计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(3):86−94 [doi: 10.15888/j.cnki.csa.008999]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

气象大数据云平台监控告警系统①

刘　洋[1], 黄　志[2], 徐　娟[1], 唐建新[3], 卢伟萍[4]

[1](甘肃省气象信息与技术装备保障中心, 兰州 730020)
[2](广西壮族自治区气象信息中心, 南宁 530022)
[3](兰州理工大学 计算机与通信学院, 兰州 730050)
[4](广西壮族自治区气象科学研究所, 南宁 530022)
通信作者: 黄　志, E-mail: 616646373@qq.com

摘　要: 气象大数据云平台 (简称“天擎”) 作为省级气象业务的核心系统, 需要保持 7×24 小时不间断的稳定、高效运行. 针对“天擎”系统运行模块多、处理任务多且复杂, 传统的人工监控模式监控效率低且无法及时发现业务中存在的故障等问题, 本文采用 Java、Python 和 Bash shell 语言开发了基于企业微信的“天擎”业务全流程监控告警系统, 该系统通过对“天擎”各个模块业务运行过程中所产生的综合状态信息等进行采集并格式化为监控告警信息, 最终通过企业微信推送至运维人员, 实现了对“天擎”各业务运行模块运行状态的快捷感知. 系统业务运行效果表明, 该系统运行安全可靠稳定, 能够帮助运维人员及时定位系统故障并提高故障处理效率, 在“天擎”数据监控和运行保障方面取得了良好的应用效果.

关键词: 气象大数据; 监控告警; Python; 网络爬虫; 消息队列; 微信

引用格式: 刘洋,黄志,徐娟,唐建新,卢伟萍.气象大数据云平台监控告警系统.计算机系统应用,2023,32(3):86–94. http://www.c-s-a.org.cn/1003-3254/8999.html

Monitoring and Warning System of Meteorological Big Data Cloud Platform

LIU Yang[1], HUANG Zhi[2], XU Juan[1], TANG Jian-Xin[3], LU Wei-Ping[4]

[1](Gansu Meteorological Information and Equipment Technical Support Center, Lanzhou 730020, China)
[2](Guangxi Meteorological Information Center, Nanning 530022, China)
[3](School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China)
[4](Guangxi Institute of Meteorological Science, Nanning 530022, China)

Abstract: As the core system of provincial meteorological service, the meteorological big data cloud platform (referred to as Tianqing) needs to work stably and efficiently for uninterrupted 7×24 hours. The Tianqing system has many operation modules and complex processing tasks. However, the traditional manual monitoring mode has low monitoring efficiency and cannot find faults or other problems existing in the operation in time. In this study, the Java, Python, and Bash shell languages are used to develop a full-process monitoring and warning system for Tianqing based on Enterprise WeChat. The system collects and formats the comprehensive status information generated during the operation of each module of Tianqing into monitoring and warning information and finally sends it to the operation and maintenance personnel through the Enterprise WeChat, which realizes the quick perception of the operation status of each operation module of Tianqing. The operation effect of the system shows that the system is safe, reliable, and stable, and it can help operation and maintenance personnel to locate system faults in time and improve the efficiency of fault handling promptly. In addition, it has achieved positive application effects in Tianqing data monitoring and operation guarantee.

Key words: meteorological big data; monitoring and warning; Python; Web spider; message queue; WeChat

① 基金项目: 国家自然科学基金 (62162040); 广西科技厅重点研发项目 (桂科 AB21196041); 2022 年甘肃省气象信息与技术装备保障中心科技创新基金 (202201)
收稿时间: 2022-08-10; 修改时间: 2022-09-15; 采用时间: 2022-09-27; csa 在线出版时间: 2022-12-09
CNKI 网络首发时间: 2022-12-13

气象行业数据种类繁多, 包括各类结构化、半结构化和非结构化数据, 随着互联网技术、信息化技术以及气象观测技术的迅速发展, 气象数据采集频次从逐天逐时转变为逐分钟, 使气象行业累积了海量数据, 数据量已达到 PB 以上, 具有典型的大数据特征. 传统的计算方式无法对海量数据进行深度挖掘, 在应对海量数据的高并发访问同样遇到了处理瓶颈, 导致在气象防灾减灾领域, 气象数据的价值无法有效的实现. 因此, 气象部门对于应用大数据相关技术以提升气象信息化水平有着迫切的需求, 在此背景下, 气象大数据云平台“天擎”[1,2] 作为气象部门信息化的核心系统应运而生.

“天擎”是一个庞大的、复杂的大型综合性气象业务系统, 它是云计算、大数据、分布式计算、人工智能等各种技术的综合体, “云+端”业务应用云化是“天擎”设计的一个亮点, “天擎”通过提供“数算一体”的业务新形态, 整合气象部门的各类数据资源和处理流程, 消除数据烟囱和信息孤岛, 实现气象业务数据资源的汇聚互联、共建共享, 从而推进气象业务技术体制改革和转型.

2021 年底“天擎”正式投入业务运行, “天擎”包括 CTS 传输、DPC 解码、SOD 存储、MUSIC 接口、PORTAL 等多个集群子系统共 100 多台服务器, 业务环节冗繁且运行流程复杂, 如何保障整个平台全链条全天候的正常运行给运维人员带来了巨大的挑战, 因此, 建立“天擎”的全链条自动化监控告警系统是十分有必要的, 高效的监控告警系统可以及时定位故障并提升故障处理的响应时效保证系统持续稳定运行, 减少运维成本. 目前, 在系统及应用监控方面出现了一些优秀的开源分布式监控软件, 如 Zabbix、Prometheus、Nagios 等[3–5], 可以实现系统资源和应用状态的图形化动态监控, 在资源监控领域使用比较广泛. 但是这些开源的框架部署困难、开发技术难度大、并且都是面向一些通用性的监控任务, 对于特殊的气象业务场景并不适用.

针对上述问题, 本文对基于“天擎”整体业务处理流程的关键环节和技术进行了研究, 实时监视和采集各处理环节的运行状态并转换为告警信息, 在保证网络安全的前提下, 通过企业微信将告警信息实时的发送给系统运维人员, 实现对“天擎”各业务系统运行状态的全方位的监控, 助力运维人员及时定位并处理系统故障.

1 系统总体架构

1.1 系统总体架构

本文系统架构分为 3 层, 包括数据源支持层、数据加工处理层及应用层. 其中数据源支持层以“天擎”的数据环境为基础, “天擎”数据环境包含地面、高空、雷达、数值预报、行业共享数据等 14 大类的数据集和数据产品, 集成了“虚谷分布式关系型数据库、Gbase 分布式列式数据库[6]、Cassandra 分布式表格系统[7]、Ceph 分布式文件系统[8,9]、分布式对象存储[10]”等于一体的适合不同数据格式、满足不同应用场景的大数据存储体系. “天擎”以传统分布式 NAS 结合数据湖技术[11], 对于运行在此基础上的 CTS (传输)、DPC (解码)、SOD (存储)、MUSIC (API 接口)、微服务众创接口等各个子系统运行产生的状态消息、数据入库信息、运行处理日志、API 接口访问日志、管理信息、NAS 存储状态[12] 等信息, 从原始数据接收、分发、质控解码开始, 到最终数据的存储与发布, 都详细地记录每一条数据的产生和动态流向的过程, 完成了对不同类型的数据处理全流程的分析和溯源, 并具备完善的数据生命周期管控能力, 为管理员提供了一体化数据全流程的溯源监控. 此外, 数据湖有着多样化的分析处理能力, 例如基于 STORM 的雷达、地面 Bufr 分钟数据流的解码入库架构, 基于 SPARK 并行分布式微服务众创接口、数据图谱构建与日志分析等, 满足用户对多种应用场景的数据服务需求并能为用户主动推送热点数据. 本系统的告警数据源就是源于上述“天擎”各模块子系统运行过程中所产生的综合状态信息.

数据加工处理层是整个系统的核心模块, 它按照一定的频次, 通过周期性的任务请求, 以 API 接口、数据库直连等方式从数据源支撑层对数据进行检索采集、然后经过数据清洗、格式转换、数据分析计算并完成最终数据推送的全过程. 数据加工处理层根据业务的实际需求, 设计了多个独立的采集处理模块, 每个采集处理模块都是独立开发, 相互之间没有依赖关系, 并且都是并行运行, 这样极大地提高了数据采集效率. 监控数据在采集过程中需要进行数据清洗以尽可能地消除脏数据, 比如网络传输异常或者故障产生的重复数据、不完整数据等, 其数据无法被利用和纠正, 此类数据只能进行清洗、修复和剔除才能使用. 数据计算

主要对是数据表的操作, 比如聚合计算, 通过计算得到具有业务意义的相关指标, 随后将数据格式化, 形成基础告警数据.

应用层主要是对数据的正确性进行验证并完成告警数据的迭代更新, 是通过被动的方式获取加工处理层的基础数据, 即数据加工处理层是将数据“推”到应用层, 而不是应用层从数据加工层“拉”取数据, 应用层被动的接收数据加工层上报的数据. 采用这种数据传输方式符合网络安全的要求, 最后将告警数据发送给微信客户端. 系统总体架构如图 1 所示.

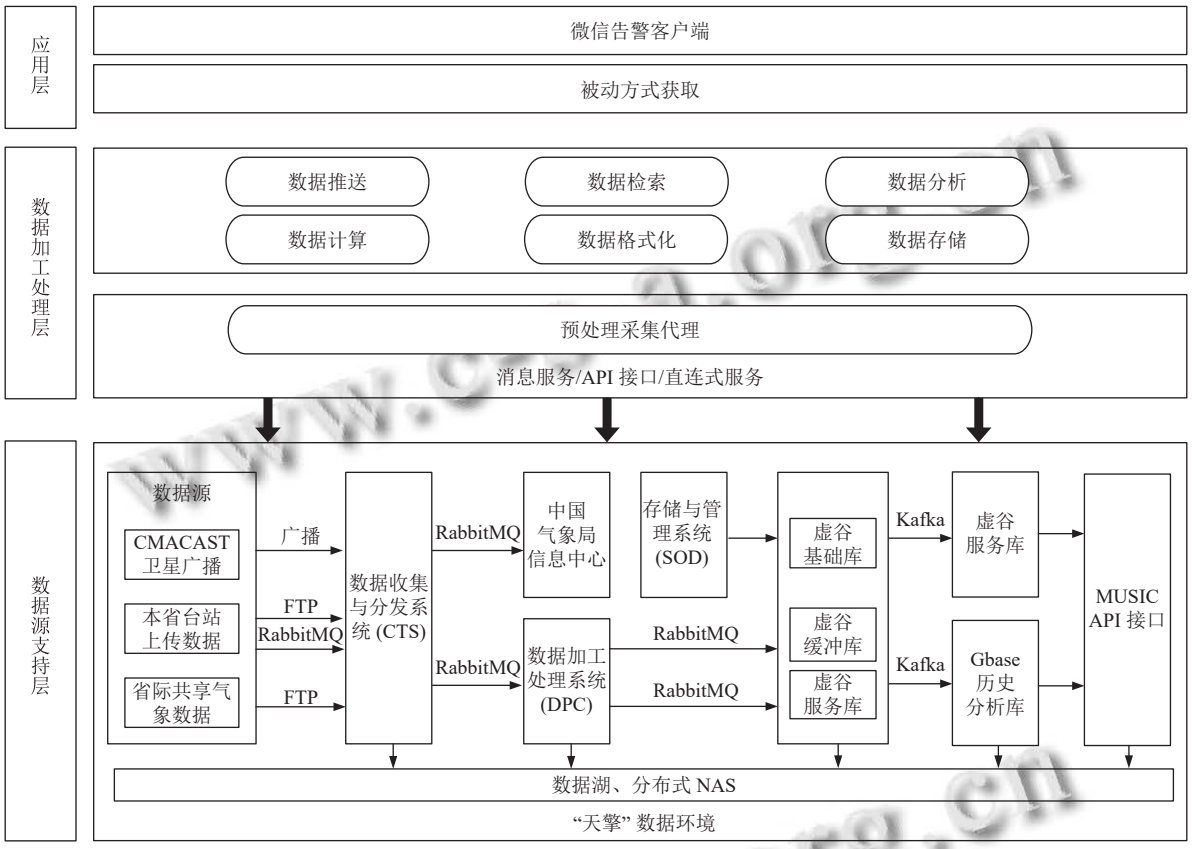

图 1 系统总体架构

1.2 系统硬件结构

系统硬件包含 8 个部分, 分别为万兆交换机、负载均衡设备、数据处理服务器集群、SAN 交换机、磁盘阵列、接口、防火墙和安装了企业微信的终端. 本系统采用分布式负载均衡的架构, 以满足系统长时间不间断密集型的稳定运行、以及大流量、高并发和高可用的应用需求. 硬件都是以集群的方式部署, 这是实现分布式架构和高可用的物理基础[13], 在数据加工处理层与数据源支持层的数据交换是通过代理程序实现的, 代理程序负责数据采集, 随后将采集的监控任务发送给数据加工处理层前端的负载均衡设备, 由负载均衡统一调度分配任务给后台加工服务器, 将加工处理后的信息通过单向传输和感知的方式推送给应用层, 同时为了保证数据的一致性和数据同步带来的延迟, 本系统采用共享 SAN 存储方式, 将每个节点访问的数据存放在同一个共享存储空间[14], 即多个节点共享同一份数据有效的解决不同节点数据同步带来的延迟和数据不一致的现象. 集群中的每个部分相互协作共同完成数据交互, 共同实现本系统的各项功能. 其物理架构如图 2 所示.

2 系统实现关键技术

本系统开发的硬件环境为: (1) Windows 环境 Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz 8 核 CPU, 8 GB 内存, 500 GB SSD; (2) Linux 16 核 CPU, 8 GB 内存, 500 GB SSD. 软件支撑环境: 操作系统采用 CentOS 7.4 和 Windows 10. 开发工具采用 IntelliJ IDEA 和 Vim. IntelliJ IDEA 用于 Java 代码开发, Vim 用于

Python 代码开发. 开发和测试环境中, IntelliJ IDEA 集成开发环境和所在的 Windows 主机需要能够访问互联网, 同时 Vim 开发环境所在的主机也需要能够访问互联网和“天擎”服务接口, 便于系统开发和测试.

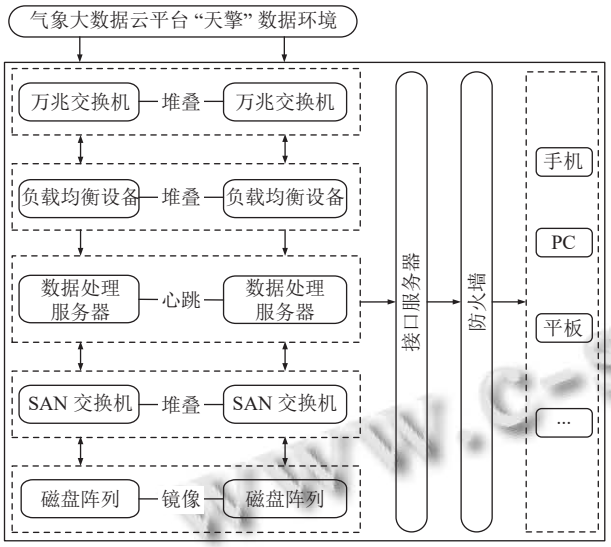

图 2 系统物理架构

2.1 RSA 加密算法

根据系统开发的规范标准, 配置文件和程序中对于出现的敏感字符, 比如数据库的用户名、密码等不能出现明文. 本系统采用 RSA 加密算法[15] 主要是对程序和配置文件中出现的用户名、密码及 URL 进行加密, 避免主机被攻击之后导致安全信息被泄漏. 其加密示例代码如下.

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1_v1_5
import base64
message = "http://localhost:15672/api/queues"
rsakey = RSA.importKey(open("public.pem").read())
cipher = Cipher_pkcs1_v1_5.new(rsakey) cipher_text = base64.b64encode
(cipher.encrypt(message.encode('utf-8')))
print(cipher_text.decode('utf-8'))
```

2.2 Python 多进程

为了能够加快站点数据的处理速度, 本系统以多核 CPU 资源为基础, 利用 Python 提供的多进程 Multiprocessing 包, 完成了从单进程到并行执行的切换. Multiprocessing 能够支持子进程、通信和共享数据、执行不同形式的同步, 同时提供了 Process、Queue、Pipe、Lock 等组件. 本系统在“天擎” Web 服务和 FTP 服务监控中采用了多进程技术, 其示例代码如下所示.

```
if __name__ == "__main__":
    file_exist_res = is_file_exist('url_address.txt')
    if file_exist_res != 0:
        file_num, is_ack = split_file('url_address.txt')
        print("---cpu 的核数为:{}".format(cpu_count()) + '---')
        print("---父进程开始执行---")
        print("父进程 PID: %s" % os.getpid())
        p = Pool(file_num)
        for i in range(0, file_num):
            print('生成的文件名为: '+ 'url_address.00%d'%i + "
记录的日志为: failed%d"%i +'.log')
            p.apply_async(get_result, args=('url_address.00%d'%i,
'failed%d'%i +'.log', ))
        print("---等待所有子进程结束---")
        p.close()
        p.join()
        print("---父进程执行结束----")
```

2.3 Python 调度器

APScheduler (advanced Python scheduler) 是 Python 的一个轻量级定时任务框架, 包含的任务类型有 Linux Crontab 风格的调度、基于时间间隔的周期性执行调度等, 能够以守护进程的方式启动应用, 并且具有任务持久化的功能. APScheduler 包含 4 个组件, 分别是触发器、作业存储、执行器和调度器. 本系统采用 APScheduler 中的 BlockingScheduler 调度器实现间隔执行任务, 主要用于 FTP 服务采集模块, 时间间隔设置为每 10 分钟采集一次.

2.4 Python 网络爬虫

网络爬虫[16] 就是向网络站点不断的发送 HTTP 请求, 通过程序模拟浏览器访问站点的行为, 把站点返回的 JSON 数据、图片、视频等保存到本地, 提取需要的数据进行存储. 流程为发送请求、获取响应内容、解析内容、保存数据. 本系统是通过服务器管理口爬取硬件日志进行分析, 爬虫请求库为 Requests, 解析库为正则和 beautifulsoup. 本系统网络爬虫技术主要用于“天擎”服务器硬件监控. 其示例代码如下所示.

```
sess = requests.session()
sess.cookies = cookielib.LWPCookieJar(filename = "cookies.txt")
userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0)
Gecko/20100101 Firefox/93.0"
header = {
    "Referer": "https://ip/api/session",
    'User-Agent': userAgent,
}
sess.cookies.load()
url = 'https://ip/api/alarm_info'
```

```
result = sess.get(routeUrl, headers = header, allow_redirects = False, verify = False)
print(result.text)
if result.status_code != 200:
    print("Failed")
else:
    print("Success")
```

2.5 Java 线程池

在线程池中[17], 线程是可以复用的, 不需要每次都创建一个新的线程, 有效地减少创建和销毁线程的时间开销. 线程的数量可以限制和手动设置, 这样可以避免由于创建线程过多而耗尽系统资源导致服务器宕机, 同时也把实际的任务与线程本身进行了解耦, 本系统使用 Executors 类提供工厂方法创建不同类型的线程池, ExecutorService executorService = Executors.newFixedThreadPool(6), 通过调用返回值 ExecutorService 接口的基础方法 execute 或 submit, 将具体执行的线程任务加入线程池中并执行. 线程池技术主要用于主机网络连通性监控.

2.6 Python 正则表达式

所有的监控告警信息最终都需要在互联网上进行传输, 由于网络安全的要求, 本系统采用正则表达式[18]对一些告警信息中的敏感字符进行屏蔽和转换, 保证传输的数据不包含敏感信息, 正则表达式可以处理任何关于字符串的操作, 是一个强大的字符串处理工具, 具有自己独特的语法和数据处理引擎. 其示例代码如下所示.

```
ip_port_re = r'(6[0-5]{2}[0-3][0-5]|[1-5]\d{4}|[1-9]\d{1, 3}|[0-9])'
re_result = re.findall(ip_port_re, first_part)
    if len(re_result) == 5:
        ip_address = re_result[0] + '.' + re_result[1] + '.' + re_result[2] + '.' + re_result[3]
        ip_rep = first_part.replace(ip_address, 'ip')
        port_rep = ip_rep.replace(re_result[4], 'port')
    elif len(re_result) == 4:
        ip_address = re_result[0] + '.' + re_result[1] + '.' + re_result[2] + '.' + re_result[3]
        ip_rep = first_part.replace(ip_address, 'ip')
```

3 系统核心功能设计与实现

系统的数据采集处理流程位于数据加工处理层, 数据加工处理层是数据平台建设的核心, 目前包括 11 个独立处理的模块, 可以进行扩展, 这些模块实现了对“天擎”核心服务器运行状况信息的采集和处理存储, 包含 CTS、DPC 等分系统的消息采集解码, SOD 存储、数据同步、清除策略运行任务监控信息的提取以及各类上行数据传输状况信息、后台日志采集等. 采集后的数据使用 MySQL 和文件索引的方式进行存储. 数据加工处理层监控信息的采集与处理流程如图 3 所示, MySQL 数据库表结构如表 1 所示.

3.1 “天擎”消息队列数据采集功能设计

“天擎”消息队列主要包括 Kafka 消息队列[19]、Bufr 数据消息队列、CTS 文件消息队列和 DPC 文件通知消息队列. 消息队列及消费者类型如表 2 所示.

(1) Kafka 数据同步消息队列. 作为“天擎”虚谷和 Gbase 数据库的数据同步消息中间件, “天擎”数据流从 CTS 输入、经过 DPC 进行解码存入虚谷缓冲库, 同时将数据写入 Kafka 消息队列中, 虚谷服务库和 Gbase 历史分析库以消费者身份获取该消息数据, 完成数据从缓冲库到服务库和分析库的同步. 当前需要监控的 Kafka 消息队列 (主题) 有 149 个. 通过 Kafka 集群目录 /bin 下的 kafka-consumer-groups.sh -bootstrap-server HostName:9092 -list 获取每个队列名称, kafka-consumer-groups.sh --bootstrap-server HostName:9092 --describe --group queue name 计算每个队列所有分区的消息数目. 本系统采集 Kafka 消息队列信息的示例代码如下所示.

```
cat /space/cmadaas/sod/shell_ly/new_code/consumer_group.tmp| while read consumer_group
do
    if [ "consumergroup"="music−group−2"]||["consumergroup"="music−group−2"]||["${consumer_group}" = "mygroup" ] || [ "${consumer_group}" = "getendpos" ];then
        continue
    fi
sh /space/cmadaas/sod/kafka_2.11-2.1.1/bin/kafka-consumer-groups.sh --bootstrap-server ${node01}:${port} --describe --group ${consumer_group}| grep '^bclz_'|awk '{print $1, $5, $7}' > /space/cmadaas/sod/shell_ly/new_code/consumer.tmp
    partition_name='cat /space/cmadaas/sod/shell_ly/new_code/consumer.tmp |head -1|awk -F "'{print $1}"
    partition_num='cat /space/cmadaas/sod/shell_ly/new_code/consumer.tmp |wc -l '
    jiya_sum='cat /space/cmadaas/sod/shell_ly/new_code/consumer.tmp |awk '/bclz_/ {sum += $2};END {print sum}"
    echo ${jiya_sum}
    consumer_node='cat /space/cmadaas/sod/shell_ly/new_code/consumer.tmp |head -1|awk -F "'{print $3}"
    fi
done
```

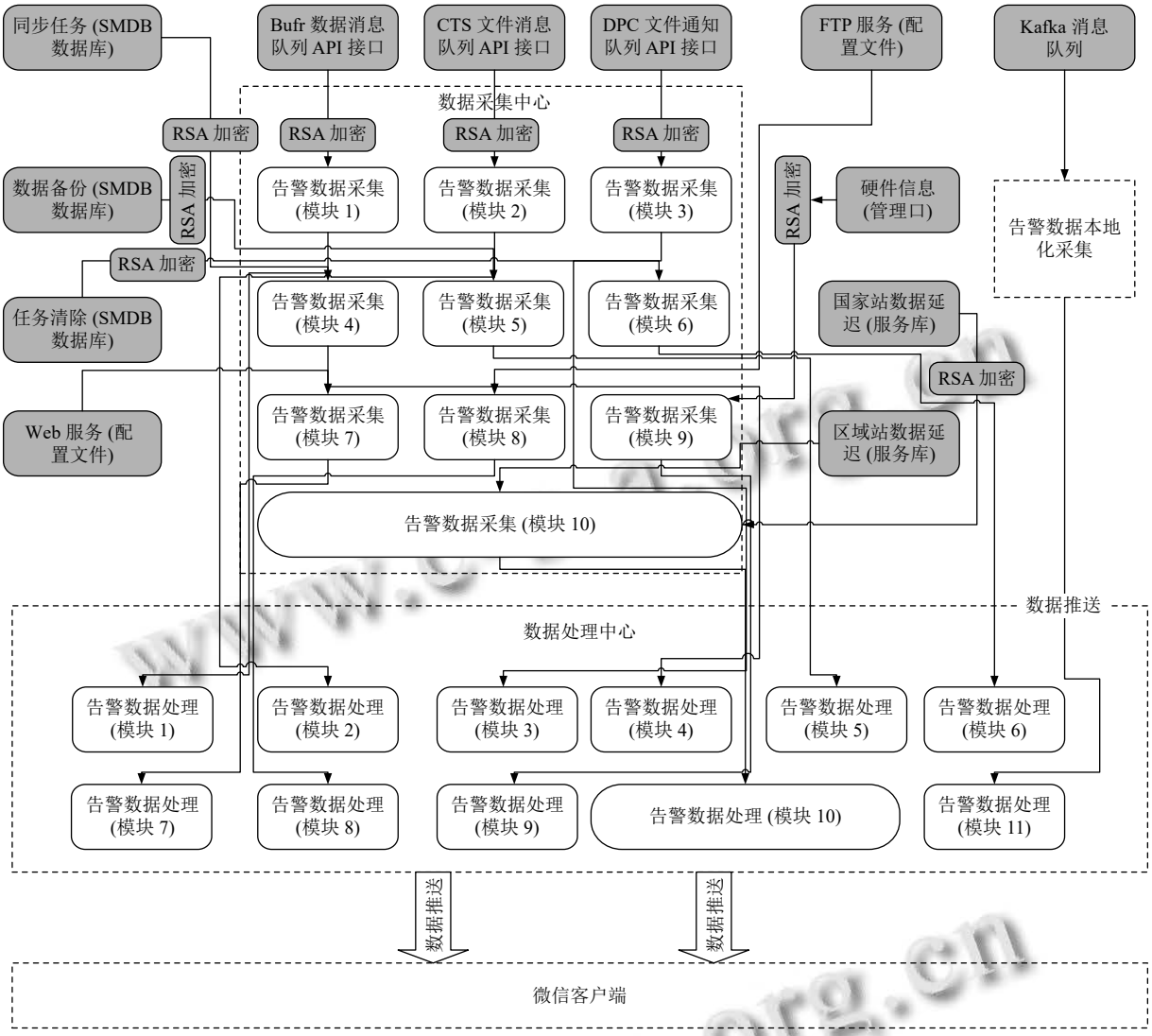

图 3 系统数据采集与处理流程框图

表 1 监控告警信息数据库表结构

字段	字段描述	是否为空	数据类型
InsertTime	入库时间	N	DATETIME
QueueName	队列名称	N	CHAR
RunningNode	运行节点	N	CHAR
RunningStatus	运行状态	N	CHAR
MessageNum	消息积压数	N	VARCHAR
ReadyNum	预消费数	N	VARCHAR
UnackedNum	未确认数	N	VARCHAR
JiYaTime	消息积压时间	N	DATETIME
Threshold	阈 值	N	VARCHAR

(2) RabbitMQ 消息队列. 此消息队列包含了 Bufr 数据消息队列、CTS 文件消息队列和 DPC 文件通知消息队列. Bufr 数据消息队列用于接收地面自动站上传的 Bufr 二进制数据, 此类数据主要包括小时和分钟数据, 自动站将数据以 Bufr 的形式写入到 RabbitMQ 的消息队列中, CTS 的质控程序进行数据质量控制, 将质控后的数据回写到原来的 RabbitMQ 消息队列中, 然后供消费者获取, 当前需要监控的消息队列有 44 个; CTS 文件消息队列主要用于省级向中国气象局传输数据, 当前需要监控的消息队列有 11 个; DPC 文件通知消息队列主要用于 CTS 和 DPC 之间的解耦, CTS 将文件通知打包消息推送到 DPC 上游的 RabbitMQ 消息队列中, 然后进行消息拆分, 随后将拆分的消费转发到下游 RabbitMQ 的各个消息队列中让每个入库程序进行消费, 当前上游需要监控的消息队列有 1 个, 下游需要监控的消息队列有 372 个.

表 2 消息队列和消费者

中间件类别	队列名称	消费者类型
Kafka	STDB_USR_SOD_SURF_WEA_CHN_PRE_MIN_ACCU_TAB	服务库
	HADB_SURF_WEA_CHN_MUL_HOR_TAB	历史分析库
	…	—
RabbitMQ	CAWN_MDOS_G.0002.0008.R001_001 SURF_PQC_A.0001.0041.R001_001 AGME_PQC_E.0001.0007.R001_001 …	缓冲库

RabbitMQ[20] 提供了 HTTP API 手册, 通过 API 接口 (http://localhost:15672/api/queues) 可以获取每个队列的全部属性, 所有 API 调用都需要做权限验证, 需在请求头部中加入权限验证信息. 本系统采集 RabbitMQ 消息队列数据信息的示例代码如下所示.

```
list_tmp_name = []
list_tmp_num = []
list_tmp_consumer = []
res = requests.get(url=dec(queue_url), auth=HTTPBasicAuth(username=dec(usr), password=dec(pwd)))
if res.status_code == 200:
    queues = json.loads(res.text)
    for queue in queues:
        node = queue.get("node")
        state = queue.get("state")
        queue_name = queue.get("name") queue_consumer_count = str(queue.get("consumers"))
        jiya_num = str(queue.get("messages"))
        ready_num = str(queue.get("messages_ready")) unacked_num = str(queue.get("messages_unacknowledged"))
```

3.2 “天擎” SOD 同步、备份、清除监控功能设计

“天擎” SOD (存储与管理系统) 中同步任务主要是结构化数据同步, 是对虚谷缓冲库同步到虚谷服务库和 Gbase 历史分析库的每一张数据表的同步任务进行监控, 采集的信息包括同步任务 ID、任务运行状态、任务名称、任务运行失败的时间等; 备份任务是对虚谷缓冲库、Gbase 历史分析库、虚谷文件索引库、虚谷结构化数据库_基础库_台站信息表、Gbase 历史分析库_基础库_台站信息表中的数据进行备份 (包含表结构)、将以上数据周期性的备份到 NAS, 采集的信息包括备份任务停止运行的类型、任务名称、备份表名、任务停止运行的时间、四级编码等; 清除任务是对虚谷缓冲库中的数据表进行周期性的删除, 采集的信息包括清除任务运行失败的时间、数据库的类型、清除资料的名称、资料表名、清除时间等. 相关的信息如表 3 所示.

3.3 上行数据传输状况信息采集

3.3.1 “天擎”国家站和区域站 Bufr 数据延迟监控功能设计

本监控功能是对 81 个国家站和 1 735 个区域站小时 Bufr 数据是否存在传输延迟的情况进行监控, 以 2 分钟时效作为标准, 通过数据到达数据库时间进行判断. 利用虚谷缓冲库数据表 SURF_WEA_CHN_MUL_HOR_TAB 的字段 D_DATETIME, D_IYMDHM 检索数据, 最终生成数据数据传输延迟报表.

表 3 存管系统监控任务

监控任务名称	监控任务个数	数据表	API接口	数据格式
同步	211	T_SOD_JOB_SYNC TASK_INFO	http://localhost:8048/sod_sync/rest/getallstatus	JSON
备份	540	T_SOD_JOB_INFO T_SOD_JOB_BACK UP_INFO		
清除	120	T_SOD_JOB_INFO T_SOD_JOB_CLEA R_INFO		

3.3.2 “天擎”自动站土壤水分数据完整性监控功能设计

自动站土壤水分数据完整性监控是通过土壤水分数据中心站和“天擎” API 接口进行监控[21]. API 接口资料代码为 AGME_CHN_SOIL_HOR, 服务接口为 getAgmeEleByTimeAndStaIDAndDepth, 通过不同的土壤深度 (20 个层级) 进行监控. 需要监控的站点有 81 个, 采集的信息包括站名、站点地址、站点的联系方式、数据开始缺失的时间、连续没有接收到数据的时间、土壤深度、时次等.

3.3.3 “天擎”高空和精细化文件监控功能设计

“天擎”高空数据和精细化数据是气象数据中的核心资料, 是制作天气预报的基础, 气象部门对此类数据有着非常严格的时效考核标准, 高空数据每天有 4 个考核时次, 包含 9 个站点. 精细化数据每天有 5 个考核时次, 包含 82 个站点, 本系统采用直接访问 NAS 的方式采集监控对象的相关属性, 采集信息包括站号、考核时次、数据落盘的时间、文件存储的地址、文件名、文件类型、站点数量、文件大小等.

3.4 “天擎”服务器硬件监控功能设计

“天擎”服务器硬件监控是通过管理口获取服务器

硬件的运行状态, 比如硬盘、内存、CPU、电压、温度等. 通过网络爬虫的方式提取相关硬件的信息, 当前需要监控的服务器有 93 台.

3.5 日志信息管理

日志管理是对应用程序及基础设备生成的日志事件进行处理的全过程. 包含日志的收集、解析、分析、存储和归档等, 日志管理的目标是为系统故障溯源及分析提供依据. 本系统采用日志管理方式, 对系统的每个功能模块、关键流程、网络状态、主机运行状态等信息进行实时日志写入. 在每个被监控的主机上采用部署代理的方式进行信息采集, 每个代理都不一样, 都是根据实际业务需求定制开发.

3.6 参数配置

本系统提供了参数配置功能, 可以在系统业务化运行以后根据不同需求进行动态调整, 灵活的参数配置功能不仅使得程序开发和基础配置信息进行了分离, 而且提高了系统的扩展性和通用性. 参数配置项主要包括 5 个部分: 站点信息配置、站点 URL 配置、数据库配置、FTP 服务配置、微信告警平台[22] 配置 (参数包括 agentid、corpid 及 corpsecret) 等.

4 系统应用效果

目前该系统已经在实际业务中进行了长时间的应用, 实际应用效果总结如下: (1) 系统稳定性方面. 在一年多的时间保持连续不间断的运行, 每个功能和模块都运行稳定可靠, 未出现任何故障; (2) 安全性方面. 需要和互联网进行数据交互, 本系统部署在气象业务内网, 和互联网进行数据交互时采用单向访问的方式, 即由气象业务内网将告警数据推送给互联网, 相反, 从互联网是不能访问气象业务内网的, 这样保证了系统的安全性, 也符合相关网络安全的要求和规范; (3) 系统扩展性方面. 扩展性是应对业务系统变化的能力, 良好的扩展性可以减少系统开发运维的成本, 本系统的每个模块都是根据实际的业务需求进行开发, 与其他的模块是相互独立的, 对可扩展的部分进行了隔离. 部分模块告警效果图如图 4 所示.

图 4 系统告警界面

5 结束语

本文根据“天擎”实际业务的运行和发展需求, 设计了对气象大数据云平台“天擎”的监控告警系统, 实现了对“天擎”各类数据资源的统一告警, 保证了“天擎”系统的稳定运行. 该系统设计的总体思路是对监控资源的集约化管理, 符合全国气象信息标准化、集约化的服务理念. 系统投入业务运行之后, 显著地提高了数据传输的及时性和完整性, 解决了传统的人工监控效率低、自动化管理程度不高的问题, 对气象信息化高质量可持续发展具有重要的保障意义. 后续工作将会利用系统每个模块产生的日志对系统常见故障进行汇总分析, 并加入机器学习和人工智能算法, 对系统运行可能产生的故障进行预测, 给运维人员预留故障处理时间.

参考文献

1 黄志, 黄珩, 梁维亮, 等. 基于“天擎” DPL 的业务融入设计

与应用初探. 气象研究与应用, 2022, 43(1): 73–77. [doi: 10.19849/j.cnki.CN45-1356/P.2022.1.13]
2 刘媛媛, 何文春, 王妍, 等. 气象大数据云平台归档系统设计及实现. 气象科技, 2021, 49(5): 697–706. [doi: 10.19517/j.1671-6345.20210083]
3 王宝云, 卢兴来, 黄晓龙, 等. 基于 ZABBIX 的新一代天气雷达 ROSE 系统监控平台. 气象科技, 2021, 49(5): 730–737. [doi: 10.19517/j.1671-6345.20200526]
4 赵伟, 王蓓, 张士祁, 等. 基于 Prometheus 的 openGauss 监控系统的关键技术及验证. 郑州大学学报 (理学版), 2022, 54(6): 74–81. [doi: 10.13705/j.issn.1671-6841.2022084]
5 彭振华, 苟伟强, 张兰英. 基于 Nagios 的气象设备监控系统. 计算机系统应用, 2017, 26(8): 60–65. [doi: 10.15888/j.cnki.csa.005916]
6 邱菊, 王岩, 黄佩卓, 等. 大型电力企业基于 GBase 分布式数据仓库建设初探. 计算机应用与软件, 2018, 35(5): 184–189. [doi: 10.3969/j.issn.1000-386x.2018.05.033]
7 李鹏程, 刘应波, 王锋, 等. 海量异构科技文献信息资源的非结构化存储研究. 计算机应用与软件, 2018, 35(5): 73–77, 88. [doi: 10.3969/j.issn.1000-386x.2018.05.013]
8 张晓, 张思蒙, 石佳, 等. Ceph 分布式存储系统性能优化技术研究综述. 计算机科学, 2021, 48(2): 1–12. [doi: 10.11896/jsjkx.201000149]
9 陈法河, 柴小丽. 基于 Ceph 存储系统的小文件存储优化方案. 计算机系统应用, 2022, 31(2): 108–113. [doi: 10.15888/j.cnki.csa.008322]
10 王锦涛, 张海明. 面向科研领域的分布式对象存储系统. 计算机系统应用, 2020, 29(7): 82–88. [doi: 10.15888/j.cnki.csa.007456]
11 刘志勇, 何忠江, 刘敬龙, 等. 统一数据湖技术研究和建设方案. 电信科学, 2021, 37(1): 121–128. [doi: 10.11959/j.issn.1000-0801.2021019.]
12 刘绍刚. 基于 NAS 的私有云存储平台的设计与实现. 计算机测量与控制, 2017, 25(4): 205–208, 212. [doi: 10.16526/j.cnki.11-4762/tp.2017.04.056]
13 陈乐, 余粟, 王盟. 基于分布式集群的高可用日志分析系统的设计. 中国电子科学研究院学报, 2020, 15(5): 420–426. [doi: 10.3969/j.issn.1673-5692.2020.05.005]
14 孙庆鑫, 雷迎春, 龚奕利. 基于共享存储的 MPP 数据库连接执行研究. 计算机工程, 2018, 44(6): 24–28. [doi: 10.19678/j.issn.1000-3428.0047104]
15 Mumtaz M, Ping L. Forty years of attacks on the RSA cryptosystem: A brief survey. Journal of Discrete Mathematical Sciences and Cryptography, 2019, 22(1): 9–29. [doi: 10.1080/09720529.2018.1564201]
16 左卫刚. 基于 Python 的新闻聚合系统网络爬虫研究. 长春师范大学学报, 2018, 37(12): 29–33. [doi: 10.3969/j.issn.1008-178X.2018.06.007]
17 张杨, 柳晨光, 张冬雯, 等. 面向 Java 多线程机制的软件重构方法. 北京理工大学学报, 2018, 38(11): 1149–1155. [doi: 10.15918/j.tbit1001-0645.2018.11.008]
18 常征, 吕勇. 基于正则表达式的海量数据清洗系统. 计算机应用, 2019, 39(10): 2942–2947. [doi: 10.11772/j.issn.1001-9081.2019030492.]
19 郝鹏海, 徐成龙, 刘一田. 基于 Kafka 和 Kubernetes 的云平台监控告警系统. 计算机系统应用, 2020, 29(8): 121–126. [doi: 10.15888/j.cnki.csa.007611]
20 余永城, 翁秋华, 段卿, 等. RabbitMQ 在气象通信系统中的应用研究. 计算机技术与发展, 2020, 30(4): 216–220. [doi: 10.3969/j.issn.1673-629X.2020.04.041]
21 董良淼, 李宇中, 覃月凤, 等. “天擎”预报服务客户端开发及接口应用技巧. 气象科技, 2022, 50(2): 297–302. [doi: 10.19517/j.1671-6345.20210380]
22 刘静静, 王光宇. 监控与告警技术在信息系统运维中的研究与应用. 中国计算机用户协会网络应用分会 2017 年第二十一届网络新技术与应用年会论文集. 重庆: 《计算机科学》编辑部, 2017. 259–262, 270.

(校对责编: 孙君艳)