文章编号：2095-2163(2020)12-0108-08 中图分类号：TS198;TP391 文献标志码：A

数据路由在纺织工业互联网平台的设计与实现

李 锋，王凯跃
（东华大学 计算机科学与技术学院，上海 201620）

摘 要：针对纺织工业互联网中数据在云端和边缘侧传输的问题，本文引入数据网关作为两端数据传输的中间件，并在数据网关中加入数据路由模块，此模块通过“路由元数据”规定不同类型数据的传输逻辑与传输位置，如针对高频的“周期性数据”采取压缩数据规模的方式，“事件数据”则通过动态路由的方式及时传输到指定应用，“非实时数据”通过动态二级缓存的形式传输到指定位置。经过测试，在加入数据路由模块后，云端获取非实时数据的时间减少 61%-64%；周期性数据传输的请求等待时间减少 87%-88%，数据上传规模减少 47%-52%；事件数据经过动态路由能够准确传输至指定应用，实现了数据在云端和设备端准确、高效传输，优化了纺织工业互联网的局部架构。

关键词：工业互联网；数据网关；数据路由

Design and implementation of data router in Internet platform of textile industry

LI Feng, WANG Kaiyue
(College of Computer Science and Technology, Donghua University, Shanghai 201620, China)

【Abstract】 Aiming at the problem of data transmission in the cloud and the edge side in the textile industry internet, the data gateway is introduced as the middleware of data transmission at both ends, and the data routing module is added into the data gateway. This module specifies the transmission logic and transmission location of different types of data through " routing metadata". For example, the data scale is compressed for high-frequency " periodic data", The " event data" is transmitted to the specified application in time through dynamic routing, while the " non real time data" is transmitted to the specified location in the form of dynamic secondary cache. After testing, after adding the data routing module, the time for cloud to obtain non real-time data is reduced by 61% - 64%; the request waiting time of periodic data transmission is reduced by 87% - 88%, and the data upload scale is reduced by 47% - 52%; the event data can be accurately transmitted to the specified application through dynamic routing. It realizes the accurate and efficient data transmission in the cloud and equipment, and optimizes the local architecture of the textile industry internet.

【Key words】 Industrial Internet; Data Gateway; Data Router

0 引 言

近年来，纺织行业整体保持平稳发展，但受国际经济不景气、市场需求下滑、国家环保政策日益严厉、生产成本不断抬高等多种因素的影响，面临前所未有的结构调整压力。在此背景下，智能制造、绿色制造、可持续发展正成为行业发展的主旋律[1]。

通过建立纺织工业互联网平台，可使得企业有效监控生产过程，极大提升纺织产业的生产效率[2]。之前的工作中，通过 OPC UA 信号网关实现了设备的标准接口与互联互通，并对不同通信协议的设备进行统一管理[3]，但统一管理下的设备如何将不同类型的数据以独有的传输逻辑上传至云端、云端如何实现远程精准操控设备等问题仍为突出。

为此，将数据在两端传输的过程进行封装并实现模块化，命名为“数据传输模块”[4]。此部分嵌入工业互联网的云端与边缘侧，加入“数据传输模块”后的工业互联网体系结构如图 1 所示[5]。

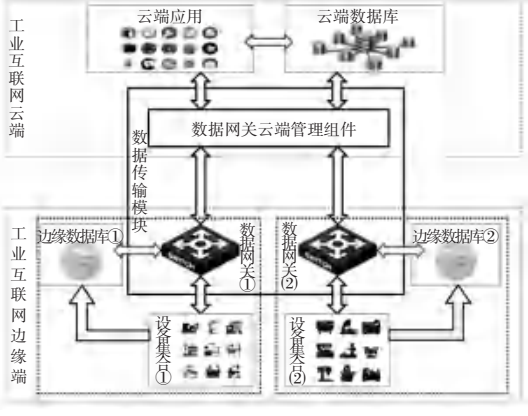

图 1 工业互联网体系

Fig. 1 Industrial Internet system

基于图 1 中的模块化设计涉及到以下概念：

(1)数据传输模块：同时横跨云端与边缘端，是

作者简介：李 锋(1969-)，男，博士，教授，主要研究方向：人工智能、嵌入式开发；王凯跃(1996-)，男，硕士研究生，主要研究方向：工业互联网。
通讯作者：李 锋 Email：lifeng@dhu.edu.cn

收稿日期：2020-10-17

云端与边缘端数据交互的核心组件，由若干边缘侧数据网关与数据网关管理组件组成。

（2）数据网关组件：是指位于边缘端的数据网关及其相连的边缘数据库与设备集合。图 1 中“数据网关①”+“设备集合①”+“边缘数据库①”的组合即为一个数据网关组件。

数据网关与边缘数据库做直接交互，而数据网关云端管理组件与云端数据库和云端应用交互，此过程涉及到数据的提取、计算、存储逻辑等内容，而数据类型多样，通过手动编写逻辑的方式虽可完成数据处理逻辑，但若无规范的数据类型处理逻辑将不便于体系的后续开发与维护。

基于上述不足之处，在数据传输组件中加入数据路由组件。在对数据路由展开论述前，对数据路由的基本概念做简单介绍：

（1）数据路由本身是一个较为抽象的概念，广泛存在于“数据传输模块”的边缘数据网关和数据网关云端管理中心。

（2）数据路由以组件形式存在，一个数据路由组件下包含“逻辑执行区”与“元数据配置区”两部分。其中，元数据配置区域的每一条“路由元数据”都规定了一个类型或一个属性的执行逻辑。

1　需求分析

针对数据路由的需求分析从内外两个角度入手：

（1）数据路由内部执行过程分析；

（2）依据不同类型的数据，数据路由在“数据传输模块”中的执行逻辑分析。

1.1　内部需求分析

基于“以配置文件的形式规定处理逻辑”这一概念，需要在数据路由中实现配置区域与执行区域的“分离逻辑”。在配置区域中，规定了某类型或某个数据的配置项称为“元数据”，配置区域由若干元数据组成。而执行区域即数据路由的逻辑代码部分，一旦实现分离逻辑，需尽可能保证“逻辑执行区”代码的通用性，不涉及具体元数据的逻辑执行。

数据路由处理数据的过程主要包括[6]：

（1）数据类型/名称判断：数据路由组件识别接收数据的类型或名称；

（2）元数据匹配：根据上一步得到的数据类型或名称，找到匹配该类型或名称的元数据；

（3）元数据执行：依据此元数据规定的逻辑执行并输出结果。

1.2　外部执行逻辑需求分析

任何类型数据在数据传输中都严格遵守数据路由内部逻辑，但外部执行逻辑则存在较大不同。

在纺织设备的生产过程中，需要处理的数据主要有两类，即实时数据、非实时数据，实时数据又分为周期数据和事件数据[7]。

（1）周期数据。在设备生产中产生的数据，特点是频率高、数据传输规模大，故使用传统关系数据库作为缓存实时接收，不能满足实时性要求。周期数据往往需要经过来自两个组件中的数据路由组件顺序处理后才可存入目标位置。

（2）事件数据。设备在生产过程中设备端状态发生改变时产生的实时数据（如启动、停止、报警、故障等）[8]，特点是较其他数据的传输优先级更高，需尽快上传至云端对应接收该事件的应用中。同时在关系数据库中对此事件进行记录，称为“事件日志”。

（3）非实时数据。主要包含了设备的配置信息（如设备编号、设备包含的监控单元等配置信息），属于静态数据，几乎不会发生改动。

相对于前两种数据，数据路由处理非实时数据的逻辑相对简单，但却有较大性能优化空间。

数据路由处理 3 种数据类型的过程以及云端应用获取 3 种数据类型的过程如图 2 所示。

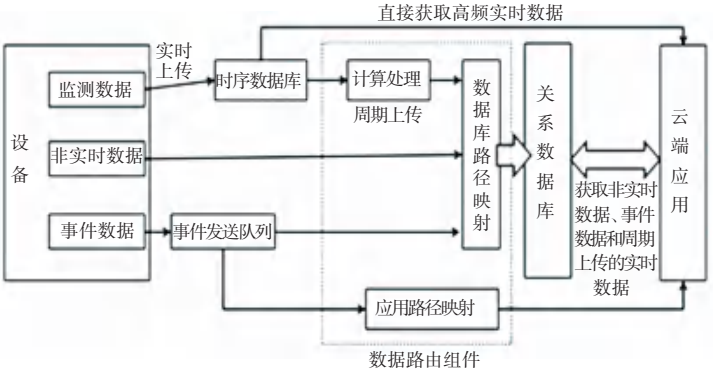

图 2　数据传输与云端获取数据过程

Fig. 2　Data transmission and cloud data acquisition process

经过上述存储方案,初步规划了 3 种数据类型的存储方案,数据路由实现每种数据类型操作。

2 数据路由分析

2.1 路由元数据定义

2.1.1 元数据类型

数据路由对不同数据的处理方式,决定了数据路由内部亦存在不同类型的路由元数据。具体说来,有处理监控数据的“计算元数据”、负责将数据存入关系数据库的“数据库路径元数据”以及直达云端应用的“事件动态元数据”等。不同元数据的形式大同小异,但都包含了输入元素与输出路径两大基本要素。

2.1.1.1 计算元数据

计算元数据是周期数据上传时的底层基础元数据。针对实时数据的操作形式也不仅限于普通计算,格式或单位的转换、直接映射等也包含在内,计算元数据遵循“就近处理”的基本原则[9]。

由于计算元数据主要面向的周期数据,周期数据主要来自单属性在周期时间内的一组数据集,故计算元数据绝大多数都是单输入类型。执行结果上传至云端后继续执行路径映射过程。在云端实时性要求较低的情景下,云端可过滤掉大量非必要高频实时数据,简化数据的分析处理过程,而将承受高频大规模实时数据的过程交给边缘侧时序数据库,实现边缘侧与云端的异步发送与接收。即使在少数情况下云端应用获取一段时间的高频数据,也可通过对该数据集进行高效压缩后上传。

计算元数据极大简化了云端用户获取数据的方式,主要体现在用户只需要从云端的存储设备中获取所需数据,而处理实时数据的过程交给数据网关中的数据路由组件处理即可。

计算元数据的配置文件属性及其解释见表 1。

2.1.1.2 数据库路径元数据

作为面向关系数据库的路由元数据,数据库路径元数据规定了输入在关系数据库的映射位置——具体到表和列[10]。

与计算元数据不同,数据库路径元数据面向所有需要存储在数据库中的数据类型,主要包括:

(1)周期数据经过计算元数据处理后的结果;

(2)事件数据的日志记录功能;

(3)非实时数据的路径映射。

由于每个元数据中所需的输入与数据库表的列名一一对应,故数据库路径元数据属于多输入类型[7]。

数据库路径元数据的配置文件同样存在“输入”和“输出”两大配置属性,不同的是数据库路径元数据中直接规定了每个输入的输出位置,不存在中间的“处理逻辑”,形式见表 2。

表 1 计算元数据配置文件内容

Tab. 1 Calculation metadata configuration file content

位置	元数据名称	配置属性		属性值
云端数据路由	设备监控元数据	元数据映射位置		ServerMysql-router-monitorPropertyTable
		元数据触发数据类型		MonitorProperty
		输入	属性 1	avg
			属性 2	max
			属性 3	variance
边缘侧数据路由	周期数据统计元数据	元数据类型		MonitorProperty
		操作 1	操作名	CountAvg
			结果名	avg
		操作 2	操作名	CountMax
			结果名	max

表 2 数据库路径元数据配置文件内容

Tab. 2 Contents of database path metadata configuration file

配置属性			内容解释
配置信息	元数据名称		(同计算元数据)
	元数据 ID		(同计算元数据)
	输入类型		(同计算元数据)
输出目标配置信息	输出数据库类型		元数据路径映射的数据库类型
	输出数据库名称		该数据库类型下的数据库名称
	输出数据库表名称		该数据库下的指定表名
	输入名称 1 输入名称 2 输入名称 3 ……	输入列名 1 输入列名 2 输入列名 3 ……	规定进入该元数据的输入映射到表中的具体列

2.1.1.3 事件动态元数据

数据库路径元数据与计算元数据均属于静态元数据,即这两种元数据以文件形式存在,数据路由将其读入内存后,只负责执行而不对元数据的配置内容进行修改。将依据云端应用订阅情况在数据路由内存中动态生成的路由元数据称为“事件动态元数据”。

此类元数据同样包括静态元数据的输入→输出的过程,并包含了路径的动态生成,步骤如下:

步骤 1 云端应用向数据路由发送针对某事件的订阅;

步骤 2 数据路由将此订阅转换为一条“事件数据→云端应用”的映射路径,即路径生成;

步骤 3　对应事件数据依据输入为此事件的事件动态元数据执行，传输到订阅应用中。

引入事件动态元数据，程序得以进一步简化，使数据路由在逻辑上更为灵活。

2.1.2　元数据的实例化与触发

路由元数据以配置文件形式存在，若需要发挥作用还需要将该元数据从“元数据配置区”导入“逻辑执行区”中，形成一个元数据实体，此过程称为元数据实例化[7]。

定义 1　数据路由模块直接操控元数据实体执行路由。

定义 2　元数据实体的配置信息来自元数据配置文件，而执行时所需输入则来自外部接收数据。

定义 3　一个元数据实体形成后，仍然需要等待所需的所有输入就绪后执行，该过程称为“触发”。否则此时实体处于“等待”状态。

定义 4　元数据实例化过程，不会主动产生实体，而是取决于输入和元数据类型。

多输入的元数据实体的触发过程如图 3 所示。

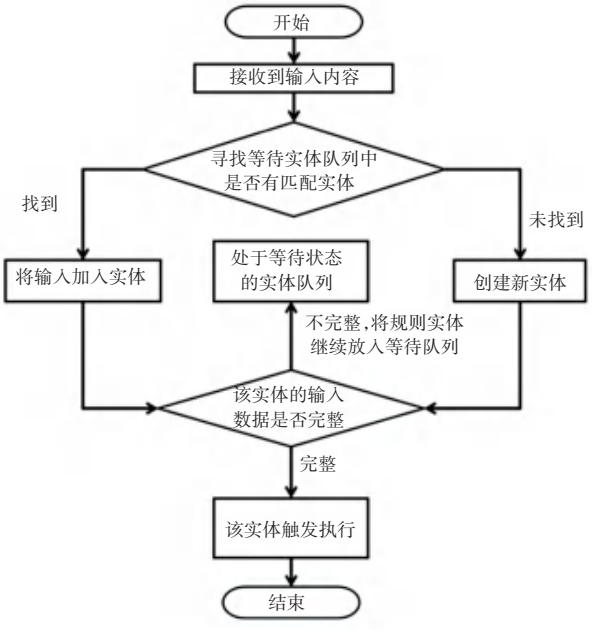

图 3　多输入元数据实体的触发过程

Fig. 3　Trigger process of multi input metadata entity

2.1.3　路由过程

从数据的路由执行过程看。将数据从某数据源，经过若干路由元数据的操作，最终映射到目标存储位置的过程，称为一个路由过程[7]。

由于关系数据库中每张表均有不同的应用场景，故一个属性可能存储于多张表中，也会出现在多个数据库路径元数据所需的输入中，即一个属性产生了多个路由过程。

2.2　数据路由组成

在数据路由的“逻辑执行区”中，存在着多组件的协调配合。依据不同功能，分为元数据库、管理器、操作中心、执行器、内容接收器[9]。

(1)元数据库。包括了以配置文件的形式存在的元数据内容，配置文件以相对统一的标准格式规范执行逻辑。

(2)管理器。该部分是数据路由中与路由元数据库直接连接的组件。负责检测元数据库中路由元数据的变化(增加、删除、修改等)。数据路由初始化时，该部分首先从路由元数据库中将所有元数据的 ID 号、名称和所需输入内容读入，等待实例化形成元数据实体，供元数据执行器进行调用[11]。此外，元数据管理器还负责管理“实体等待队列”中处于“等待”状态的元数据实体。该部分是数据路由中的核心内容。

(3)操作中心。该组件包括预置的针对周期数据的操作，几乎所有计算元数据的处理逻辑均需调用此部件中相应功能，作为工具类提供所需的基础操作。

(4)执行器。该部分执行被触发的元数据实体，依据元数据实体规定的操作。对于计算元数据，在操作中心中寻找该操作的执行逻辑，调用该方法，得到输出结果；对于其他元数据，则依据元数据中的输入输出对应关系映射到指定路径即可[11]。

(5)内容接收器。该部分是数据路由的入口组件，负责接收所有请求，包括接收底层数据、获取云端应用请求等等，依据不同请求类型执行不同处理逻辑。

2.3　数据路由优化

面对云端纷繁复杂的请求类型，数据路由同样需要作出相应优化调整。

2.3.1　静态二级缓存路由模式

云端应用主要通过访问云端数据库获取非实时数据，若所有非实时数据都需要存储在云端关系数据库，则可能造成短时传输负载激增，并增加云端数据库存储负担。若只允许云端应用高频访问的非实时数据存入云端数据库，而将所有非实时数据缓存于数据网关数据库中，云端应用即可在云端获取高频非实时数据，而对于低频访问的内容则由云端向指定数据网关发起，此模式即称为静态二级缓存路由模式[6]，其中“二级”是指数据网关和云端两级。虽然在访问低频数据时可能会延长请求时间，但由此却减轻了云端数据库存储压力。至此，数据路由实现了初步的缓存功能。静态二级缓存如图 4 所示。

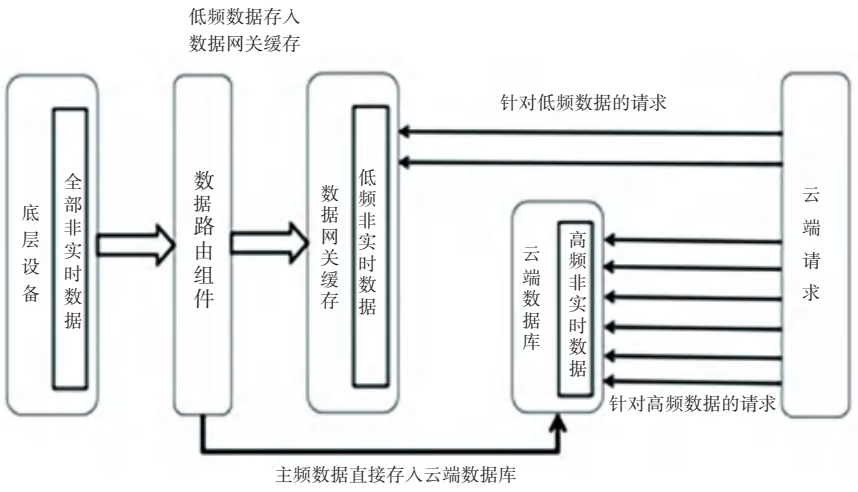

图 4 静态二级缓存结构

Fig. 4 Static L2 cache structure

2.3.2 动态路由模式

虽然静态二级缓存可极大优化云端请求逻辑，但在此模式下的高频数据仍为静态，若云端请求的高频数据发生改变，此模式效率将有所下降。为进一步提升系统运行效率，可在云端数据路由组件中创建“请求接收池”，使数据路由基于云端用户的请求类型对部分元数据实体的映射路径作出适应性调整，此过程称为动态路由[12]。动态路由主要存在于非实时数据中。现给出动态路由模式中的相关定义。

定义 5 请求接收池。在数据路由中创建的用以统计云端应用访问非实时内容的区域。

定义 6 平均访问时间(AVT)是指在云端访问非实时数据时的平均等待时间。针对非实时数据读取存在 3 种形式，即不采用路由元数据、静态二级缓存模式、动态二级缓存模式，平均等待时间的计算方式也截然不同，其中动态模式相较于静态模式还需计算“元数据内容迁移”的执行时间。

3 测试与分析

3.1 搭建平台

为测试数据路由在数据传输中的性能表现，搭建一套仿真工业互联网平台进行实验。

在实验平台内，一台服务器模拟为云端，内部装有模拟应用、云端数据网关管理中心和关系数据库。边缘侧包括一台终端，并包括模拟数据网关、模拟信号网关及模拟底层设备。

实验平台通过微服务实现整体软件架构，通过相互调用实现通信。形式上，模拟云端部署名为“云端数据服务”的微服务作为云端数据网关管理中心，部署名为“云端路由服务”负责云端路由操作。同时云端采用 mysql 关系数据库用来存储数据路由映射后的结果。边缘侧数据网关分别包含用来完成本地数据处理的“本地数据服务”和实现数据网关本地映射的“本地路由服务”。边缘侧数据网关采用微型 mysql 数据库和接收实时数据的 TSDB 时序数据库。数据网关通过信号网关连接了模拟退煮漂机和模拟印花机各一台。搭建平台结构如图 5 所示。

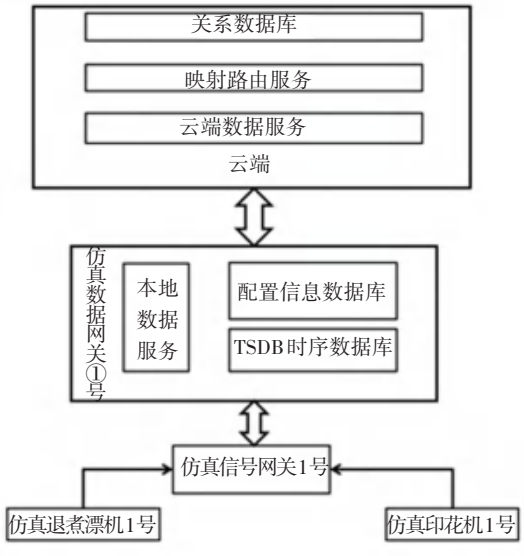

图 5 搭建平台结构

Fig. 5 Build platform structure

3.2 性能测试

3.2.1 场景 1：设备、数据网关与云端的绑定测试

本测试场景中组件之间的相互绑定与连接效果的优劣决定了系统的运转性能。

3.2.1.1 测试 1：测试映射路由基本功能并搭建“静态二级缓存”

在数据网关检测到设备连入时，主动从设备所属的信号网关获取设备的非实时数据如设备的配置信息、设备下监控单元层级结构等非实时数据。由于在数据网关和云端的数据路由组件中均存在针对非实时数据的数据库路径元数据，经过元数据映射到数据库的对应位置。

在边缘侧数据网关与云端的数据路由模块中均加入以下数据库路径元数据：

(1)设备信息元数据。

(2)监控单元信息元数据。

虽然两部分存在同名的映射元数据，但依据“路由存储策略”，云端的元数据较为简化，保留设备和监控单元的基本信息以及云端应用高频访问的属性，形成“静态二级缓存”。

之后模拟云端请求环境：云端模拟应用不定时向云端发送访问非实时数据的请求，模拟请求内容中，调高针对放置于云端的高频非实时数据的请求频率。

最后设置“不添加静态二级缓存”的对照组。让模拟请求环境分别向两种情况下的系统发送请求。性能结果见表 3。

结果表明，加入静态二级缓存系统的较不加入二级缓存的对照组系统在平均请求等待时间上有一定程度的缩减。

表 3　静态二级缓存性能比较结果

Tab. 3　Static L2 cache performance comparison results　ms

请求数量	存储模式	
	不使用二级缓存	静态二级缓存
25	164	120
50	298	209
100	576	313

3.2.1.2　测试 2：非实时属性动态路由测试

在测试 1 加入“静态二级缓存”的基础上，云端数据路由中设置“请求接收池”，统计请求访问类型，开启“动态二级缓存”功能。模拟云端请求环境时，同样指定若干属性为初始高频请求。加入测试 1 中的静态二级缓存实验组作为对照组，进行测试，结果见表 4。

表 4　动态二级缓存性能比较结果

Tab. 4　Comparison results of dynamic L2 cache performancems

请求数量	存储模式	
	静态二级缓存	动态二级缓存
25	120	58
50	209	114
100	313	210

结果表明：随云端请求变化的动态二级缓存模式，虽然存在元数据迁移的时间损耗，但相较于静态二级缓存而言，平均请求等待时间上进一步缩减。

3.2.2　场景 2：周期数据自动化上传与存储测试

本部分测试步骤为：

(1)周期数据从设备中实时提取；

(2)使用时序数据库存储周期数据并将其以固定时间间隔从时序数据库提取；

(3)将一组周期数据集放入数据路由中执行，并存入关系数据库中。

步骤 1　在信号网关中提供了“subscribe”订阅的方法，可以追踪到其订阅属性的数据变化；

步骤 2　使用 OpenTSDB 时序数据库存储周期数据。存储的过程中，将周期数据所属设备作为 Metric，若该实时属性属于该设备下的某监控单元，则 Tags 填入周期数据所属监控单元的名称和 Id 号，若直接属于设备的实时属性则 Tags 处标明该属性直系设备。在 TSDB 中查询一段时间内的实时数据的结果如图 6 所示。

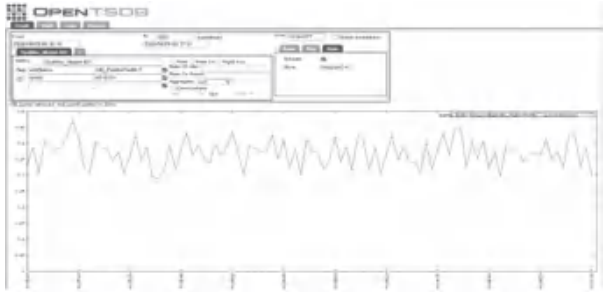

图 6　TSDB 查询结果

Fig. 6　Query results of TSDB

步骤 3　在云端和边缘侧的数据路由中加入以下路由元数据：

(1)周期数据统计元数据；

(2)设备监控元数据。

两个路由元数据的内容见表 5。

为了验证本部分计算元数据传输实时数据时在带宽使用、传输效率上的优势，需将未添加计算元数据的场景作为对照组。从上传数据规模和请求等待时间进行分析，结果见表 6、表 7。

从结果可以看出，添加了计算元数据的实验组在上传数据规模上较对照组大幅较少。在请求等待时间上，由于对照组在请求等待时间内进行颗粒度转换的操作，故等待时间较实验组大幅增加，性能上也远不如实验组。

表 5 设备监控元数据与周期数据统计元数据内容

Tab. 5 Metadata content of equipment monitoring and periodic data statistics

位置	元数据名称	配置属性		属性值
云端数据路由	设备监控元数据	元数据映射位置		ServerMysql-router-monitorPropertyTable
		元数据触发数据类型		MonitorProperty
		输入	属性1	avg
			属性2	max
			属性3	variance
边缘侧数据路由	周期数据统计元数据	元数据类型		MonitorProperty
		操作1	操作名	CountAvg
			结果名	avg
		操作2	操作名	CountMax
			结果名	max

表 6 请求等待时间比较结果

Tab. 6 Comparison results of request waiting time ms

请求数量	操作方式	
	不使用颗粒度转换	使用颗粒度转换
25	771	89
50	1229	161
100	2193	297

表 7 上传数据规模比较结果

Tab. 7 Comparison results of upload data scale KB

请求数量	操作方式	
	不使用颗粒度转换	使用颗粒度转换
25	163.7	79.6
50	309.5	159.8
100	596.1	319.2

3.2.3 场景 3:设备“事件数据”动态路由测试

事件数据的动态路由执行步骤为:

(1)针对监控单元和设备分别模拟三种随机事件,事件类型见表 8。

(2)在云端建立若干模拟应用,分别订阅应用所需事件,具体订阅细节见表 9。

表 8 模拟随机事件列表

Tab. 8 List of simulated random events

事件名称	中文解释
unit_act_up	监控单元实际值超过上限
unit_act_down	监控单元实际值低于上限
unit_shutdown	监控单元停止工作
device_start	设备启动
device_stop	设备停止
device_hot	设备过热

表 9 模拟应用订阅事件列表

Tab.9 List of simulated application subscription events

事件名称	订阅该事件的应用序号
unit_act_up	App2
unit_act_down	App3
unit_shutdown	App1、App2、App4
device_start	App0、App2
device_stop	App0
device_hot	App2、App3、App4

(3)设备端随机发送若干事件,云端查看每个模拟应用接收事件情况,应用接收到的事件结果见表 10。

表 10 应用接收事件情况汇总

Tab. 10 Summary of application receiving events

应用序号	接收事件	
	设备或单元名	事件名称
App0	DSBMac_Model	device_stop
	DyeMac_Model	device_start
	DSBMac_Model	device_start
	DyeMac_Model	device_stop
App1	Obj_PadAirPreMU2	unit_shutdown
App2	Obj_PadAirPreMU2	unit_shutdown
	Obj_SurTempVenMU1	unit_act_up
	DyeMac_Model	device_start
	DSBMac_Model	deveice_start
	Ojb_PadAirPreMU9	unit_act_up
App3		
App4	Obj_PadAirPreMU2	unit_shutdown

从结果看,数据路由根据云端应用的订阅请求准确将事件数据发送至对应应用,初步实现“事件动态路由”功能。

4 结束语

本文在数据网关中加入数据路由,规定不同数据类型向云端传输的逻辑,进一步完善了工业互联网架构。本文在数据路由设计方面主要取得以下成果:

(1)以数据路由为核心,以配置文件的形式规定非实时数据、周期数据和事件数据的传输模式,将传输逻辑从程序中解耦;

(2)实现边缘侧数据网关内周期数据颗粒度转换以降低数据传输规模;

(3)提出并实现了非实时数据的“动态二级缓存”功能,提高了云端请求效率;

(4)针对云端应用,按需订阅设备端事件数据类型,形成“事件动态路由”。

与此同时,实际测试中发现,云端请求非实时数

据时，云端的请求属于静态执行过程，在没有规定请求路径时，请求首先会默认经过云端数据库，之后再去边缘侧数据库查询，如果查询内容不在云端数据库，将造成请求过程的复杂度，造成云端资源浪费。

引入以数据路由为核心的数据网关，将更便于纺织融入工业互联网元素，完善工业互联网体系，帮助纺织企业进行较为精准的战略决策。

参考文献

[1] 蒋高明，高哲，高梓越. 针织智能制造研究进展[J]. 纺织学报，2017，38(10)：177-183.

[2] 张洁，吕佑龙，汪俊亮，等. 大数据驱动的纺织智能制造平台架构[J]. 纺织学报，2017，38(10)：159-165.

[3] 李锋，张坤，原丽娜. 基于 OPC UA 的纺织智能染整车间信息模型研究与实现[J]. 纺织学报，2020，41(2)：149-154.

[4] 邹萍，张华，马凯蒂，等. 面向边缘计算的制造资源感知接入与智能网关技术研究[J]. 计算机集成制造系统，2020，26(1)：40-48.

[5] 赵勇军. 基于云计算的数据存储架构研究[J]. 智能计算机与应用，2014，4(4)：90-93.

[6] 周颖. IFC 数据到关系型数据库的自动映射方法研究[C]// 郭红领，罗柱邦. 第四届全国 BIM 学术会议论文集. 2018.

[7] CHEN Xinyong. On the general data mapping and transformation scheme between complex systems[J]. Computer programming skills and maintenance，2017(18)：49-51.

[8] 李海威.基于云计算的物联网数据网关的建设研究[J].计算机技术与发展，2018，28 (1)：188-190.

[9] 梁吉胜，李天阳，王惠霞，等. 数据映射技术在 ETL 过程中的应用[J]. 计算机系统应用，2012(7)：155-159.

[10] 陈心咏. 试谈复杂系统间的通用数据映射转化方案[J]. 电脑编程技巧与维护，2017(18)：49-51.

[11] Mohammad Fikry Abdullah. The mapping process of unstructured data to structured data[C]// Kamsuriah Ahmad. International Conference on Research & Innovation in Information Systems. IEEE，2014.

[12] 段春梅. 云计算分布式缓存技术在海量数据处理平台中的应用[J].智能计算机与应用，2016，6(1)：13-15.

(上接第 107 页)

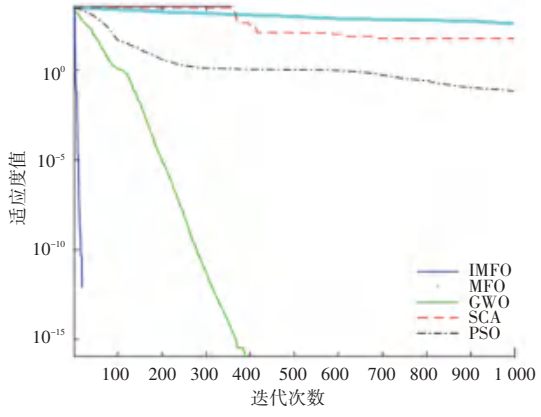

图 8　F_8 函数平均收敛曲线

Fig. 8　The average convergence curve of F_8 function

4　结束语

本文在标准飞蛾扑火算法优化的基础上，采用佳点集理论初始化种群，使得初始种群分布更加均匀，更具遍历性；随后在位置更新公式中引入了正弦函数描述的、随迭代次数变化的动态惯性权重，平衡算法的探索和开发能力；采用重心反向变异策略，拓宽了搜索广度，避免算法出现早熟收敛。通过 8 个经典测试函数验证算法性能，在最优值、标准差、Wilcoxon 秩和检验等检测指标下，与 MFO 算法、GWO 算法、SCA 算法、PSO 算法进行比较，IMFO 算法具有更好的性能。

参考文献

[1] MIRJALILI S. Moth-flame optimization algorithm：A novel nature-inspired heuristic paradigm[J]. Knowledge - Based Systems，2015，89(11)：228-249.

[2] BHANDARI A K，MAURYA S，MEENA A K. MFO - based thresholded and weighted histogram scheme for brightness preserving image enhancement[J]. Image Processing，IET，2019，13(6)：896-909.

[3] XZ A，YFA B，LE L A，et al. Ameliorated moth-flame algorithm and its application for modeling of silicon content in liquid iron of blast furnace based fast learning network - ScienceDirect[J]. Applied Soft Computing，2020，94.

[4] SAYED G I，HASSANIEN A E，NASSEF T M，et al. Alzheimer's Disease Diagnosis Based on Moth Flame Optimization[C]// International Conference on Genetic and Evolutionary Computing. Springer，Cham，2016.

[5] YASIR S，ADNAN H，FARHAN A，et al.CAMONET：Moth-Flame Optimization (MFO) Based Clustering Algorithm for VANETs[J]. IEEE Access，2018，12：1-1.

[6] MUDULI D，DASH R，MAJHI B . Automated breast cancer detection in digital mammograms：A moth flame optimization based ELM approach[J]. Biomedical Signal Processing and Control，2020，59：101912.

[7] ANDERSON - COOK，CHRISTINE M . Practical genetic algorithms[J]. Publications of the American Statal Association，2004，100(471)：1099-1099.

[8] DJELLALI H，GHOUALMI N. Improved Chaotic Initialization of Particle Swarm applied to Feature Selection[C]// International Conference on Networking and Advanced Systems. LRS Laboratory，Badji Mokhtar University，Annaba，Algeria；LRS Laboratory，Badji Mokhtar University，Annaba，Algeria；，2019.

[9] RAHNAMAYAN S，JESUTHASAN J，BOURENNANI F，et al. Computing opposition by involving entire population[C]// Evolutionary Computation. IEEE，2014.

[10] MIRJALILI S，MIRJALILI S M，LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software，2014，69(7)：46-61.

[11] MIRJALILI S. SCA：A sine cosine algorithm for solving optimization problems[J]. Knowledge Based Systems，2016，96(96)：120-133.

[12] KENNEDY J EBERHART，EBERHART R. Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neural Networks，1995：1942-1948.