 DOI: https://doi.org/10.12349/iser.v7i2.9259

Microservices-Oriented Design Scheme for Embedded Platforms in the Communication Domain

Tianlong Sun

The 10th Research Institute of CETC, Chengdu, Sichuan, 610036, China

Abstract

In the field of communication, to rapidly adapt to the frequent demands for embedded software iteration and updates, this paper proposes a microservices architecture design for embedded software platforms based on the principles of open system architecture. The proposed solution builds upon a layered architectural model, where each layer is decomposed into independent microservice components. Communication between layers is achieved through standardized service interfaces, while interactions among service components are facilitated by lightweight microservices communication middleware and event-driven communication middleware, enabling effective module decoupling. The embedded software platform developed based on this architecture achieves high cohesion and low coupling within its functional components, supports independent testing, deployment, and rapid development iterations, As a result, embedded software microservicesization has become a core technological approach for constructing next-generation communication systems.

Keywords

Microservices-oriented design; Software platform design; Interface design; Module decoupling

面向通信领域的嵌入式平台微服务化设计方案

孙天龙

中国电子科技集团公司第十研究所，中国 · 四川 成都 610036

摘　要

在通信领域，为了能够快速适应嵌入式软件迭代更新的频繁需求，本文基于开放式系统架构思想研究设计了嵌入式软件平台的微服务架构方案，该方案在分层架构基础上，设计每一层划分为微服务组件，层与层间通过标准服务接口进行通信，服务组件间通过轻量级微服务通信中间件和事件通信中间件等通信方式进行模块解耦合设计，基于该方案研发的嵌入式软件平台，能够实现软件内部功能组件高内聚低耦合，支持独立测试、部署和快速开发迭代，嵌入式软件微服务化已成为构建新一代通信系统的核心技术途径。

关键词

微服务化设计；软件平台架构；接口设计；模块解耦

1 引言

随着通信技术的不断发展，对软件平台能够快速适应外部新需求变化、快速迭代开发和灵活部署提出了更高的需求，传统单体分层架构虽然具有容易构建、初期开发快等优势，但其无法快速拓展、软件成熟度低，后续的运维成本较高。

本方案在传统分层架构基础上，进行微服务组件化设计，在此架构中不同的服务通过标准化的“服务接口契约”进行交互，微服务组件可以独立开发、测试和部署，具有高拓展、低耦合、敏捷迭代开发等优势，尤其在军用领域能够快速适应外部变化，使得技术和人力资源能够合理的进行优化配置，因此军用嵌入式软件平台正加速向云原生方向发展和转型。

2 总体设计原则和微服务组件定义

2.1 总体设计原则

方案的总体设计原则包括：1）微服务组件划分合理，每个微服务组件需要根据领域驱动设计方式进行业务划分，具有独立的上下文，服务划分不能过粗或过细；2）接口标准化和规范化，所有微服务间的接口应该遵循统一的接口标准；3）可独立开发、测试部署和运维，微服务组件能够独立进行开发测试以及升级、各服务组件间应该解耦合，不能相互影响；4）服务可复用，微服务组件应该遵循标准化接口设计，设计时应考虑系统级、组件级的业务服务；5）混合通信模式，嵌入式软件要考虑实时和非实时场景，根据业务适应性使用同步和异步的通信方式。

【作者简介】孙天龙（1991–），满族，中国黑龙江绥化人，硕士，工程师，从事通信软件系统设计研究。

2.2 微服务组件定义

微服务组件是微服务架构的软件实体，软件组件的属性一般设计包括：

1）微服务组件名称；2）微服务组件标识，唯一的 ID 号进行索引；3）微服务组件地址，微服务架构设计中，服务地址作为服务组件的通信地址，如 URL 地址；4）版本号：每个组件应该一定阶段具有它的版本号。

3 架构设计方案

3.1 软件平台架构

本方案嵌入式软件平台微服务化设计主要采用面向服务的软件架构进行服务组件的设计以及服务组件间的通信，软件架构按照平台资源层、平台通信层、应用服务层进行划分，各层通过标准化接口解耦，服务组件间通过服务方式通信，软件架构如图 1 所示。

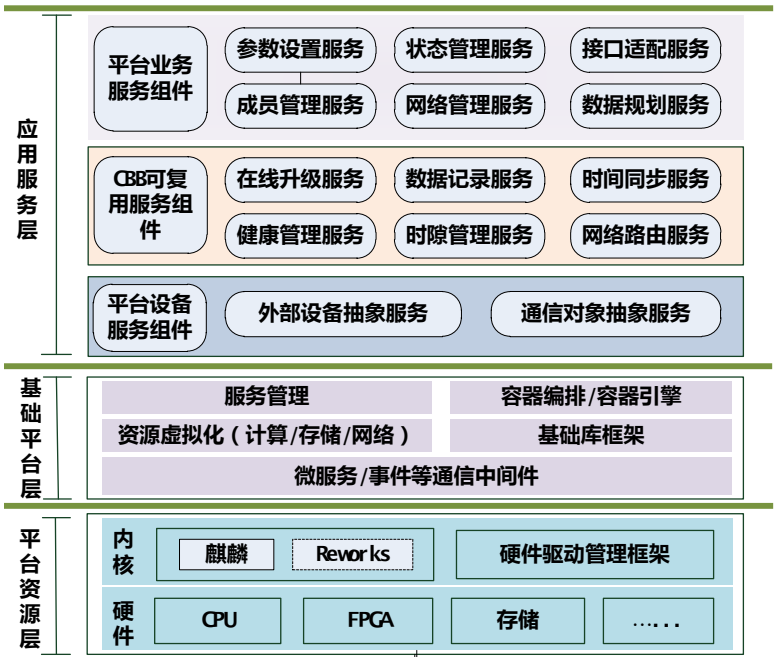

图 1 软件平台架构设计

3.2 应用服务层

3.2.1 平台业务服务组件

平台业务服务组件按照实际业务需求进行划分，如控制管理类软件划分为参数设置服务，状态监视服务等，消息处理类软件划分为消息适配、消息处理规则、消息打包、消息转译等，通过对相关功能进行抽象和服务化封装，对外提供标准化的服务接口。

3.2.2 CBB 可复用服务组件

CBB 可复用服务组件作为通用组件，与硬件平台和平台业务解耦合，其只调用操作系统底层接口和基础平台层接口，核心业务功能与特定的平台无关，可以被反复使用，不断开发迭代，逐渐形成软件产品化，从而提升软件的稳定性。例如平台通用服务只能调用平台基础框架获取平台相关信息，不能耦合平台其他模块业务。

3.2.3 平台设备服务组件

平台设备服务组件主要将外部设备、通信对象等进行抽象和封装，提供规范化的服务调用接口及基础平台通信能力。其主要作为应用层的服务组件，作为中间层隔离了物理外设，起到隔离硬件或特定平台的目的。如与外部系统交互，则抽象一个外部系统通信对象；如与底层链路进行交互，则抽象一个底层链路通信对象。

3.3 基础平台层

包括基础库框架、服务管理、容器、资源虚拟化、通信中间件等，为上层提供平台层的调度、总线通信、服务接口等基础服务。如通过平台基础库框架、通信中间件等实现跨平台通信，使得应用层只关心业务，不关心平台。

3.4 平台资源层

针对不同硬件平台提供标准统一的硬件驱动管理，能够适配 reworks、麒麟、欧拉、天脉等操作系统，对上层提供统一资源层接口。

4 平台接口设计

本方案接口设计主要采用面向对象的 C++ 语言进行描述。

4.1 通用服务接口设计

通用服务接口设计主要目的是为所有微服务组件提供标准化的统一服务接口，以实现跨平台、可重用的软件架构。定义通用服务组件基类 CServiceComponentBase，该类接口设计包括：1）Initial 函数；2）ServiceStart 函数；3）ServiceStop 函数；4）ServiceDestroy 函数；5）GetServiceConfigInfo 函数；6）GetServiceState 函数；7）ServiceInterfaceCall 函数，

其中 Initial 函数为服务组件初始化调用接口，包括配置服务信息、进行服务注册等，ServiceStart 函数、ServiceStop 函数、ServiceDestroy 函数为服务启停接口，用于实现组件调度管理。GetServiceConfigInfo 函数和 GetServiceState 函数主要用于过去服务配置信息和服务当前状态。ServiceInterfaceCall 函数是通用操作入口，用于调用具体业务逻辑。

4.2 通信服务接口设计

通信服务接口设计主要目的为平台提供标准化、服务化的通信能力抽象，通过订阅发布机制实现与业务逻辑解耦合，主要与外部系统、进程间的服务组件通信，如与外部系统软件、内部设备等通过通信服务组件进行通信。应用服务层的平台设备服务组件一般需满足通信服务接口设计规范。定义通信服务组件基类 CComServiceComponentBase，该类接口设计包括：1）sendData 函数；2）OnRecvData 函数；3）PublishTopic 函数；4）SubScribeTopic 函数，其中 send Data 函数和 OnRecvData 函数接口主要实现通信对象的收发接口，如通过 FC 总线、IO 通道、设备通道等与其他软件进行通信。PublishTopic 函数和 SubScribeTopic 函数主要用于与其他非通信对象服务组件间通过微服务等通信中间件实现，主要用于服务间数据收发。

4.3 应用服务层接口设计

应用服务层主要为平台具体业务流程，一般业务流程可分为几个事件流程步骤，因此可定义事件类服务组件，事件服务组件间的通信需要使用事件发布订阅框架，事件服务组件与非事件服务组件间的通信通过主题发布订阅，通过订阅发布机制实现与业务逻辑解耦合。定义事件类服务组件基类 CEventServiceComponentBase，该类接口设计包括：1）PublishEvent 函数；2）SubScribeEvent 函数；3）Event CallFunc 函数；4）PublishTopic 函数；5）SubScribeTopic 函数；6）TopicCallFunc 函数，其中 PublishEvent 函数和 SubScribeEvent 函数主要进行事件服务组件间通信，用于业务流程步骤通知。EventCallFunc 函数为事件订阅后的回调函数。SubScribeTopic 函数和 TopicCallFunc 函数主要进行事件服务组件与非事件服务组件间的通信，用于服务间数据收发。

4.4 基础平台层服务接口设计

基础平台层接口设计要符合标准化与一致性，一般设计包括服务认证接口、配置中心接口、服务注册发现接口、通信中间件数据收发接口、API 网关接口等。

4.5 平台资源层接口设计

该层通过统一封装设计，定义统一的硬件驱动管理框架接口，对上提供接口服务，一般使用标准的 POSIX 接口，包括 Open 函数、Ioctl 函数、Write 函数、Read 函数、Close 函数等。

5 结语

在面向通信领域的嵌入式软件平台架构设计中，微服务架构设计无疑具有重要的意义。本文给出了一种适用于通信领域的嵌入式软件平台微服务架构设计方案，该方案对微服务组件进行了详细的定义，设计了一种嵌入式软件平台微服务化的开放式架构，使用该架构进行嵌入式软件设计，能够隔离平台变化，便于软件平台的拓展式和迭代式演进，从而提高软件的鲁棒性、安全性和可靠性。

参考文献

[1] 尹伟，缪万胜，王念伟，等. 大规模复杂系统的开放式软件架构研究[J].航空电子技术，2017,48(6):23-29.

[2] 王 亮，王 璇，谢博琳，基于FACE的航电系统软件架构设计，信息技术与信息化，2021.

[3] 张起睿，曲卡尔，模块化开放式航空电子系统架构标准研究，航空电子技术，2022.

[4] 未来机载能力环境技术标准(FACE™).3.1版，2020.

[5] 熊智勇，刘青春，面向云服务的直升机航电系统架构研究，航空科学技术，2023,34(9): 80-86.

[6] 石钊铭，刘传波，基于微服务架构的作战筹划应用系统研究，舰船电子工程，2022,42(8): 13.

[7] 钟伟宏. 基于微服务架构的航天应用软件集成平台设计与实现[D]. 北京：中国运载火箭技术研究院，2021.

[8] 吴化尧，邓文俊.面向微服务软件开发方法研究进展[J]. 计算机研究与发展，2020,57(3):525-541.