基于微服务架构的气象服务支持系统设计与实现

沈晨笛 兰海波* 郭杰 宋瑛瑛 王然 赵大鹏 吕飞 常占来

（中国气象局公共气象服务中心，北京 100081）

摘要 采用传统的以中心控制和大集成模式单体架构为主要形式研发的数据服务系统，因其新增功能困难、版本更新周期长而导致服务体验不佳。本文介绍了基于微服务架构的气象服务支撑系统的建设工作，其主要特点是在研发过程中把系统的各功能逐一细分、原子化并逐一进行服务封装，同时根据不同的服务特点，对不同的业务板块采用不同的成熟技术，并最终进行有机组装，从而使系统在完美实现各种设计目标的前提下，进一步具备了敏捷开发、功能持续更新发布以及服务热更新等特点，有效满足了气象服务业务对数据支撑系统的性能要求和服务功能持续更新要求。

关键词 气象数据；微服务；软件架构

中图分类号： TP31、P409 **DOI：** 10.19517/j.1671-6345.20210545 **文献标识码：** A

引言

目前的气象服务主要以信息服务方式为主，因此数据的稳定供给能力便成为气象服务所有环节中的最重要一环。在气象服务的业务过程中，为应对层出不穷的新的服务方式、服务内容，以及各种需求各异、功能各异的气象服务业务系统，数据平台应运而生，并从一开始便成为气象服务业务的核心。

传统的气象服务数据平台的构建，多采用中心控制和大集成模式的单体架构[1]形式，虽然需求明确、功能全面，但在面对用户定制数据需求、数据本身格式与内容上的不断变化、数据体积不断大幅增长时，这种模式的服务体验不佳，主要表现在大集成模式的单体架构所带来的对二次开发的紧耦合束缚、牵一发而动全身，导致新的功能增加缓慢，版本更新周期长等难点，无法及时满足用户新的服务需求。

随着当前互联网技术的高速发展与行业深度融合、云平台和云原生架构的稳定，敏捷方法、持续交付、精益工程等软件工程方法论的快速发展，以及 DevOps[2-4] 深入人心等，新型的开发、运维、服务方式已经应运而生，即基于微服务的敏捷开发模式[3,5]。

微服务自 2014 年首次提出后，很多大型互联网公司都已开始实践并运用微服务来完成公司的主要业务逻辑的重定义，如 Google、Amazon、Twitter 以及国内的阿里巴巴、百度、腾讯等[6]。

本文借鉴 Amazon 的开发模式，借助模块化的设计思想，研发了一套全国气象数据服务支撑系统，通过采用微服务的技术框架，按照微服务的技术要求，将系统所有功能逐一细分并原子化[7]（即：每个功能完成且只完成一项独立的操作），按照需求对“原子”功能进行服务封装，并为相应的服务都配置了接口，这样开发出来的系统具有良好的增/删/改功能、灵活的功能组装和强大的功能可复用性，使得整个系统拥有了良好的可持续开发的架构，从而将系统未来的二次开发从传统单体式研发模式的束缚中解脱出来，使其能够依托数据平台快速添加新的功能以及调整原有功能，及时满足新的业务需求。同时，在开发过程中，根据不同业务板块的业务特点、进度安排以及开发团队的技术背景，采用了不同的微服务技术方案，最终通过接口将所有功能块及业务板块拼接成一个有机整体，共同实现系统的总体功能需求，从而快速、完整、准确地实现了系统的业务目标。

作者简介：沈晨笛，男，1994 年出生，硕士，工程师，从事公共气象服务信息技术支撑工作，Email：shencd@12379.cn
收稿日期：2021 年 12 月 16 日；定稿日期：2023 年 2 月 2 日
* 通信作者，Email：lanhb@12379.cn

1 面临的问题与目标

1.1 面临的问题

中国气象局公共气象服务中心(以下简称“公服中心”)自成立以来,便开展了以FTP传输文件为主要手段的针对各项业务的数据服务,并以单体式研发模式开发出了相应的数据服务应用系统。随着工作的持续展开,对于提供数据服务的单位而言,以下问题逐渐显露出来:①对于新的数据产品,因没有相应的解析功能,无法及时向用户提供新数据产品的要素级数据服务。②在峰值时段,对接口并发传输出现瓶颈时无法快速扩容,使得系统对外服务在性能上无法满足不断增长的需求。③无法快速添加或更新新的服务方式和能力。④陈旧的定时任务式文件数据传输,无效等待时间长,环节多,数据传输时效低,稳定性不高。⑤数据监控和故障处置缺乏有效的状态数据支持,数据安全要求难以满足。为彻底解决上述问题,公服中心决定建设气象数据服务支撑系统。

1.2 定位与目标

根据需求分析,公服中心确定,气象数据服务支撑系统(以下简称“数据平台”)应能为用户提供方便、快捷、稳定的数据API接口,满足企业级业务场景需求和性能要求的接口服务,并提供平台级的服务统计、态势感知等精细服务。通过数据服务效果监控与统计,增加个性化服务,提高用户的粘性。利用公有云的ECS、RDS、SLB等资源及组件实现水平扩展,为企业级和内部用户提供要素级、文件级的气象数据与产品接口服务。

面对外界不断增加的数据服务需求,数据平台的功能需要快速迭代、升级与更新,而经典的软件工程方法难以适应对外界需求的敏捷响应。因此,使用一套全新的架构进行数据平台系统开发便成为必然的选择。微服务架构提供了一套全新的系统开发架构模式,并凭借其诸多优势,在系统开发领域逐步形成趋势,并成为了本项目的首选技术。

2 架构与解决方案

2.1 架构体系

数据平台在设计之初,考虑到敏感数据的安全性要求,以及气象信息网络系统的独特性,将系统分为内网和外网两部分[8-9](图1)。系统整体逻辑架构分为以下几层:①资源层:内网服务依赖中国气象局园区内物理机房中的虚拟机、物理机、分布式存储系统和网络等资源,外网服务依赖云主机、云数据库、共享网盘等软硬件提供底层支撑。②数据层:主要搭建数据存储,内网部分全面开展数据治理活动,建立数据全生命周期管控;外网部分则开发数据采集流程,通过鉴权的公有云服务器采集来自数据服务支撑平台的数据资源,进而完成后续的数据处理、入库、服务。③服务层:参考现有的行业内的相关数据标准,建立以用户为中心的集中、统一的数据访问接口,提供站点产品的检索、统计;格点产品的压缩、解析、合成、裁剪和入库;文件产品的查询、下载等功能,实现各类服务统一出口、统一管理。④应用层:内网数据应用面向内部单位用户和内部加工系统;外部数据应用面向商业用户、科研合作用户的业务系统。⑤数据监控层:实现数据全生命周期的监控,实现用户服务和数据服务统计分析,实现订单统计,为系统的运维和系统服务策略的优化改进提供数据支撑。

2.2 解决方案

传统的单体架构开发模式一般将团队分为UI、后端、数据库这3个团队。3个团队分别实现各自业务逻辑。这样的分组符合系统的组织结构,但有个致命缺点:实现或调整一个常见的功能,通常也要一并开发并修改UI、后台和数据库,因为这3个层级中,UI要为客户端呈现新的功能,后台要实现对应的逻辑,数据库必须创建用户存储相应数据的接口。这导致了如果有人员要实现某个新功能,就必须和3个团队(亦即开发团队整体)都进行沟通。这种模式牵扯面广、工作量大、耗时甚多,不利于系统版本的快速更新[10-11]。

而如果将整个系统切分成若干个领域组件[10](domain component),每个组件实现有限而特定的功能,组件之间彼此独立(组件间接口规范事先确定),组件内部在技术上则可分为UI(User Interface,用户界面)、后端及数据库,然后在这些领域组件的基础上组建开发团队,各团队负责开发各自的组件。这样,因为功能由组件来具体实现,而组件的规模远远小于系统整体规模,所以系统功能的新增、修改和调整所花费的人力、时间等成本都远远小于传统的单体架构开发模式。这正是微服务的核心要义。

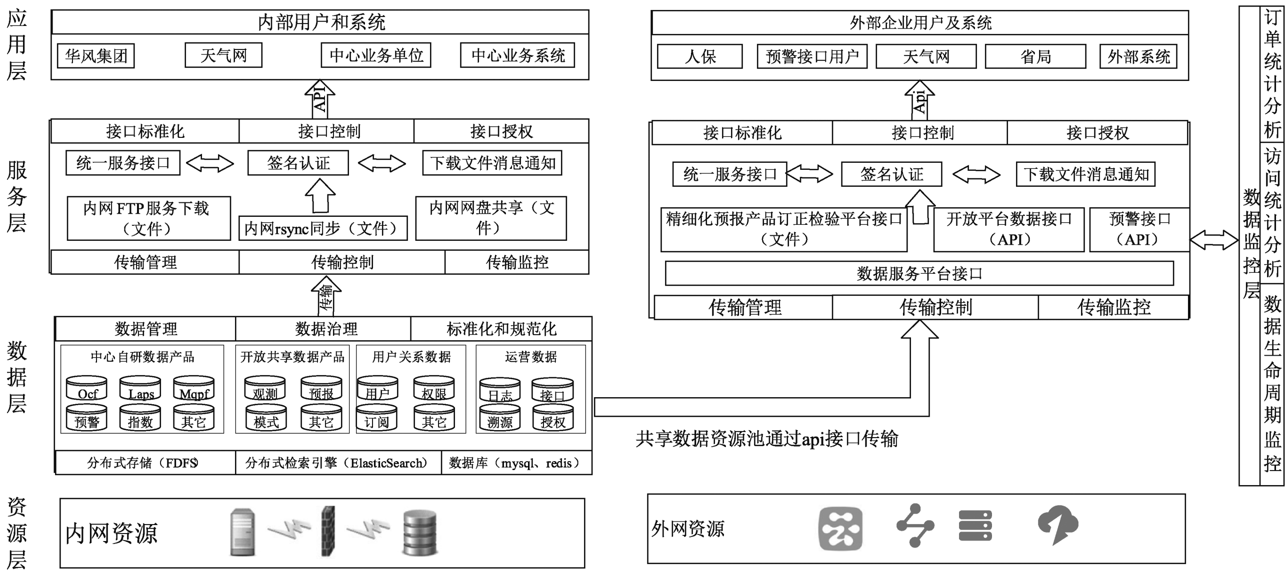

图 1　系统架构

微服务是一种新的模块化方案[12]，与单体架构不同，构成微服务的模块是以独立进程的形式可多点部署运行的。这种方式源于 UNIX 思想，该思想主要概括为以下 3 点：①1 个程序应该仅完成 1 个任务，且应该完美地完成该任务；②程序之间应该能够协同工作；③程序之间应该提供一个通用的接口。基于该思想，微服务拥有模块化强、可替代性强、可持续开发、遗留应用可进一步开发、上市时间较短、具有独立伸缩性、可自由选择技术、能持续交付等特点；可以达到边开发、边应用，可随时扩展增加功能的效果，能快速响应用户的需求[13-16]（图 2）。

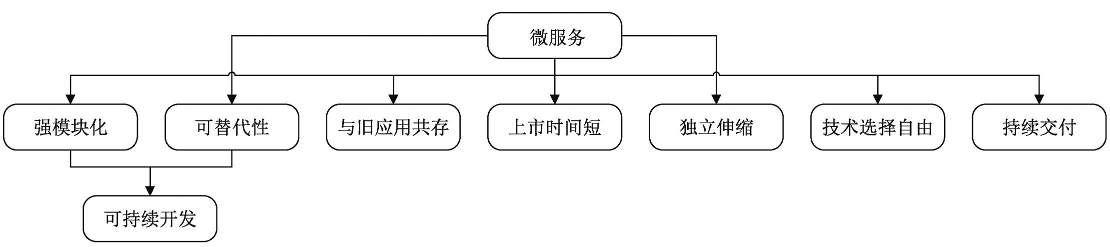

图 2　微服务优势

在设计之初，每个微服务设计成只实现一个功能，并可以交由不同团队开发。从需求工程到生产环境中的运行，这个微服务由该团队完全负责。微服务之间、微服务与单体之间均通过轻量级的通信机制相互协调调用[17-18]。前端界面也根据应用场景划分到不同的微服务中，各个微服务提供各自功能的 HTML 页面。微服务的 HTML 页面允许相互链接进行集成。微服务模块能相互独立部署，因此无需与其他微服务或团队进行协调，就能完成一个微服务的改动和部署。这样大大节省了技术人员沟通成本，方便了功能的并行开发[7]。

目前对于微服务架构并没有官方（或国际标准）的权威定义。但一般人们普遍认为微服务架构[19-23]（MicroServices Architectures，MSA）是将软件系统拆分成多个独立且可扩展的服务，使用一套小服务来开发单个应用，每个服务运行在自己的进程中，只负责完成特定的任务，并使用轻量级机制通信（通常是 HTTP API），最后集成每个服务，装配形成最终的业务系统。这个过程称为微服务（Microservices）化[24-26]。微服务间的接口采用中立的方式定义，独立于具体实现服务的操作系统、软件框架、编程语言和运行基础设施平台，可使用统一、标准、轻量级的方式进行通信，从而实现服务间的松耦合。

由于微服务自身独有的优势,它可同时支持敏捷和自动化的设计、开发、测试、封装、部署、运行、监控和管理等。

2.3 微服务组件开发中的关键技术

在系统建设初期,为能使系统的部分功能快速上线,以提供数据服务,初步设计时将系统分为“数据处理”和“数据服务”两大模块,分别由两组团队进行技术选型与开发。数据处理模块主要负责将数据拆分并由内网推送到外网,数据服务模块负责将数据分库分表定向存储,并支持接口要素页面配置。两个模块都基于 Spring Cloud 微服务的架构,采用页面交互的操作方式完成。

(1)数据处理模块。数据处理模块在微服务的服务发现上,选用了 ZooKeeper,这是因为 ZooKeeper 有着更加良好的界面,更利于技术人员的标准化开发,同时它的分布式服务能力稳定,对于大数据协调服务有着更好的支持。此外,该模块中内外网之间的数据传输采用消息队列(Message Queue)模式,应用 RabbitMQ 予以实现。

(2)数据服务模块。数据服务模块在微服务的服务发现上选用了 Nacos,它对 Spring Cloud 的接口框架有着更好的支持,同时还提供网络端的文件配置功能。这有助于在后期需要对服务进行快速扩容时,配置的同步以及扩容操作的简化。系统内所有微服务之间的参数传递选用 Busflow 予以实现。

2.4 软件开发过程

在将系统功能划分为“数据处理”、“数据服务”两大模块的基础上,对系统整体功能进行了微服务化拆分,除少部分可以复用的功能以外,其余大部分都作为微服务进行单独的功能开发和部署。

数据处理部分的微服务功能包括:内网数据拆分、数据消息发送、外网数据合并、数据入库、数据下载接口,系统在 2021 年 2 月上线提供服务。新功能上线不需要系统重启,实现了服务功能的“热更新”,达到了技术选型最初的“极短时间上市”和“功能持续开发更新发布”的设计要求。

3 系统的使用与运行效果

3.1 系统的运行维护

为了能更简单地使用各类微服务,提升运维效率,目前系统的微服务建立在一套网站 UI 上,所有的功能组件在该网站上都有可视化展现。网站整体分为首页、调度监控、TASK 管理、JOB 管理、调度日志 5 个模块,使得微服务的全流程的监控和管理变得简单。

通过可视化展现,维护人员可及时了解系统各微服务组件的运行状态,并针对出现的新问题及时进行业务调度、流程调度和功能调度,及时升级、更新微服务功能模块,使得数据平台一直处于稳定的运行状态。

3.2 系统的运行效果

表 1 是在实际测试中的结果,测试得出数据接口并发量单机能支持 1000 并发,支持横向扩展。下载接口受限于带宽。测试期间,系统运行基本稳定。目前压力测试最大并发访问量为(QPS):1000,数据接口最低延迟为:14 ms,单个文件(2 GB 左右大小)读取时间耗时 2 min 以内。比原来使用 FTP 定时传输方式节省了时间,稳定性也大有提升。经过内部评估,新系统在高频大数据量文件的传输效率提高了 80%以上,数据传输积压情况明显消失。以某商业用户的数据使用为例,如果使用 FTP 接口方式,以当前的环境和能力,其数据传输效率完全不足以支撑起相关业务应用。而现在利用新建的数据平台,以同样的设备环境,可向该商业用户提供稳定的满足其时效要求的数据服务,并可支撑用户系统平台的实时数据更新与应用。

测试结果表明,数据平台的技术架构能够完全满足目前及可预见未来公服中心的数据服务支撑要求,而采用微服务架构开发数据平台(敏捷开发模式),其软件质量能够达到系统设计之初的所有功能性和非功能性(含性能指标)需求。

在性能指标充分满足设计要求的前提下,由于选用了微服务软件架构,系统的应用软件建设和维护同时实现了:①极短的上市时间。在新建平台设计之初,平台便收到了新的数据服务请求。当时已有的数据服务方式已无法满足新用户的需求。为满足新增需求,在平台尚未完成界面开发的情况下,我们借助微服务的技术优势,提前完成了内外网数据推送、数据入库、索引建立以及数据接口的定制开发,为快速开展“气象+行业”的数据融合应用提供了支撑。②可持续性的功能开发和发布。微服务架构的确立,使数据平台具备了可灵活便捷修改、添加、调整系统应用功能的能力。在最初简单服务功能提供服务后,后续功能不断开发完成并提供服务,

表 1　系统并发量表

数据名称	并发数	循环次数	压测时长/s	平均/ms	最短用时/ms	最长用时/ms	吞吐量/(Mb/s)
城市站点实况天气服务产品(逐小时)	50	1	60	114	44	499	0.848
	100	1	60	73	33	417	1.7
	1000	1	60	80	30	20995	16.7
高分辨率精细化要素预报气象服务产品(小时级)	50	1	10	27	18	67	5.1
	100	1	10	26	19	61	10.1
	200	1	10	23	18	58	20.1
高分辨率精细化要素预报气象服务产品(天级)	50	1	60	32	22	93	1.85
	100	1	60	27	20	83	1.7
	200	1	60	46	18	1047	3.3
城市站点要素预报气象服务产品(小时级)	50	1	60	80	56	353	0.849
	100	1	60	73	46	258	1.7
	1000	1	60	73	41	360	16.7
城市站点要素预报气象服务产品(天级)	50	1	60	115	50	448	0.85
	100	1	60	127	40	640	1.7
	1000	1	60	58	33	425	16.7
道路基本要素产品 1 h 产品	50	1	60	23	18	59	0.85
	100	1	60	25	17	66	1.7
	1000	1	60	22	15	151	16.7
道路基本要素产品 20 min 产品	50	1	60	27	18	67	0.85
	100	1	60	29	18	79	1.7
	1000	1	60	42	14	21007	16.7
道路危险度预报 1 h 产品	50	1	60	27	18	89	0.85
	100	1	60	31	19	89	1.7
	1000	1	60	23	15	108	16.7
道路危险度预报 20 min 产品	50	1	60	25	18	58	0.85
	100	1	60	27	16	76	1.7
	1000	1	60	21	15	124	16.7
路面状态预报 1 h 产品	50	1	60	26	18	61	0.85
	100	1	60	30	18	131	1.7
	1000	1	60	20	15	109	16.7
路面状态预报 20 min 产品	50	1	60	25	19	57	0.85
	100	1	60	24	17	77	1.7
	1000	1	60	21	15	132	16.7

使数据平台在不间断服务的前提下,服务功能不断丰富完善,到目前为止,共新增服务功能 40 余个,平均每个组件用时 40 h,并全部实现“服务热更新”。

③独立的伸缩性。新平台的系统高可用及微服务技术架构,使得系统在某些具体应用处理上出现能力瓶颈时,可有针对性地对这部分基础资源进行扩容,

而不必每次都全系统进行扩容,节约了运维成本。

3.3　存在的问题

随着服务与功能的增加,微服务的一些问题也显现出来:①管理问题,在单体系统中,配置可以直接写在 yaml 文件中,而微服务的分布式系统则需要进行统一的配置管理,且同一个服务在不同的场景下对配置值的要求可能不一样;因此需要后续引入版本配置管理。②可靠性问题,微服务之间的调度高度依赖于网络,任何一次网络波动都有可能引起故障,需要加强环境管理,引入实时微服务调用链异常检测[27]、DeepLog 等方法[28]从而提高系统的可靠性。③部署问题,目前我们没有一套有效的统一自动化部署流程,在微服务架构中,每个服务都需要独立部署,而微服务方式的特点就是交付周期短、频率高,采用人工部署的方式已渐渐无法适应业务的快速更新迭代。④运维问题,目前的运维工作主要包括配置、部署、监控与告警和日志收集四大方面;由于微服务架构中,每个服务都需要独立地配置、部署、监控和收集日志,所以运维成本随微服务数呈指数级增长。

为解决以上问题,目前采取的方案是需再研发一套版本管理与部署平台,以用于进行微服务的版本管理与统一部署,方便运维人员在有新的软件版本迭代时立刻进行统一部署,从而减少服务上的延迟,以及部署上的失误。

4　结论

“微服务”概念虽然提出已有10余年,但引入气象部门却相对滞后。气象服务部门面向政府和社会,面对复杂多变的天气系统,面对不断新增与变化的服务需求,业务系统经常面临服务功能的调整和增加,以及业务性能的提高。不断将系统推倒重来、以旧换新的单体模式耗时费力、事倍功半,难以适应气象服务快速响应的业务特点。相较而言,“微服务”架构因其天然具备的 DevOps 能力以及“服务热更新”特色,在研发速度、成本和效益等方面有着明显的优势。因此,尽管存在一些缺陷和问题,“微服务”架构仍然应当成为气象服务业务领域技术架构中的重点技术选择之一。

对于公服中心的核心数据服务平台来说,本文所述工作仅仅是开始,随着数据的扩容、数据接口的丰富,定制需求的外延,数据格式的转换抽取等,仍有不少可以完善优化的领域,而“微服务”架构能够在功能增加、替代和调整、优化等方面继续展现它无可替代的优势和特色。

参考文献

[1]　Basel B, Youcef B. Guiding the selection of service identification methods based on SOA maturity level [J]. International Journal of Web Information Systems, 2019,15(3):17-25.

[2]　钟陈星,李杉杉,张贺,等.限界上下文视角下的微服务粒度评估[J].软件报, 2019, 30(10): 3227-3241.

[3]　黄璜,张贺,邵栋.自动化工具对中国 DevOps 实践的影响[J].软件学报,2019,30(10):3056-3070.

[4]　Li Z, Zhang Y, Liu Y. Towards a full-stack DevOps environment (platform-as-a-service) for cloud-hosted applications [J]. Tsinghua Science and Technology, 2017,22(1):1-9.

[5]　冯志勇,徐砚伟,薛霄,等. 微服务技术发展的现状与展望[J].计算机研究与发展,2020,57(5):1103-1122.

[6]　田浩. 基于 SOA 的高并发与高可用网站开发框架设计与实现[D]. 呼和浩特:内蒙古大学,2017.

[7]　刘若宇, 刘晓燕. 一种组合优化模型驱动的微服务拆分方法[J/OL]. [2022-11-09]. 小型微型计算机系统:1-8. DOI:10.20009/j.cnki.21-1106/TP.2022-0331.

[8]　辛园园,钮俊,谢志军,等. 微服务体系结构实现框架综述[J].计算机工程与应用,2018,54(19):10-17.

[9]　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.

[10]　Al-Debagy O, Martinek P. A comparative review of microservices and monolithic architectures [C]// 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 2018: 149-154.

[11]　Esposito C, Castiglione A, Choo K R. Challenges in delivering software in the cloud as microservices [J]. IEEE Cloud Computing, 2016, 3 (5): 10-14.

[12]　Vayghan L A, Saied M A, Toeroe M, et al. Microservice based architecture: towards high-availability for stateful applications with kubernetes [C]// IEEE International Conference on Software Quality, Reliability and Security (QRS), Sofia, Bulgaria, 2019: 176-185.

[13]　Kalske M, Ma-Kitalo N, Mikkonen T. Challenges when moving from monolith to microservice architecture [C]// Proc of the Int'l Conf on Web Engineering (ICWE 2017). Current Trends in Web Engineering, 2018: 32-47.

[14]　施凌鹏, 朱征, 周俊松, 等.面向微服务架构的云系统负载均衡机制[J]. 计算机工程, 2021, 47(9): 44-50+58.

[15]　Fowler M, Lewis J. Microservices [EB/OL]. (2021-08-20)[2021-12-16]. https://martinfowler.com.

[16]　Kainz A. Microservices vs monoliths: an operational comparison [EB/OL]. (2020-07-07)[2021-12-16]. https://thenewstack.io/microservices-vs-monoliths-a n-operational-compari-

son.
[17] Fielding R T. Architectural styles and the design of network-based software architectures [D]. Irvine: University of California, 2000.
[18] Lewis J, Fowler M. Microservices: a definition of this new architectural term [EB/OL]. (2014-03-25) [2021-12-16]. http://martinfowler.com/articles/microservices.html, 2014.
[19] 薛皓辰，柳先辉. 基于微服务架构的制造技术资源服务平台架构研究[J]. 科技管理研究，2021,41(14):208-212.
[20] 徐瑛楠，孙景睿. 基于微服务架构的智慧云平台任务分配研究[J]. 计算机仿真,2021(10):482-486.
[21] 江郑，王俊丽，曹芮浩，等. 一种基于微服务架构的服务划分方法[J]. 计算机科学,2021(12):17-23.
[22] 宋丽华,李士宽,张建成,等.基于微服务架构的信息服务云平台设计[J].信息技术与信息化,2022(10):156-160.
[23] 郝江波,唐卫,王慕华,等.基于微服务的气象信息决策支撑系统重构与实践[J].气象科技,2020,48(6):829-835. DOI:10.19517/j.1671-6345.20200140.
[24] James M, Oliver S, Ashish J, et al. Enterprise service oriented architectures: concepts, challenges [J]. Recommendations Springer Science & Business Media, 2006, 54(4): 211-254.
[25] 张攀,高丰,周逸,等.一种在线实时微服务调用链异常检测方法[J].计算机工程,2022,48(11):161-169. DOI:10.19678/j.issn.1000-3428.0063817.
[26] 经成,谢军.基于微服务架构的负载均衡优化算法及实现[J].南昌大学学报(理科版),2022,46(4):399-404+415. DOI:10.13764/j.cnki.ncdl.2022.04.003.
[27] 秦运龙,王迎迎,张冰松,等.省级外网气象大数据服务平台研究与实现[J].气象科技,2020,48(6):823-828+854. DOI:10.19517/j.1671-6345.20190546.
[28] Du M, Li F F, Zheng G H, et al. DeepLog: anomaly detection and diagnosis from system logs through deep learning [C]// Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security. New York, USA, ACM Press, 2017: 1285-1298.

Design and Implementation of Meteorological Service Support System Based on Micro-service Architecture

SHEN Chendi　LAN Haibo　GUO Jie　SONG Yingying　WANG Ran　ZHAO Dapeng　LYU Fei　CHANG Zhanlai

(CMA Public Meteorological Service Center, Beijing 100081)

Abstract: The traditional data service system, which is developed in the form of central control and large integrated single architecture, has poor service experience due to its difficulty in adding new functions and long version update cycles. This paper puts forward the construction idea of the meteorological service support system based on the micro-service architecture, which subdivides and atomizes the system functions and encapsulates the services. At the same time, according to different service characteristics, different mature technologies are used for different operation segments, and finally, the unit installation is carried out. Thus, under the premise of perfect realization of various design goals, the system further has the characteristics of agile development, continuous function update and release, and hot service update. At the same time, the system performance is greatly improved by using distributed deployment, elastic expansion and other technology applications.

Keywords: meteorological data; micro-services; software architecture