基于微服务的智慧医院底层架构设计和应用

黄辛夷 周小平 王忠民

（1 江苏省人民医院（南京医科大学第一附属医院）信息处 南京 210029
2 江苏省妇幼保健院 南京 210036）

（1 江苏省人民医院（南京医科大学第一附属医院）信息处 南京 210029
2 南京医科大学生物医学工程与信息学院 南京 210000
3 南京医科大学医学信息学与管理研究所 南京 210000
4 江苏省妇幼保健院 南京 210036）

〔摘要〕 阐述微服务架构概念与工作原理，基于微服务架构建立医院智慧医疗微服务体系门诊叫号系统，解决单体医疗架构开发功能耦合严重、代码臃肿维护困难、上线成本高、业务伸缩性差等问题，为医院智慧医疗服务提供底层架构支撑，为患者提供更加优质便捷的医疗信息服务。

〔关键词〕 微服务；架构设计；可扩展；智慧医院；门诊叫号系统
〔中图分类号〕 R－058 〔文献标识码〕 A 〔DOI〕 10.3969/j.issn.1673－6036.2023.01.011

Design and Application of Smart Hospital Infrastructure Based on Microservice *HUANG Xinyi, ZHOU Xiaoping, 1Department of Information, Jiangsu Province Hospital (The First Affiliated Hospital with Nanjing Medical University), Nanjing 210029, 2Jiangsu Women and Children Health Hospital, Nanjing 210036, China; WANG Zhongmin, 1Department of Information, Jiangsu Province Hospital (The First Affiliated Hospital with Nanjing Medical University), Nanjing 210029, 2School of Biomedical Engineering and Information, Nanjing Medical University, Nanjing 210000, 3Institute of Medical Informatics and Management, Nanjing Medical University, Nanjing 210000, 4Jiangsu Women and Children Health Hospital, Nanjing 210036, China*

〔**Abstract**〕 The concept and working principle of the microservice architecture are expounded. Based on the microservice architecture, the outpatient queue management system of the hospital smart medical microservice system is established to solve such problems as serious coupling of functions in the development of single medical architecture, difficult maintenance of bloated code, high online cost, and poor business scalability, providing the underlying structural support for the smart medical service of the hospital, and providing patients with more high－quality and convenient medical information services.

〔**Keywords**〕 microservice; architecture design; extensible; smart hospital; outpatient queue management system

1 引言

随着智慧医院建设的不断发展，医院在业务层面更加重视通过自助机和各类线上应用等信息化服务手段[1]，为患者提供优质便捷的服务；而且越来越多的医疗信息应用服务和场景已打破原有点对点的数据交换方式，转而通过医疗信息集成平台来实现消息的流转和分发，获取灵活、标准化和可控的互操作能力。日趋复杂的医疗业务需求和新医疗软件模块、组件的引入导致新医疗信息数据流和医疗信息接口的增加，使得医疗信息软件的可靠性、扩展性和可维护性都面临重大挑战，主要包括以下4

〔修回日期〕 2022－07－24
〔作者简介〕 黄辛夷，助理工程师，发表论文1 篇。

个方面。一是潜在影响难测。当医疗软件需求变得复杂时，需要花费很长时间来评估所做的改动对已有应用的潜在影响。二是应用过于臃肿。虽然可以通过组件化把医疗软件应用划分成多个单元，降低每个单元复杂度，但实际由于组件之间的接口相互调用，组件之间的依赖关系非常复杂。三是应用稳定性变差。由于缺少必要的隔离性，任何组件出现问题，例如内存泄漏，都会导致整个应用崩溃。四是开发功能耦合严重。在各种信息化应用服务通过数据集成平台重耦合在一起时，如果服务节点发生故障或者网络发生异常，都可能导致调用方被阻塞等待，资源很可能被耗尽，最终导致系统雪崩，见图 1。

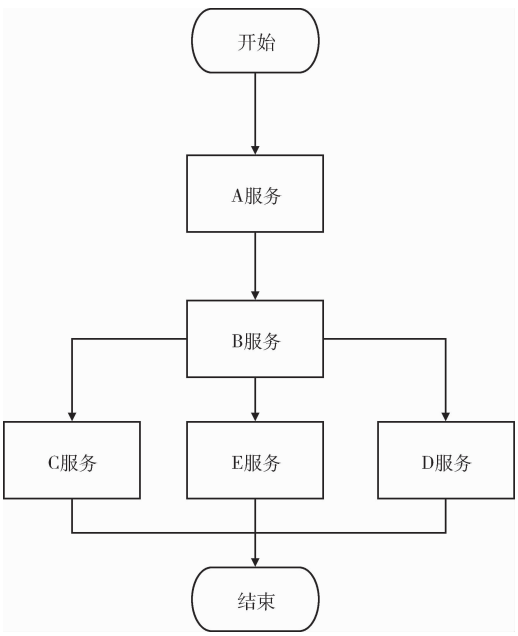

图 1 服务组件调用

一个优秀的架构，必须具有对各种医疗信息服务的提供能力、平稳不间断的运行能力、可行的故障处理能力、牢固的容灾备份能力[2]。当医疗信息应用遇到上述 4 个问题时，说明该应用架构需要调整，需对底层架构重新定义。微服务架构体系就是针对上述问题的解决方案。

2 微服务架构概述

2.1 微服务架构概念与特点

微服务（microservice）是一种架构风格[3]，简单来说就是很小的服务，一个服务只对应一种单一的功能。每个微服务仅关注于一件任务并很好地完成该任务，这个服务可以单独部署运行。即将一个复杂的单体架构应用系统按业务划分为多个独立运行的子系统，每个子系统被称为一个“服务”。应用程序则由一个或多个微服务组成。人体就是一个复杂的微服务架构系统。微服务提供应用程序接口（application programming interface，API）网关、服务注册与发现、负载均衡、服务容错机制、分布式事务处理、链路追踪等一系列微服务组件，这就如同人体的免疫系统。第 1 道防线 API 网关，提供鉴权身份验证、监控服务。当入侵发生时，第 1 道防线会立刻发现并拒绝内部访问。针对某一特定的病原体或异物特异性免疫，就相当于服务注册与发现。能够针对不同的入侵者产生特定的抗体。当再次入侵时，抗体就可以根据之前链路跟踪的结果快速解决。

2.2 微服务工作方式

微服务架构体系在继承面向服务的体系结构松耦合的基础上，将服务作为组件化的单元[4]。在微服务架构中，组件单元变成服务，服务运行在独立的进程中，不通过直接调用来访问，而是用 HTTP 这样的进程间通信方式，每个服务可以独立部署，使用 API 规范来描述其公开接口。微服务架构使用去中心化的管理模式，微服务架构中的服务都可以独立部署，这就意味着每个服务可以选择最适合的技术栈，只需要符合服务要求的标准 API 接口即可。微服务架构开发和维护变得简单，应用系统扩展性好，服务相对独立，业务清晰、代码量较少，比单体应用启动速度快，可以根据需求变更而快速迭代开发。

3 基于微服务架构的底层设计

3.1 医院叫号服务原设计

以医院门诊叫号服务应用为例[5]，有患者报到模块、医生叫号模块、语音消息提醒模块、消息显示提醒模块、微信消息提醒模块，这些都是门诊叫号服务的核心业务，这些模块又相互关联、相互调

用，见图 2。

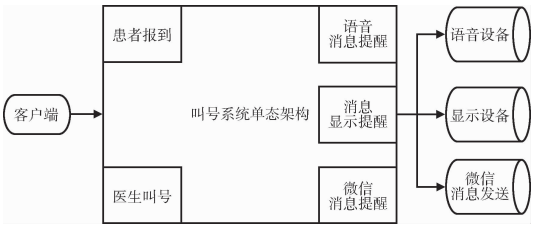

图 2 单态架构

当某个功能模块要升级更新时，整体服务重新部署，这将对核心服务造成很大的困扰，也将对患者、医生、医院的门诊业务影响巨大。

3.2 医院叫号微服务设计

微服务架构专注小的服务个体问题，通过松耦合通信机制协作起来，也就是将单体应用尽量拆分到一个适当的粒度，形成独立的服务个体，见图 3。

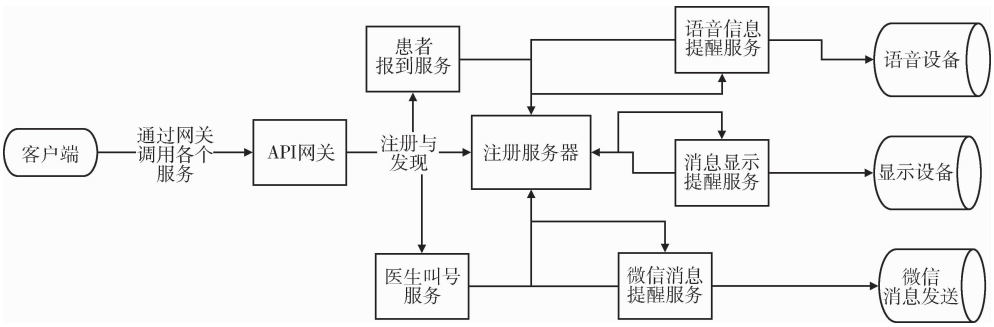

图 3 微服务架构

某个服务不可用不会导致其他服务瘫痪，因为各个服务是相互隔离的系统。微服务架构体系使对外发布服务的部署不会干预到其他核心服务，整体因为部署发布而造成的影响降到最低，增强系统整体的鲁棒性。

4 微服务功能模块

4.1 监控链路跟踪模块

在互联网分布式的多数场景下，巨量请求是暴发故障的诱因[6]。因此建设完善的链路跟踪体系，有助于早发现故障征兆，早解决故障。例如网络资源调用（RESTful Web）会牵涉其他 Web 服务，而其他 Web 服务再次调用更远的 Web 服务，整个 Web 服务请求形成一个网状结构。在这个网状结构中，如果某个 Web 服务发生故障，整个 Web 服务网的稳定性就不能得到保障。为解决此问题，必须及时记录调用请求函数、参数、堆栈区等，并记录 Web 服务上下方的 HTTP 调用参数及其关系，即链路跟踪。

4.2 API 网关模块

API 网关是一个提供服务接口的入口平台[7]。API 网关封装了系统内部架构，为每个客户端提供一个定制的 API。具有身份验证、监控、负载均衡、缓存、请求分片与管理、静态响应处理等功能。API 网关方式的核心要点是，所有的业务请求调用都通过统一的网关接入微服务。使用网关的一个问题就是要决定在多大粒度上使用：最粗粒度的方案是整个微服务一个网关，微服务外部通过网关访问微服务，微服务内部则直接调用；最细粒度则是所有调用，不管是微服务内部调用或者是来自外部的调用，都必须通过网关。

4.3 服务发现模块

服务发现模块用降低故障发生的可能性来降低故障产生的负面影响。多数服务为解决单例服务故障发生后服务无法使用的问题而采用负载均衡的方式部署多个实例，这可以分担服务请求压力并且提高服务请求性能，实现在发生故障后由其他服务实例提供服务。根据服务功能、时间段的不同，需要

不同数量的实例。应用服务将自动注册到服务发现模块上，并定时同步其他应用服务的地址列表到服务发现模块。服务发现模块定时确认应用服务的状况，去除掉线实例地址。服务发现模块根据服务实例状态自行增减服务实例。

4.4 服务容错模块——熔断、降级、限流

4.4.1 服务熔断 当下方真实业务服务因各种故障造成停止响应时，上方服务采用等待策略。如果等待时间很长又有大量请求进入，将使请求堆内存大量分配，大量内存资源一直在等待下方服务响应并释放堆内存。因此当调用下方服务失败时，采用熔断策略，记录下方服务状态，返回错误。轮询下方业务服务，确认正常后再次调用。

4.4.2 服务降级 当非核心业务下方服务停止工作后，则该服务降级，防止发生服务雪崩。例如，在叫号系统中医生叫号模块向微信提醒模块发送调用的请求，当微信提醒模块出现故障后，医生叫号模块不会一起挂掉，只需要调用准备好的降级服务。

4.4.3 服务限流 当服务故障后，通常的解决方式是重试访问。而服务被恢复后，被等候的网络流量冲击立刻失去响应。因此服务需要能够自我保护——限流。限流策略是当单位时间内请求数过多时，考虑排队限流。先接受部分服务的请求，剩余请求陆续发起减少瞬间网络压力。例如患者报到服务和医生叫号服务都需要访问下方真实业务服务，下方真实业务服务由于上方患者报到服务和医生叫号服务同时发起请求，为保证医生叫号服务的请求、患者报到服务的请求正常响应，采用患者正常报到、医生叫号服务的限流模式，在空闲时段响应请求。

4.5 分布式事务模块

分布式事务模块，致力于提供高性能和简单易用的分布式事务服务，为用户提供无侵入分布式事务模式（AT）、高性能分布式事务模式（TCC）、长事务分布式事务模式（Saga）和分布式强一致性事务模式（XA）。主要功能分两阶段处理，一阶段、二阶段提交和回滚，用户只需编写“业务 SQL”，便能轻松接入分布式事务，是一种对业务无任何侵入的分布式事务。一阶段业务数据和回滚日志记录在同一个本地事务中提交，释放本地锁和连接资源。二阶段提交异步化，当事务执行失败，回滚通过一阶段的回滚日志进行反向补偿，最终保证分布式数据一致性。

5 微服务架构的范例

5.1 Spring Cloud alibaba 架构（图4）

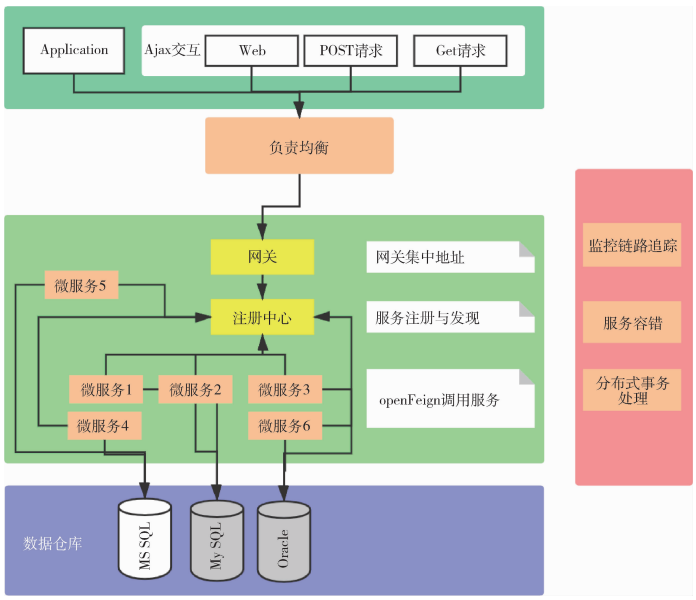

图4 Spring Cloud alibaba 架构

Spring Cloud alibaba 是微服务架构之一[7]。在设计过程中，首先需要创建注册服务器 Nacos，其次构建对应的服务程序逻辑，在 Java 开发框架 Spring Boot 基础上完成程序编写。此架构利用模块以及集成这些模块的基础架构，将分布式系统基础设施开发过程简化，更加方便部署。

5.2 Nacos 服务注册中心

使用 Nacos 在微服务架构中完成服务注册需要两个步骤。

第 1 步，搭建 Nacos 服务端，在项目对象模型（MAVEN）服务项目声明使用 Nacos 依赖。

```
<dependency>
<groupId>com.alibaba.cloud</groupId>
<artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
</dependency>
```

第 2 步，创建新的 Spring Boot 项目，要求在其项目入口文件的公共类上添加 @EnableDiscoveryClient 注解。将当前项目注册进 Nacos 注册服务器。

5.3 Spring Cloud openFeign 服务通信

基于 Spring Cloud alibaba 的微服务架构提供两种方法：一种是 RestTemplate 加 ribbon，另一种是 openFeign。设计开发中，使用 Spring Cloud openFeign 作为服务间调用。openFeign 的优势在于其实现调用的代码量很少，且在 openFeign 内核使用 ribbon 进行负载均衡处理，在进行服务间调用过程中，实现调用的负载均衡。使用 openFeign 时需要创建新接口，在启动类上添加 @EnableFeignClients（basePackages = “com.XX.XX.Service”）注解，在调用类添加@FeignClient（name = “XX”，path = “/XX”）必须声明 name 参数，表明调用服务在 Nacos 注册中心声明的服务名。此外，在调试过程直接调用服务。

```
openFeign 的声明调用
@FeignClient（name = “XXX”，path = “/XXX”）
public interface pushPageMsgService {
@PostMapping（value = “/pushTemplateMsg”）
public int pushTemplateMsg（@RequestBody PushResultReport result）;
}
在具体类中的调用
@ResponseBody
@RequestMapping（value = “/patient/register/”，method = RequestMethod.POST）
public ResultEntity registerOther（@RequestParam（“cardno”）String cardno，@RequestParam（“staffid”）String staffid，
@RequestParam（“cardtype”）String cardtype，@RequestParam（“officeid”）String officeId，
@RequestParam（“officeChildId”）String officeChildId，@RequestParam（“machine”）String machine，
@RequestParam（“staffType”）String staffType，HttpServletRequest request，HttpServletResponse response）{
PushResultReport pushRPT = new PushResultReport（）;
pushRPT.setRetCode（retCode）;
pushRPT.setRetInfo（objs）;
pushRPT.setSoundFlag（soundFlag）;
pushMsgService.pushTemplateMsg（pushRPT）;
rsEntity.setRetCode（retCode）;
rsEntity.setRetInfo（list）;
return rsEntity;
}
```

5.4 Spring Boot 服务创建

Spring Boot 用于简化应用程序的创建和开发过程，化繁为简，简化框架的配置。

```
@SpringBootApplication（scanBasePackages = “com.jsph.jzekCallSign”）
@MapperScan（“com.jsph.jzekCallSign.mapper”）
@EnableCaching
@EnableDiscoveryClient
public class jzekCallSignApplication {
public static void main（String[] args）{
SpringApplication.run（jzekCallSignApplication.class，args）;
}
}
@RestController
@RequestMapping（“/SrmEkAndJz”）
public class SrmEkAndJzController
{
@RequestMapping（“/SrmEkAndJz”）
public String SrmEkAndJz（）
{
XXXXXXXXXXX.....
.....
return “SrmEkAndJz”;
```

```
}
}
```

短短几行代码就可以建立一个简单的急诊儿科叫号服务。内嵌了如 Tomcat、Jetty 架构，所有的依赖都打到一个 jar 包里面，可以直接 java - jar 运行，部署和更新不会影响其他服务。

6 应用实践成果与反思

基于此方案，本院成功地将叫号系统从单态架构体系迁移到微服务架构[8]。在迁移过程中，对特定的业务功能保持关注，做到业务清晰，因此代码量较少，开发和维护相对简单。叫号应用是由若干个微服务构建而成的。整个应用维持在易用可控状态，实现细粒度的扩展。并与其他系统通过轻量级通信机制进行交互。原来架构应用只要有修改，就得重新部署整个应用，现在对某个业务服务进行修改，只需要重新部署这个服务即可。

在微服务设计中容易存在一些不足。一是微服务解耦后，会将一些业务程序从原来数据库查询的方式，转变成远程对象调用，这会增加系统的复杂度和需要约定的接口，带来比 SQL 查询更大的工作量。二是容易将微服务切得太细，假如一个单体被切分成一千个微服务，而且微服务之间用到了远程对象序列化和反序列化，那么，一旦一个微服务的实体对象进行了调整，许多个关联的微服务都会被污染，就要不断定位其他微服务的依赖关系并重新发布。因此，微服务的划分一定要注意，其划分的程度要根据业务不同而粒度不同；而且远程过程调用之间的对象传递尽量采用简单、松散的结构。

7 结语

随着医院集成平台重要性的提升，其底层业务的技术架构也逐渐得到重视。采用微服务架构，能提升底层业务的稳定性、扩展性，院方能从技术环节充分掌控各个业务服务，进而为医院持续推进互联互通、智慧医院建设打下稳固的信息化底层架构体系基础。

参考文献

1 李晓明，黄慧，应毅，等. 基于微服务架构的智能医疗平台设计与开发［J］. 信息与电脑，2019，31（24）：56 - 58.

2 曹斌，王奕. 微服务架构下 HIS 系统设计与实践［J］. 微型电脑应用，2021，37（3）：100 - 102.

3 王桂雁，贺松，俞思伟. 数字化转型下的区域医疗健康信息平台技术架构研究［J］. 中国数字医学，2021，16（5）：1 - 6.

4 谢得麟，严跃宁，林亚忠. 基于微服务架构的医院统一接入平台设计与应用［J］. 中国医疗设备，2021，36（2）：89 - 91，95.

5 黄丽萍，吴忠华，钱小聪. 基于微服务的智慧健康养老综合数据管理平台的研究［J］. 山西建筑，2021，47（1）：197 - 198.

6 郑文靖，王婷. 微服务架构研究方法［J］. 现代信息科技，2019，3（15）：72 - 73，77.

7 杨晓珂，张伟，杨博文. 基于物联网的长大公路隧道消防管网监测平台设计与实现［J］. 山西建筑，2019，45（7）：257 - 258.

8 唐元涛. 基于微服务架构的医院号源池管理设计与实现［J］. 信息与电脑（理论版），2021，33（9）：156 - 158.