来承载注册中心和监控中心的职责，如图 4 所示。

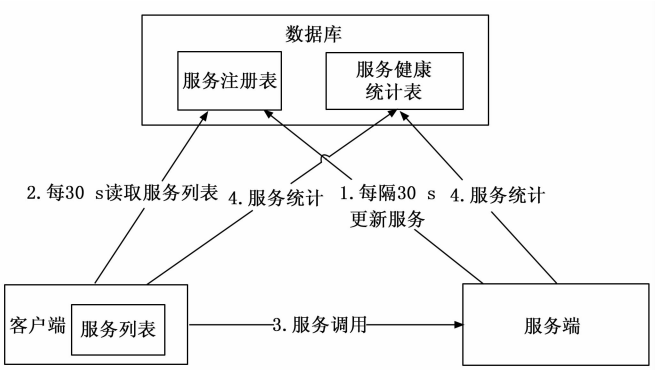

图 4　包含数据库型号软件微服务框架

根据图 4 可知，包含数据库的型号软件微服务架构原理概述如下：

首先在原有数据库的基础上增加两张表，分别为服务注册表和服务健康统计表。服务注册表主要包括服务提供方 IP、端口、最近更新时间。服务健康统计表主要包括机器 IP、端口、服务名称、调用次数、失败次数、类型（服务端、客户端）等。

然后修改服务端的服务注册逻辑，每隔 30 s 更新一次服务注册表中的服务信息。

再次在原有客户端的基础上，基于 guava 的 Localcache 构建一套本地缓存，每隔 30 s 从数据库中获取一次服务提供方列表，并且将更新时间超过 90 s 的服务剔除。并且基于客户端缓存的服务列表来实现负载均衡、流量路由、重试等策略等功能。

最后客户端和服务端都将服务统计数据上传到服务健康统计表中。

对照 Thrift－Eureka 的注册中心和监控中心，重新梳理了以下功能：

1）服务注册：服务提供方每隔 30 s 将自身服务地址和服务器时间更新到服务注册表；

2）服务更新通知：客户端通过本地缓存每隔 30 s 拉取一次服务注册表信息，来达到服务更新通知的目标；

3）服务发现：客户端读取本地缓存的服务列表；

4）客户端指标收集：客户端将调用成功次数、失败次数及耗时情况写入服务健康统计表；

5）服务端指标收集：服务端将调用成功次数、失败次数及耗时情况写入服务健康统计表。

2.2　不包含数据库的航天型号系统

上一章节介绍了如何基于数据库改造 Thrift－Eureka，下面介绍不利用数据库改造微服务框架的方法。即可以通过服务端发送广播本地服务消息的方式来实现注册中心的功能，客户端发送广播获取服务消息的方式来实现服务发现的功能，如图 5 所示。

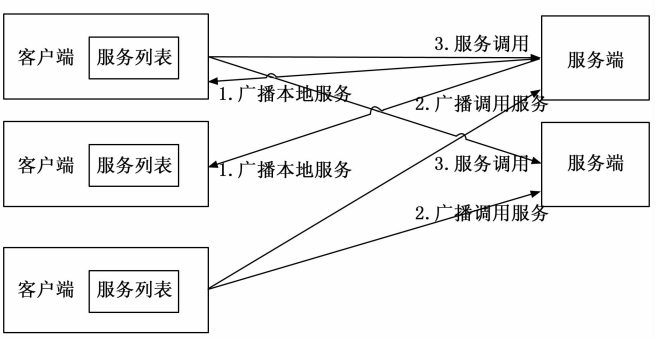

图 5　不包含数据库型号软件微服务框架

根据图 5 所示，不包含数据库的微服务架构原理概述如下：首先服务端启动之后，在局域网内广播本地服务地址和服务信息，其他服务器收到消息之后，判断该消息包含的服务信息是否是自己订阅的。如果是自己订阅的，则更新本地服务列表。如果不是自己订阅的，则忽略这条广播消息。然后服务端每隔 3 min 广播一次本地服务地址和服务消息，其他服务器收到消息后处理逻辑与第一次收到消息处理逻辑一致。

然后客户端启动，在局域网内广播自己需要的服务地址，其他服务器收到消息之后，判断该消息包含的服务信息是否是自己提供的。如果是自己提供的，则将本地服务地址和服务信息发送给客户端。如果不是自己提供的，则忽略这条广播消息。

再次在原有客户端的基础上，基于 guava 的 Localcache 构建一套本地缓存，主要存储服务地址和服务更新时间，并且将更新时间超过 3 min 的服务剔除。并且基于客户端缓存的服务列表来实现负载均衡、流量路由、重试等策略等功能。

最后客户端和服务端都将服务统计数据存储到本地日志中，方便后续定位分析问题。

对照 Thrift－Eureka 的注册中心，重新梳理了以下功能：

1）服务注册：服务提供方启动之后将自身服务地址和服务器时间广播给局域网内其他机器；

2）服务更新通知：服务提供方每隔 3 min 将自身服务地址和服务器时间广播给局域网内其他机器；

3）服务发现：客户端读取本地缓存的服务列表。如果本地缓存列表为空，则广播发送获取服务消息。

2.3 航天型号系统微服务框架使用模式

基于上述章节，航天型号系统微服务框架内置了两套使用模式。模式一针对包含数据库的型号系统，它具有可靠性高、问题排查方便的特点。模式二针对不包含数据库的型号系统，它具有依赖少，使用范围广泛的特点。系统会判断用户是否设置了使用模式，如果用户有自定义设置，则按照用户要求来使用系统。否则按照如下步骤自动启动框架。

2.3.1 连接数据库

判断数据库能否连接成功，如果能连接成功，则跳转步骤 2）。否则使用广播模式，如下所示：

1）服务端广播服务信息：服务端每隔 3 min 将本地服务信息广播给局域网内其他机器。

2）客户端获取服务信息：客户端判断本地缓存的服务列表是否为空，为空则广播发送获取服务消息。

2.3.2 创建服务注册表

判断数据库中是否已经存在服务注册表。如果存在，则跳过。否则创建服务注册表。

2.3.3 服务端每隔 30 s 更新服务注册信息

服务端每隔 30 s 将本地服务地址和服务器时间更新到服务注册表中。

2.3.4 客户端每隔 30 s 拉取服务注册信息

客户端每隔 30 s 从服务注册表中拉取服务信息，并更新本地缓存。

通过上述步骤可以得出，系统默认优先使用数据库模式，在数据库无法连接的情况则使用广播模式。

3 试验结果与分析

Thrift－Eureka 改进的微服务架构主要解决的是航天型号软件部署维护复杂的问题，为此设计了对照实验来比较原型 Thrift－Eureka 框架和改进的 Thrift－Eureka 框架的部署耗时。首先针对某航天型号软件 A，分别使用 Thrift－Eureka 框架构建系统 B，使用改进的 Thrift－Eureka 框架构建系统 C。

从开发组、测试组和运维组总共抽取了 45 名同事，将这 45 名同事随机分成 5 个队，每队 9 人，包括 3 名开发、3 名测试和 3 名运维同事。

每队需要同时部署系统 A、系统 B 和系统 C。耗时情况如表 1 所示。

表 1 系统部署耗时比较 s

部署耗时	实验队 1	实验队 2	实验队 3	实验队 4	实验队 5	平均耗时
系统 A	1	1.5	1.15	2	1.75	1.48
系统 B	1.5	2	2	2.75	2.5	2.15
系统 C	1.05	1.5	1.2	2	1.8	1.51

从表 1 中可以看出部署系统 A 和系统 C 的耗时基本相同，这符合实验预期。因为改进的 Thrift－Eureka 框架构不需要额外部署软件，实验队员部署系统 A 和系统 C 的步骤基本相同。但是部署系统 B 的耗时明显增加，这是因为除了部署基础的型号系统 A 之外，还需要部署配置 Thrift－Eureka 框架。

4 结束语

首先介绍了航天型号软件开发框架遇到的问题，分析了航天型号垂直应用框架和微服务框架的优缺点，着重介绍了目前互联网领域流行的微服务框架 Thrift－Eureka 和 Dubbo。接着描述了互联网领域的微服务框架适配航天型号软件领域遇到的问题。然后基于微服务框架思想和 Dubbo 微服务框架，提出了一种适用于航天型号软件领域的微服务框架。该服务框架有两种使用模式：一种适用于包含数据库的型号软件系统；另一种适用于不包含数据库的型号软件系统。针对有数据库的型号软件系统，利用数据库系统来实现微服务框架的注册中心功能。针对没有数据库的型号软件系统，利用广播协议、本地日志来实现微服务框架注册中心功能。经在某项目中实践后可见，该方案能够实际解决航天型号软件微服务框架使用的问题。

参考文献：

[1] 吴 洲. 基于 Thrift 的跨编程语言 Flex 应用框架研究 [J]. 计算机与现代化，2013 (5)：181－185.

[2] 梁明炯. 基于 Thrift 框架的数据交换方案 [J]. 科技创新与应用，2015 (13)：66－67.

[3] 张 晶，黄小锋，李春阳. 微服务框架的设计与实现 [J]. 计算机系统应用，2017，26 (6)：259－262.

[4] 洪华军，吴建波，冷文浩. 一种基于微服务架构的业务系统设计与实现 [J]. 计算机与数字工程，2018，46 (1)：149－154.

[5] 龙新征，彭一明，李若森. 基于微服务框架的信息服务平台 [J]. 东南大学学报（自然科学版），2017，47 (z1)：48－52.

[6] CIUFFOLETTI，AUGUSTO. Automated deployment of a microservice－based monitoring infrastructure [J]. Procedia Computer Science，2015，68：163－172.

[7] VILLAMIZAR M，GARCES O，CASTRO H，et al. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud [C] // Computing Colombian Conference，IEEE，2015：583－590.

[8] SURYOTRISONGKO H，JAYANTO D P，TJAHYANTO A.

Design and development of backend application for public complaint systems using microservice spring boot [J]. Procedia Computer Science, 2017, 124: 736 - 743.
[9] HASSELBRING W, STEINACKER G. Microservice architectures for scalability, agility and reliability in E－commerce [C] //IEEE International Conference on Software Architecture Workshops, IEEE, 2017: 243 - 246.
[10] SCHROEDER M D, BURROWS M. Performance of the Firefly RPC [J]. ACM Transactions on Computer Systems, 1990, 8 (1): 1 - 17.
[11] WHITEHOUSE K, TOLLE G, TANEJA J, et al. Marionette: using RPC for interactive development and debugging of wireless embedded networks [C] // International Conference on Information Processing in Sensor Networks, IEEE, 2006.
[12] 许卓明，栗　明，董逸生. 基于 RPC 和基于 REST 的 Web 服务交互模型比较分析 [J]. 计算机工程，2003，29 (20)：6 - 8.
[13] 冯新扬，沈建京. REST 和 RPC：两种 Web 服务架构风格比较分析 [J]. 小型微型计算机系统，2010，31 (7)：1393 - 1395.
[14] 戴亚娥，俞成海，尧飘海. 基于 REST 架构风格的 Web2.0 实现 [J]. 计算机系统应用，2009 (2)：165 - 168.
[15] 丁振凡. Spring REST 风格 Web 服务的 Json 消息封装及解析研究 [J]. 智能计算机与应用，2012，2 (2)：9 - 10.
[16] VINOSKI S. RPC and REST：Dilemma，disruption，and displacement [J]. IEEE Internet Computing，2008，12 (5)：92 - 95.
[17] MCGOVERN J，TYAGI S，STEVENS M E，et al. JAX－RPC－Java web services architecture－chapter {10} [J]. Java Web Services Architecture，2003：313 - 403.
[18] FENG X，SHEN J，FAN Y. REST：An alternative to RPC for Web services architecture [C] // First International Conference on Future Information Networks，IEEE，2009.
[19] WEI P C，NTAFOS S. The zookeeper route problem [J]. Information Sciences，1992，63 (3)：245 - 259.
[20] 谢璐俊，杨鹤彪. 基于 Dubbox 的分布式服务架构设计与实现 [J]. 软件导刊，2016，15 (5)：13 - 15.

(上接第 208 页)

通过实际综合化机载 CNS 系统研制可以得出，基于面向切面思想的系统设计方法，采用关注点分离技术识别并分别独立实现 VHF 话音、ATC 航管应答等无线电功能和参数调谐、日志管理等非功能性需求，再通过组件编制技术将二者集成为统一的无线电系统。实际系统测试表明，综合后的 CNS 系统能够实现预期的无线电功能。增加 ACARS 数据链功能的变更设计过程表明，变更的影响被限制在新增模块和系统蓝图这两个部分，其余设计无需变更。最后通过维护性和适航性进行综合评估，可以得出本文提出的方法能够提升系统的维护性，有助于产品适航符合性验证，可以为射频综合 CNS 系统适航性研究奠定基础。

参考文献

[1] ARINC 660A，CNS/ATM Avionics functional allocation and recommended architetures [S]. SAE Industry Technologies Consortia，2001.
[2] 薛　慧，张　昊. 机载多功能综合射频一体化发展研究 [J]. 中国电子科学研究院学报，2016，11 (5)：532 - 539.
[3] 薛　慧，王　虎. 舰载多功能综合射频一体化研究发展现状 [J]. 飞航导弹，2016 (8)：46 - 50.
[4] ELRAD T，FILMAN R E，BADER A. Aspect－oriented programming：Introduction [J]. Communications of the ACM，2001，44 (10)：29 - 32.
[5] 何丽莉，金淳兆，冯　铁，等. 关注分离问题研究综述 [J]. 计算机科学，2005，32 (2)：129 - 132.
[6] 鲍　陈，汪千松. 基于 AOP 的实时系统关注点分离方法 [J]. 计算机工程与设计，2011 (9)：3082 - 3086.
[7] 邓阿群，厉小军，俞欢军，等. 一种新型软件设计方法 AOP 的研究 [J]. 系统工程与电子技术，2004 (7)：112 - 117.
[8] MOIR L. 军用航空电子系统 [M]. 北京：电子工业出版社，2008.
[9] 刘盛彬. 军用飞机航空电子系统总体技术 [J]. 科学技术创新，2019 (25)：54 - 55.
[10] 雷国志. 航空电子需求分解技术及其算法实现 [J]. 军民两用技术与产品，2015 (4)：6.
[11] 孙文杰. 基于功能综合的模块化通信导航识别系统总体设计 [J]. 电讯技术，2016，56 (1)：67 - 70.
[12] 范欢欢，伍小保，孙维佳. 一种射频数字一体化宽带收发模块设计 [J]. 雷达科学与技术，2020，18 (3)：340 - 344.
[13] 赵　冬，叶克江. 对时间输入/输出自动机一致性测试的改进 [J]. 郑州大学学报 (理学版)，2002 (4)：30 - 33.
[14] DORI D. Object－process methodologya holistic systems paradigm [M]. Springer Berlin Heidelberg，2002.
[15] 郑　展，姚　剑，杨　峰，等. 基于 OPM 风险管理的武器系统设计方法 [J]. 系统仿真学报，2017，29 (9)：2000 - 2008.
[16] 张瞩熹. 基于 AOP 技术的通用线程监控平台的研究与实现 [J]. 计算机工程与科学，2007，29 (5)：120 - 122.
[17] 蒋兴城，曹　力，邓雪云，等. 基于 MSK 的地空数据链通信调制解调方法 [J]. 信息技术，2012 (8)：5 - 8.
[18] 张力支. 机载甚高频 ACARS 数据链系统及通信管理单元设计 [J]. 电讯技术，2011，51 (12)：101 - 104.
[19] 中国国家标准化管理委员会. 系统与软件维护性 [S]. GB/T 29834 - 2013，2013.
[20] 胡玉农，夏正洪，王俊峰，等. 复杂电子信息系统效能评估方法综述 [J]. 计算机应用研究，2009，26 (3)：25 - 28.
[21] 雷国志，王　雷，王涛涛. 民机 CNS 系统适航验证方法分析与实践 [J]. 电讯技术，2014，54 (7)：996 - 1001.
[22] SAE ARP4754A. Guidelines for development of civil aircraft and systems [S]. SAE Industry Technologies Consortia，2010.
[23] SAE ARP4761. Guidelines and method for conducting the safety assessment process on civil airborne systems and equipment [S]. SAE Industry Technologies Consortia，1996.
[24] 黄家成，蒋晓松. 机载通信导航设备综合检测系统的研制 [J]. 计算机测量与控制，2020，18 (11)：132 - 134 .
[25] 王　凯. 航电设备运行环境动态模拟系统设计方法研究 [J]. 计算机测量与控制，2014，22 (4)：337 - 340.