

一种数据驱动定时调度的方法

李 亮^{1,2} 田 丰^{1,2} 刘中一^{1,2} 蒋晓莲^{1,2} 王思宁¹

LI Liang TIAN Feng LIU Zhongyi JIANG Xiaolian WANG Sining

摘 要

运价发布业务产品是民航运价系统的数据基础,为航空公司提供数据维护、发布和查询业务,通过大量定时执行的应用任务或夜维程序,为航空公司导入大批量数据,处理后发布或导出到运价计算或运价搜索系统。不同航空公司有不同的调度需求,需要方便灵活管理作业的调度策略,也要易维护和高可用。基于此,提出一种利用数据库技术,将定时任务信息存入数据库表,基于数据驱动的定时调度方法,实现任务调度配置与业务代码之间的解耦,具有任务调度数据集中、便于维护等特点,可提供多种调度模式,并通过前后端分离的方式,通过前端页面方便地维护任务调度数据,极大地提升应用的处理效率和维护成本。

关键词

数据驱动; 定时任务; 作业调度; 调度中心; Quartz; Crontab

doi: 10.3969/j.issn.1672-9528.2024.03.011

0 引言

21 世纪以来,中国国内民航行业发展迅速,国内运价业务越来越多样化立体化,运价数据呈几何倍增^[1]。如何保证运价数据准确、快速、稳定地响应给数据消费者,给用户一个安全快捷的使用体验,成了备受关注的问题。

中国航信(Travelsky)是国内主导的信息技术及商务服务供应商,其运营的民航运价系统是支撑机票销售环境的关键系统。国内运价发布业务产品是运价系统的数据基础,为国内 40 余家航空公司提供数据维护、发布和查询业务^[2]。国内运价发布产品包括运价导出 DFE、运价维护 DFF、运价导入 DFI 及运价发布 DFP 等 4 个主要模块,其中有大量定时执行的应用任务和夜维程序。各个模块通过定时任务为航空公司导入大批量数据,又通过定时任务将处理后的数据发布或导出到运价计算和运价搜索系统。不同航空公司有不同的调度需求,因此需要方便灵活管理作业的调度策略,实现易维护和高可用。

目前主流的定时调度服务方式通常是采用 Linux 服务器的 Crontab 定时任务配置或 Spring-Quartz 作业调度框架^[3-4]。采用 Crontab 调度在更新任务配置时需操作员登入应用服务器进行人工修改^[5-6],增加了操作风险。Spring-Quartz 框架可以实现定时配置管理,但是每次修改配置都需重启应用服务生效,甚至某些场景需要修改代码重新部署应用^[7-8]。而且国

内运价发布模块 DFP 和国内运价导入模块 DFI 要求为每个航空公司配置各自的定时任务,40 多个航司都需要人工手动配置管理,生产环境的多节点部署,需要人工对配置文件拆分任务,不同航司之间配置也不能有交叉,维护难度大,定时任务无法随时启停,仅能通过修改配置项参数后重启服务完成,无法随时随地监控任务执行状态,不方便排查问题。

国内运价发布产品通过调度中心将调度任务数据准确地提供给调度服务器,同时调度中心与业务代码解耦势在必行。

1 技术方案

利用数据库技术可以解决现有不同类型的调度应用中调度方式和调度配置不一致的问题,实现任务调度配置与业务代码之间的解耦,具有任务调度数据集中、便于维护等特点。该方法可提供多种调度模式,通过前后端分离的方式,前端页面实现图形化操作^[9],方便地维护任务调度数据,极大提升了应用的处理效率和维护成本。本方案包括任务扫描、任务调度、任务报警等功能,保证了任务调度的及时性和准确性。

1.1 系统组成

本方法主要包括任务管理、日志管理、失败告警等组件,如图 1 所示。(1)任务管理组件:根据数据库中定义的任务表配置对应的任务数据,用于在程序运行后,加载到任务驱动引擎中。定时扫描数据库,通过数据的变化来控制任务的启停。每条任务都有各自的参数和触发时间定义。(2)日志管理组件:任务执行的生命周期中,对任务的执行过程进行记录,记录的数据存储于数据库中,便于排查问题和管理维

1. 中国民航信息网络股份有限公司 北京 101318

2. 北京市民航大数据工程技术研究中心 北京 101318

护。(3)失败告警组件:任务执行失败或者遇到异常导致的任务执行结果为失败,该组件可以通过邮件的方式,发送给指定的相关人进行告警提醒处理。

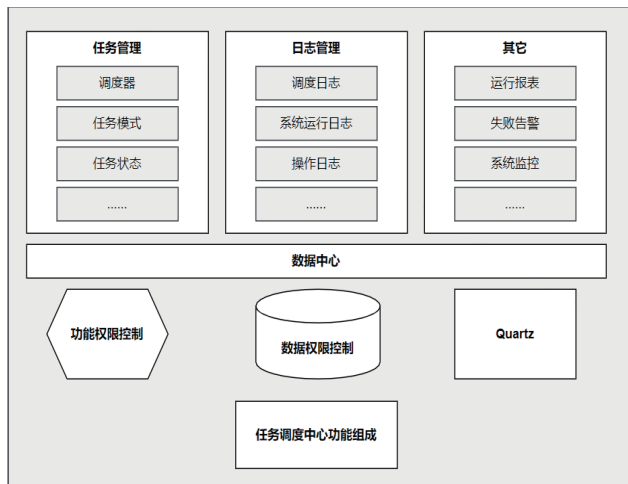


图 1 任务调度中心模块组件

1.2 实现原理

(1) Quartz 框架基础上二次开发

Quartz 是 OpenSymphony 开源组织在 Job scheduling 领域又一个开源项目,它可以与 J2EE 与 J2SE 应用程序相结合,也可以单独使用。Quartz 可以用来创建简单或运行十个、百个、甚至是好几万个 Jobs 这样复杂的程序。

本方案在 Quartz 2.3.2 的基础上,进行二次封装,实现任务调度中心,保证任务调度的稳定性。

(2) 调度器增量加载任务数据设计

系统启动后,初始化一个静态变量日期,系统扫描数据后,把资源表中该调度器的“最后更新日期”最大值存储至该字段,下次扫描的时候,依据该值,进行下一次的扫描,从而实现增量更新数据的设计。因此,每次修改数据后,想让新修改的数据可以被调度器来识别,应该也同时修改资源表中该调度器的“最后更新日期”字段,尽量生成当前日期。

(3) 自定义扫描任务设计

应用节点根据配置文件,按照指定的配置时间间隔,执行定时扫描,扫描任务数据,通过程序校验和处理,组装成调度器任务数据,并加载到指

定的调度器中,完成调度器数据加载。加载后,调度器即可按时触发相应任务。

(4) 任务实现接口反射设计

设计一个接口方法,在系统执行过程中,通过数据库,得到任务数据中配置的“执行类”数据,通过反射判断是否是接口的实现类,如果是实现类则执行该实现类的实现方法。利用反射技术可以实现多种数据源或实现类,具体的业务实现应该在该实现类中实现完成,通过不同的参数满足不同业务的需要^[10]。

(5) 新增业务维度:调度器资源、任务、任务日志

通过这三个维度的数据库表设计,前端页面可以实现任务管理页面查看具体任务执行状态和任务数据的功能,通过任务数据又可以查看到具体任务日志,也可以查看到任务的集群执行节点地址。

(6) 详细流程

步骤一:将任务数据按照指定字段配置到数据库指定表中。

步骤二:设定调度任务的触发时间、触发参数、触发节点,等待调度任务自行触发。

步骤三:设定集群中调度程序的配置文件,设定集群名称以及其他参数。

步骤四:程序实现调度框架接口,部署程序并启动,通过调度任务参数,任务数据自动加载,并按照设定执行。

步骤五:任务运行后,可以通过任务日志表,可以查看任务运行状态以及运行日志。

任务调度中心实现关系如图 2 所示。

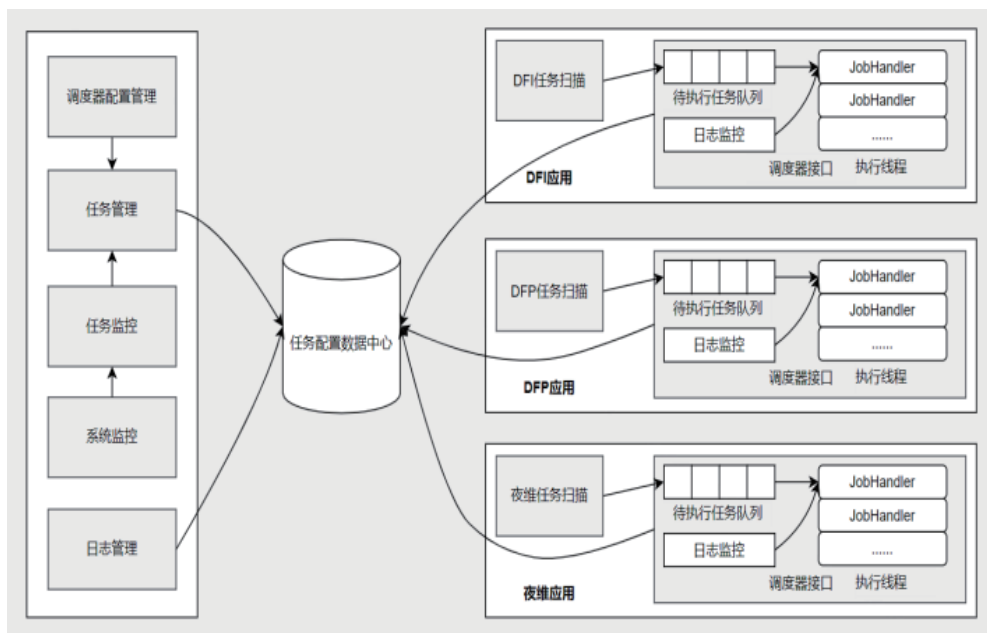


图 2 任务调度中心实现关系

2 工程实践

调度行为抽象成调度中心公共平台，平台自身并不承担任务业务逻辑，调度中心负责发起调度请求，任务调度流程完全异步化，可对密集型调度进行流量削峰^[1]。将调度和任务两部分解耦，提高系统整体稳定性和扩展性；通过数据驱动的方式，可以轻松支持可视化配置；通过页面对任务进行增删改查操作。

内部的实现机制利用 Quartz 的核心调度来完成，应用一个主任务按照配置的时间间隔扫描数据库指定数据，动态地对调度器内部的调度数据进行增删改操作，调度器内部调度数据动态同步，而调度数据又存储于数据库中，数据在后期通过各种方式提供给前端展示或者报表统计。业务应用在使用该方法时，需要用到唯一的接口 ITaskJob.doTaskJob（Map param），具体的调度任务，则需要在这个接口中开发完成。下面是 ITaskJob 接口说明。

```
/**
 * 任务接口
 */
public interface ITaskJob {
/**
 * @Description 业务实现接口方法
 * @param param 业务请求参数
 * @return void
 * @Authon test@126.com
 * @Date 15:00 2022/3/22
 */
void doTaskJob(Map param) throws Exception;
}
```

组件提供了日志记录功能，在调度接口的实现类中，使用静态方法 LogContext.appendBusinessLog（String logMessage），传入具体记录日志的信息，就可以将业务日志记录到任务日志表中，对应到具体的执行记录中，方便后续查看运行状态。表 1 是任务日志表数据说明。

表 1 任务日志表

任务日志表	task_job_log			
属性	属性名称	类型	是否必填	备注
log_id	日志 id	String	Y	与 job_id 的值相同
job_id	任务 id	String	Y	任务表主键
job_status	任务执行状态	String	N	任务状态：0-成功；1-失败
job_log	调度日志	String	N	
business_log	业务日志	String	N	
start_time	任务开始时间	Date	N	
finish_time	任务完成时间	Date	N	
create_time	任务创建时间	Date	Y	
instance_ip	执行节点 ip	String	N	
sched_name	调度器名称	String	Y	调度器名称，用于关联任务调度资源表中的数据

开发完成调度任务抽象接口的实现类后，需要将该实现类的类路径，配置到数据库 task_job（任务表）中的 job_business_class 中，任务表中的其他数据，根据实际情况配置即可。表 2 是任务表说明。

表 2 任务表

任务表	task_job			
属性	属性名称	类型	是否必填	备注
job_id	任务 id	String	Y	
job_group	任务组	String	Y	
job_name	任务名称	String	N	
job_status	调度状态	String	Y	任务状态：N-正常；S-暂停；R-运行中
is_delete	是否删除	String	Y	任务是否删除：Y-删除；N-不删除
job_policy	任务执行策略	String	Y	任务策略：C-cron 执行；F-固定时间执行
job_remarks	任务备注	String	N	
job_expr	任务表达式	String	Y	Cron 表达式或单词执行表达式 (yyyy-MM-dd HH:mm:ss)
job_param	任务执行参数	String	Y	json 字符串
job_business_class	任务执行类	String	Y	任务执行接口实现类路径
job_email	任务邮件通知	String	N	
create_time	任务创建时间	Date	Y	
create_user	任务创建人	String	N	
update_time	任务修改时间	Date	Y	
update_user	任务修改人	String	N	
role_id	角色 id	String	Y	数据权限对应的角色 id
sched_name	调度器名称	String	Y	调度器名称，用于关联任务调度资源表中的数据

任务资源表中的数据，不用用户关注，在系统启动后，自动记录表中的数据，用于监控集群部署时各个节点的状态。表 3 是任务资源表说明。

表 3 任务资源表

任务资源表	task_job_resources			
属性	属性名称	类型	是否必填	备注
sched_name	主键 id	String	Y	
instance_id	实例 id	String	Y	
instance_ip	实例 ip	String	Y	
instance_class	实例所包含的实现类	String	Y	
create_time	创建时间	Date	N	
update_time	节点资源更新时间	Date	N	

程序启动前，需要完善 task.properties 配置文件，该配置文件中绝大部分使用默认配置，但是以下几项需要注意或修改。

- (1) system.scan.class.patch=com.test，指定扫描任务调度接口实现类路径。
- (2) 邮件部分需要根据自己的实际情况进行配置，下面是一个邮件配置示例：

```
# 邮件服务器 SMTP 地址
mailHost=smtp.126.com
# 邮件服务器端口
mailPort=465
# 发送人邮箱账号
mailUser=test@126.com
```

```
# 发送人邮箱密码
```

```
mailPassword=test
```

连接时间限制，单位毫秒，限制邮件服务器建立连接的时间长短，默认无限制

```
connectionTimeout=3000
```

邮件发送时间限制，单位毫秒，发送邮件时内容上传的时间长短，默认无限制

```
writeTimeout=3000
```

(3) 数据库相关配置根据实际情况修改，下面是一个数据库配置示例：

数据库驱动

```
driver=oracle.jdbc.driver.OracleDriver
```

数据库连接串

url=jdbc:oracle:thin:@//127.0.0.1:1521/test

用户名

user=test

密码

password=test

(4) `org.quartz.scheduler.instanceName=D-FP-DPRICV3-NODE999`，如果调度任务采用集群部署，那么在集群中每个实例都必须有一个唯一的 `instanceId`，但是应该有一个相同的 `instanceName`（默认“`QuartzScheduler`”）。

(5) 其他的配置，可以根据实际情况进行对应修改，也可以不修改采用默认配置。

程序启动后,调度中心会自动扫描数据库中配置的数据,并将任务加载到调度器中,图3是启动日志。

参考文献:

- [1] ALVES U, CAETANO M. Analysis of ticket price in the airline industry from the perspective of operating costs, supply and demand[J]. Aviation in focus-journal of aeronautical sciences, 2017,7(2):109-113.
- [2] 刘辉,李亮,李莉,等.一种可回溯的缓存数据存储方法[J]. 信息技术与信息化, 2023(8): 99-101+105.
- [3] 屈洪雪,陈双,徐海洋.基于 Quartz 的可视化代理定时任务的研究与设计[J]. 电子技术与软件工程, 2021(18):44-46.
- [4] 殷玮玮. 基于 Quartz 的任务调度系统的设计与实现[D]. 南京: 南京大学, 2022.
- [5] 郭迪,赵政文,王玺.基于 cron 的计划任务时间管理的设计与实现[J]. 现代电子技术, 2011,34(14):62-64+74.
- [6] 侯增江,王勇,饶磊.一种高可用性的计划任务管理方法[J]. 计算机与现代化, 2012(12):153-156+160.
- [7] 胡鹏. 基于共享计算资源的任务调度系统研究与设计[D]. 南京: 南京邮电大学, 2023.
- [8] 彭浩西. 基于 Quartz 的分布式定时任务调度系统的设计与实现[D]. 南京: 东南大学, 2022.
- [9] 任桂禾. 大数据处理支撑平台调度子系统的设计与实现[D]. 北京: 北京邮电大学, 2015.
- [10] 费廷伟,刘淑芬,屈志勇,等. Java 反射驱动的规则引擎技术研究[J]. 计算机应用, 2010,30(5):1324-1326+1330.
- [11] 彭永强. 分布式综合任务调度平台的设计与应用[D]. 北京: 北京交通大学, 2020.

【作者简介】

李亮(1977—),男,湖南祁阳人,硕士,高级工程师,
研究方向:民航信息化技术。

田丰(1980—)，男，北京人，本科，高级工程师，研究方向：民航信息化技术。

刘中一(1987—),男,山西大同人,硕士,高级工程师,
研究方向:民航信息化技术。

蒋晓莲(1977—)，女，四川达州人，硕士，工程师，研究方向：民航信息化技术。

王思宁(1987—),男,辽宁沈阳人,本科,工程师,
研究方向:民航信息化技术。

(收稿日期: 2023-12-26)

[illegible]

图3 启动日志

3 结语

本文提出了一种基于数据驱动的定时调度方法，将定时任务存入关系型数据库如 Oracle、Mysql 中，实现调度中心与任务代码解耦，并像对普通数据一样，通过数据库对任务状态和任务调度进行控制管理。本文方法已在中国航信国内运价发布产品中完成了实践验证，达到了轻量级、接入简单、开发迅速、功能稳定的设计目标，具有操作简单、易于控制和维护、热部署、高可用的优点。