doi:10.11835/j.issn.1000-582X.2020.241

面向流数据的实时处理及服务化系统

狄 程[a,b],杨中国[a,b],韩燕波[a,b],刘 晨[a,b]

(北方工业大学 a. 大规模流数据集成与分析技术北京市重点实验室;b.数据工程研究院,北京 100144)

摘要:流数据的处理需求复杂多变,业务人员要进行相应的算法定制,不仅需要相关的编程知识,更要应对繁琐的处理流程和冗长的开发周期。为解决上述问题,文中设计并实现了基于流程建模的流数据处理及服务化系统,提供了对于多源流数据的实时接入,流数据服务化以及流数据处理服务化的能力。该系统将流数据处理过程封装为服务提供给用户,允许用户拖拽组合流数据处理和服务化模块、配置相关参数,定义流数据处理及服务化的过程,快速又自然地实现流数据处理及服务化的任务,将处理结果经由服务路由实时推送到其他应用系统,满足不同的业务需求。案例分析表明,与传统的流数据处理系统相比,本系统具有高效、灵活、可配置等特点,在实用性、可用性和伸缩性方面都更有优势。

关键词:流数据;视图驱动;实时处理;规则引擎;服务化

中图分类号:TP311.1 **文献标志码**:A **文章编号**:1000-582X(2020)07-075-09

View-driven flow data oriented real-time processing and service system

DI Cheng[a,b], *YANG Zhongguo*[a,b], *HAN Yanbo*[a,b], *LIU Chen*[a,b]

(a. Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data;
b. Institute of Data Engineering North China University of Technology, Beijing 100144, P. R. China)

Abstract: The processing requirements of these data are also complex and changeable. The business personnel need to carry out corresponding algorithm customization, which involves not only relevant programming knowledge, but also cumbersome processing flow and lengthy development cycle. In order to solve the above problems, this paper designed and implemented a stream data processing and service system based on process modeling, which provided real-time access to multi-source stream data, stream data service and stream data processing service. The system encapsulated the stream data processing process into a service provided to the user, allowing the user to drag and drop the combined stream data processing and service module, configure relevant parameters, define the process of stream data processing and service, and implement stream data processing and service-oriented tasks quickly and naturally. The processing results were pushed to other application systems in real time through service routes to meet

收稿日期:2019-11-24

基金项目:国家重点研发计划资助项目(2017YFC0804406)。

Supported by National Key Research and Development Program of China(2017YFC0804406).

作者简介:狄程(1996—),男,硕士研究生,主要从事服务计算的研究,(E-mail) 10237528@qq.com。

通讯作者:杨中国,男,助理研究员,(E-mail)yangzhongguo@ncut.edu.cn。

different business needs. Case studies show that the system is more efficient, flexible, and configurable than traditional streaming data processing systems, and has advantages in terms of usability, availability, and scalability.

Keywords: stream data; view driver; real-time processing; rule engine; service

随着互联网的高速发展,海量的流数据不断从数千个来源生成,包括社交网络、移动应用、传感器、电子商务交易等等[1]。流数据具有有序、数量大、快速和连续到达等特征,使得其价值密度会随时间不断变化、难以有效共享和利用[2]。目前,大批开源的流数据处理框架和应用系统之间较为独立分散,没有形成一套从数据接入到计算再到数据应用服务化的流程。同时,流数据处理系统的配置与编写程序较为繁琐,所需的技术往往具有陡峭的学习曲线,让用户无法把工作的重心放到业务逻辑本身。

针对这些问题,Laska 等[3]设计了一种管道体系结构,能够实时处理时空数据流的可重用管道体系结构。Lee 等[4]提出了一种新的基于 xml 的传感器数据流处理中间件 UbiCore (Ubiquitous Core)从传感器收集数据,处理收集到的传感器数据,并自动将处理后的结果交付到相关的后端应用程序。黄廷辉等[5]提出了一种在 Spark 平台上基于梯度优化决策树的分布式城市交通流预测模型,实现了对城市实时交通流量的准确预测。Lee 等[6]提出一种分布式流处理系统,该系统支持多种负载自适应技术和多种故障情况下的容错机制。

在流程配置方面,Cao 等[7]提出一种用于过境网络计算的工作流,该工作流用于分析多个无限制的传输数据提要的连续到达。主流的一些数据挖掘平台,如 Weka[8],RapidMiner[9],Berthold[10]和 Orange[11]都实现了基于视图的流程建模,用户可以通过拖放组件来构建复杂的工作流,缺点是并不支持流数据的实时处理。Kranjc[12],González-Jiménez [13]和 Zhang 等[14]都设计实现过基于云的流数据处理工具,并提供可视化建模语言来描述执行流程。

笔者基于数据流程与处理规则分离的思想,设计并实现了基于流程建模的流数据处理及服务化系统。实现流数据服务化与流数据处理服务化,在解决复杂的数据接入问题的前提下,还可以保证快速的实现流数据处理流程,并及时生成服务提供给其他用户或系统,满足了不同业务人员对数据获取的不同需求。系统为进一步研究流数据主动服务、动态演化机制提供验证系统和测试平台,在并行接入大规模传感器和处理逻辑动态变化适应的测试实验中,取得较高的性能。

1 系统设计

1.1 需求分析

系统为了满足用户在大数据平台下大规模流数据无缝接入、个性化定制所需流数据处理逻辑和自适应选择服务,用户只需要关心自己的数据处理逻辑,不用关心代码实现等问题。同时,流数据的处理过程中,处理逻辑通过代码片段组合式以及已有算法的选择式进行控制,即通过拖拽式界面实现算法的选择和代码片段输入。其核心设计思想是实现用户自主选择个性化业务系统所需的处理逻辑,数据按照定义好的格式流入,实现各种数据处理、数据服务等功能,实现应用与数据、平台脱耦。研究工具提供了业务中常见的数据操作,是多个 Flink 底层算子的结合和具体化,具有更好的业务适用性。

根据以下功能需求,将系统的功能划分为 5 个模块,分别是流程配置模块、规则解析引擎模块、数据接入模块、实时处理模块和服务化模块。系统的功能模块如图 1 所示。

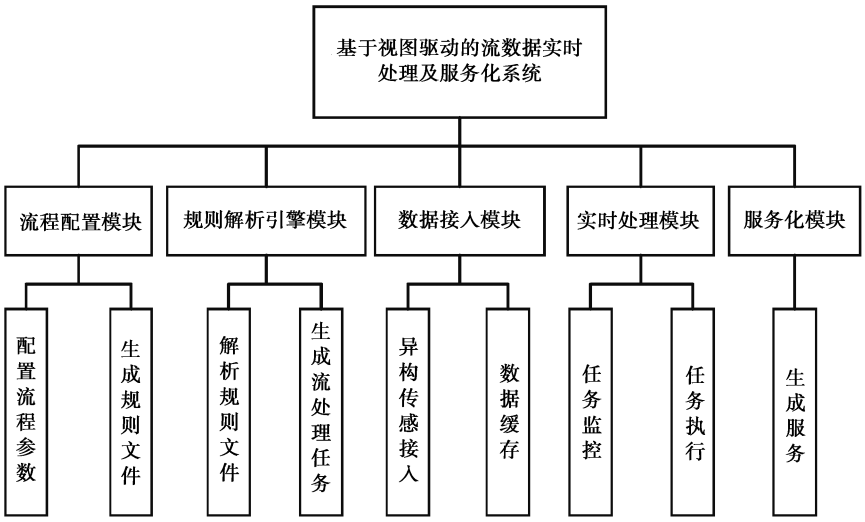

图 1　系统的功能模块图

Fig. 1　The function module diagram of the system

1.2　整体架构及流程

系统的整体架构如图 2 所示,分别是承载大数据平台的物理层、对数据进行核心处理的服务层、适用于多种流数据业务的应用层。

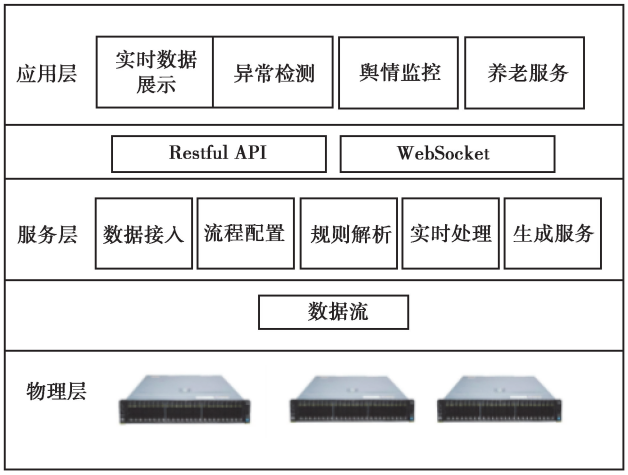

图 2　系统整体架构图

Fig. 2　System architecture diagram

物理层提供资源调度、均衡容灾、服务注册、动态扩容缩容功能,一般是由用户所在单位提供环境。

服务层衔接物理层和应用层,为应用层提供数据支撑。允许用户拖拽组合流数据处理单元(如过滤、展示、清洗、多流合并等操作),定义流数据处理过程,配置相关参数(IP 地址、端口号、阈值等参数),快速又自然地实现流数据处理的任务,并将处理结果封装成服务(REST,WebSocket 等)提供给用户,实现服务化的自动化完成。

应用层主要是展现几种可以用到流数据处理平台的业务系统,包括实时数据分析、异常检测、金融交易等涉及流数据实时性问题的业务。

综上所述,物理层主要是涉及集群的部署(Zookeeper,Kafka,Flink 等),应用层主要是可以使用系统处理结果的一些业务系统,而服务层为系统的核心部分。系统的主要工作是服务层以及物理层部分的工作,服务层主要涉及各个模块的设计。

2　系统实现

2.1　模块设计与实现

第 1 节简述了整个系统的架构以及运行流程。下面分别介绍流程配置模块、规则解析引擎模块、数据接

入模块、实时处理模块和服务化模块的详细流程设计与模块架构。系统的模块架构如图 3 所示。

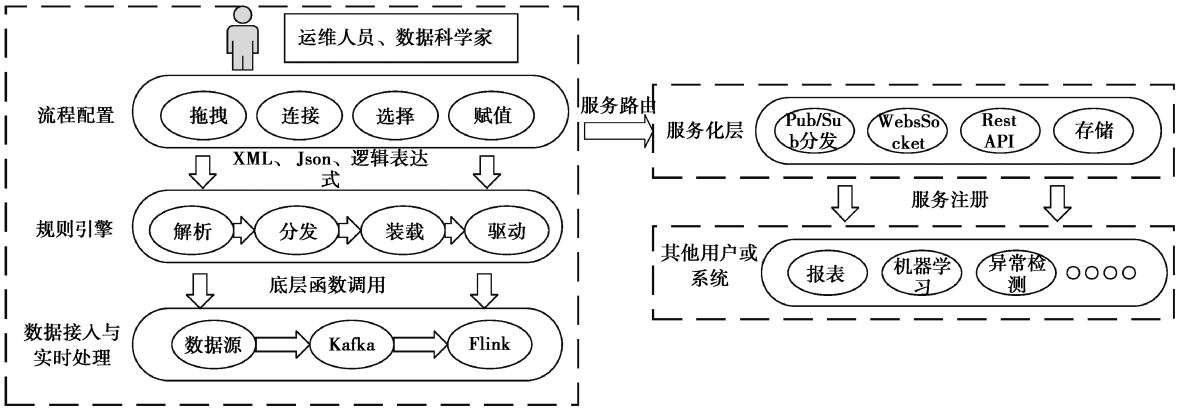

图 3 模块架构图

Fig. 3 Module architecture diagram

流程配置模块对外提供一个交互页面，页面上有不同的基本操作单元，并且可以自由配置参数值，支持从数据准备，模型部署到正在进行的模型管理的完整工作流程。用户可以用拖拽的方式选取数据工作流程中需要进行的处理单元，将具有前后数据依赖关系的操作单元进行连接。这些操作以及依赖关系将会形成逻辑文件，具体将以 XML、Json 以及逻辑表达式的形式发送到规则引擎。

流程解析引擎模块接收由流程配置模块得到的操作逻辑及相关参数进行解析，再下达到底层的实时处理模块。用户对流数据的处理过程有自己的处理规则和定义，需要提供相应的接口来实现这些处理逻辑。可行的做法是提供有限的操作准则，用户配置实现的部分是较为容易集成到并行计算中的操作。这里需要设计良好的构造接口，让处理逻辑能自动并行化。

数据接入模块为上层应用提供实时流数据接入，系统接入的绝大部分流数据都是原始未经处理的数据，这些数据格式不统一，种类繁多，直接接入系统不易进行分析计算。实时处理模块是系统中最核心的功能模块，目的是对系统接收到的流数据进行实时的分析处理。本模块将接受从规则解析引擎传递过来的函数方法以及参数，触发本地的各种逻辑操作。服务化层提供处理结果展示，WebSocket、REST 等服务。用户在流程配置模块中选择需要的服务，系统在收到实时处理过的数据后，会提供相应的服务给用户或其他系统。为了更好地支持应用和服务的并发请求、保障服务的质量，流数据查询处理操作的结果可通过消息队列将处理结果分发到应用和服务中，也往往分发到数据库中进行持久化，再以服务的方式对外提供数据访问接口。

2.1.1 流程配置模块

笔者实现了一个通过拖拽组件可视化配置任务流程，并动态生成 XML 文件的前端页面。页面由 4 个部分组成，如图 4 所示。

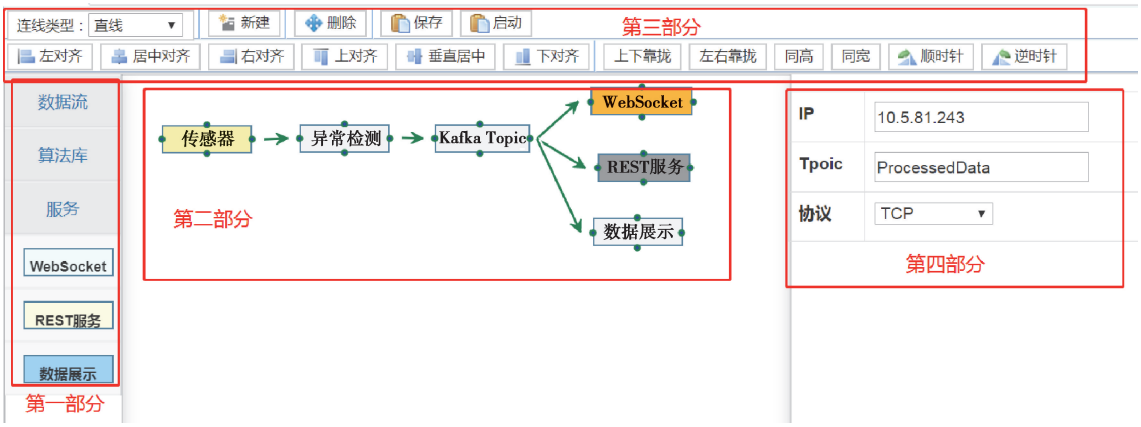

图 4 流程配置界面

Fig. 4 Process configuration interface

第一部分为左侧的组件选择模块,其中,有数据流、算法库和服务 3 类组件。传感器中有传感器、KafkaTopic、HBase 等组件,用户可以自由选择和配置数据来源;服务中有数据展示、WebSocket 和 REST,用户可以选择一个或多个服务;算法库中有异常检测、数据清洗、流拆分、流合并、WordCount、WindowJoin 等多个操作算子,用户可以根据业务需求选择操作。第二部分为右侧的属性模块,用户可在其中配置(在组件选择模块中)已选择的组件属性。其中,传感器属性有名称、IP 地址、传输协议和解析方法;服务属性有 IP 地址、端口和传输协议;算法库属性有阈值、窗口长度、窗口滑动距离等属性。第三部分为流程配置模块,用户可将选择的组件拖入其中,通过连线来动态的配置业务流程。第四部分为操作模块,有选择、新建、删除、保存、对齐等操作按钮。其中保存按钮在用户点击后,程序可以根据配置好的业务流程,自动生成 XML 文件,并发送给规则解析引擎模块。新建按钮则是用户上传之前业务流程的 XML 文件,程序可以解析文件,并在前端页面中自动生成对应的业务流程图。

2.1.2　流程解析引擎模块

在任务流程构建之后,数据流模型的解析引擎算法将任务流程解析为 XML 描述文档。XML 描述文件存储在任务流程实例数据库中。然后,XML 描述文件被转换为 DAG,用于指导任务流程实现。最后,后端程序解析 DAG,与提前封装好的代码片段做映射,“拼凑”出可执行的任务代码,即一个 Flink 作业。DAG 是一种有向无环图,用于重新编码任务流程的执行过程。XML 描述的是一种中间形式,可以轻松地将工作流转换为 DAG。流程解析引擎如图 5 所示。

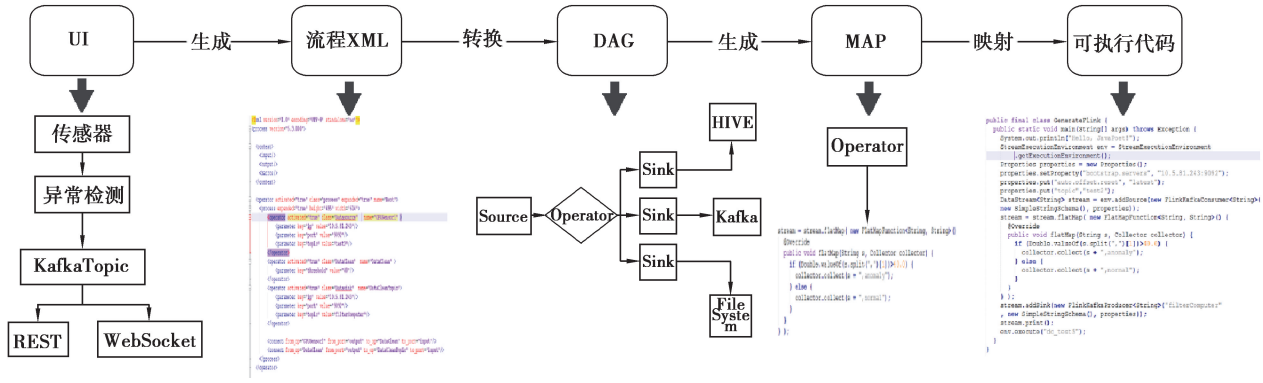

图 5　流程解析引擎

Fig. 5　Process parsing engine

1)规则单元。用户在前端界面通过拖拽连接等规则单元定义数据的处理流程,系统会将这部分逻辑以 XML 文件的形式进行存储,基本操作之间的关系将以 XML 中不同元素层次的方式展现。关系分为算子类和连接类。

算子类:在可视化界面中的每个独立单元对应一个算子 operator,每一类算子映射到对应转换接口,每个算子实例都有唯一 id,算子参数以键值对形式存在,前端页面通过填写赋值。

连接类:具有前后依赖关系的算子实例通过连接类 connect 记载到流程链上,每个连接都会自动填写生成输入输出。

2)规则解析。用户在界面的配置会形成 XML 文件,经过解析将前端逻辑映射到底层实现,如图 5 所示,在计算平台中,上述处理流程会分解成不同的算子,基于 Kafka 运维操作的会实现诸如新建主题,修改分区等操作,计算相关逻辑会在 Flink 中按照处理前后顺序形成一个有向无环图(directed acyclic graph),由于实时流数据的特殊性,计算平台会按照时间窗口对数据进行分割计算,并最终汇总,以特定形式的 Sink 流输出。

2.1.3　数据接入模块

系统支持接收的数据源包括来自传统的数据库存储、文件系统存储、消息队列、传感器直接接入等。其

中传感器的多源异构性带来了很大的挑战。

1)异构传感接入。不同类型传感器的数据格式和传输协议都不相同,造成了异构传感器接入的难题。如果为每一类传感器设备分别开发接入功能模块,开发完成后将各个模块进行集成,会导致工作量大且重复,开发成本高。因此,需要研究传感器的适配,包括连接器、解码器以及数据发布程序的适配。系统为用户设计并提供一个交互界面,用户通过界面注册要接入传感器的信息,例如,传感器名称、IP地址、数据解析方法等。用户在交互界面完成注册传感器信息,要接入的传感器模型就被系统建立。建立传感器模型后,异构传感器接入模块利用分布式大数据处理引擎Flink动态生成传感器适配器,Flink将这些适配器作为任务的一部分提交至分布式集群中运行,完成传感器数据的实时处理。

2)消息队列。Kafka是一种基于发布/订阅的分布式消息系统,但是在大数据时代,传统的消息队列不适用于大规模的数据处理,Kafka的引入主要是构建实时流数据管道,在系统或应用程序之间可靠地获取数据,解决数据接口两端的交互问题,降低系统组网和编程的复杂度。在多个分布式消费者并发的情况下,也能保证消息的有序性及消息的负载均衡。在流数据处理中,系统可以从Kafka的主题中获取连续的流数据,对数据执行一些实时处理,并将产生的连续结果输出至其他Kafka主题。Kafka有助于解决流数据处理系统面临的难题:无序数据的处理、代码更改时重新处理输入数据、有状态计算的执行等。

2.1.4 实时处理模块

实时处理模块使用Apache Flink开源流处理框架实现。Flink是一个面向分布式数据流处理和批量数据处理的开源计算平台,可对有限数据流和无限数据流进行有状态计算[15]。在服务器集群上部署Flink,系统采用Flink提供的Restful接口向集群提交Flink任务的Jar包,作业可被分解成多个任务,依靠Flink提供的高吞吐低延时计算能力,分布在集群中并发执行。

2.1.5 服务化模块

用户自定义流数据处理流程后,需要得到处理结果。系统动态为用户提供KafkaTopic,WebSocket,REST 3种服务化接口。用户可以根据自己的业务需求,选择合适的服务化接口,服务动态生成,并提供标准化的使用方法,对外提供统一的接口,保证复杂环境下的可靠传输,便于不同的应用系统和用户使用。

1)实时结果展示。通过系统的计算处理,得到的结果需要以折线图、柱状图、散点图、饼图、K线图等可视化图表在前端进行展示,将处理结果直观、丰富地展现给用户。系统使用开源插件ECharts进行可视化展示,该插件是一个使用JavaScript实现的开源可视化库,可以流畅地运行在PC和移动设备上,兼容当前绝大部分浏览器。

2)WebSocket。为实现处理结果能实时发送给用户,系统提供了WebSocket协议作为服务化接口,具体实现该服务的架构如图6所示。主要包括Kafka分布式发布订阅消息系统、服务器和客户端三大部分。在实时处理部分,将计算结果通过Producer(生产者)的send命令发布到指定的Topic(主题)中,服务器通过Consumer(消费者)向指定的Topic订阅消息以得到计算结果数据,通过WebSocket协议将数据从服务器发送给客户端,能更好地节省服务器资源和带宽,更实时地发送数据。客户端从WebSocket中得到数据,为用户提供服务。

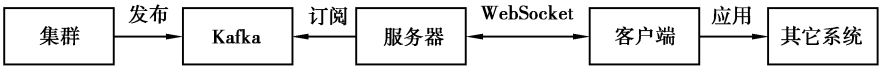

图6 WebSocket服务架构图

Fig. 6 WebSocket service architecture diagram

3)REST服务。除了通过WebSocket协议为用户提供服务之外,系统还可为用户提供Restful API接口。接口的交互数据使用JSON格式进行统一和规范。以一个异常检测结果查询来简要讲解其实现流程。

通过路由来定位服务器处理程序,具体代码为:

'GET/api/anomalydetection/getData':'AnomalyDetectionController.getData'

运行服务器处理程序,获取相应的数据,使用 JSON 格式来统一响应数据,其返回结果如图 7 所示,其中 value 为时间序列,label 标注当前值是否为异常值。

```
{
    "data": [
        { "value":"2019-8-14 08:38:17,35.6" , "label":"normal" },
        { "value":"2019-8-14 08:38:18,36.2" , "label":"normal" },
        { "value":"2019-8-14 08:38:19,36.8" , "label":"normal" },
        { "value":"2019-8-14 08:38:17,41.1" , "label":"anomaly" },
    ]
}
```

图 7　异常检测返回结果

Fig. 7　Anomaly detection results

3　测试与验证

3.1　实验环境

系统测试与验证会用到 1 台本地测试机和 6 台服务器构成的测试集群。6 台服务器构建了 Flink 集群,并配置了 Hadoop,Kafka 和 Zookeeper 等大数据相关环境。测试机启动 Web 服务,通过浏览器分别测试系统各个功能模块,进行功能、可行性和高效性测试。各个机器的配置信息如表 1 所示。

表 1　实验室配置环境

Table 1　Laboratory configuration environment

机器	硬件配置	软件配置
服务器	内存:8G 硬盘:250G 处理器:双核 机器数:6 台	操作系统:Ubuntu18.04LTS flink－1.7.2 kafka_2.12-2.2.0
测试机	内存:8G 硬盘:500G 处理器:四核 机器数:1 台	操作系统:Windows 10(64 位) 浏览器:Chrome

3.2　评价指标

3.2.1　功能测试

在系统的交互页面进行流程配置,配置相关参数,选择可视化服务。任务读取存储在本地 HBase 中的智能电网电力设备传感数据,并发送至 Kafka 的 Topic,然后对数据进行异常检测,最后将处理结果生成数据展示服务,如图 8 所示,其中红色圆点为异常点。

3.2.2　系统可行性测试

为了测试系统可行性,实验选用阿里云天池公开数据集里的淘宝用户行为数据集。数据集包含了淘宝上某一天随机一百万用户的所有行为(包括点击、购买、加购、收藏)。先过滤出其中的购买行为,然后发送至指定 KafkaTopic,通过系统前端配置任务流程,实现“实时热销商品榜单”(窗口聚合＋TopN＋WebSocket),验证系统可行性。

通过在前端拖拽组合流数据处理单元,定义流数据处理过程,配置相关参数(IP 地址、端口号、阈值等参数),配置好的流程被系统转换为可执行的 Flink 任务代码,在集群上运行,快速实现流数据处理的任务。任务开始执行后,去服务器上查看系统生成的代码,前端配置的任务流程与相应的代码片段映射如图 9 所示。

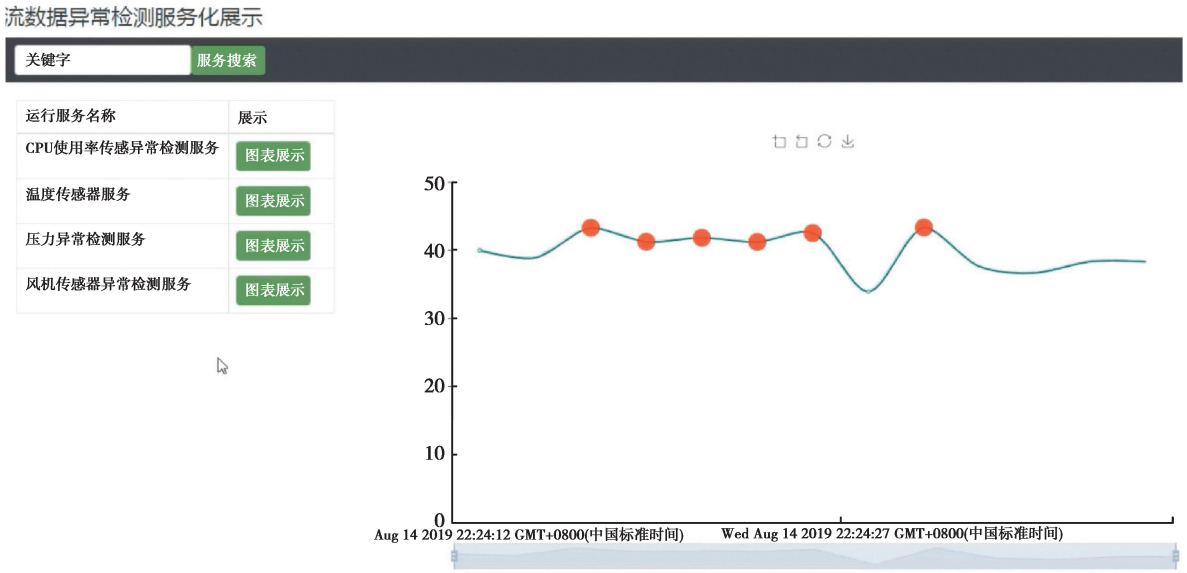

图8 实时异常检测

Fig. 8 Real-time anomaly detection

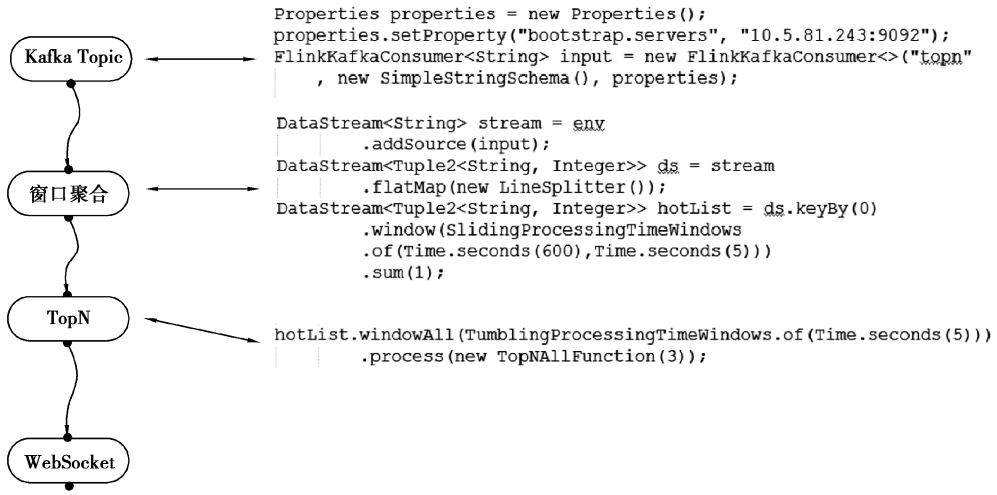

图9 任务流程与代码片段映射

Fig. 9 Task flow and code snippet mapping

最后,在客户端上查看系统提供的 WebSocket 服务,服务器通过 WebSocket 将数据实时发送给客户端,客户端从 WebSocket 中得到数据,为用户提供服务。用户在客户端查看到实时处理结果为时间与 Top3 购买量商品的 id 和购买量,方便用户在前端开发“实时热销商品榜单”功能,如图 10 所示。

```
Time: 2017-11-26 15: 20: 00
No1:  id=3845720 value=20
No2:  id=812879 value=20
No3:  id=1871901 value=18

Time: 2017-11-26 15: 25: 00
No1:  id=812879 value=20
No2:  id=3845720 value=19
No3:  id=2453685 value=16

Time: 2017-11-26 15: 30: 00
No1:  id=812879 value=19
No2:  id=3845720 value=19
No3:  id=2332370 value=17

Time: 2017-11-26 15: 30: 00
No1:  id=812879 value=21
No2:  id=3845720 value=20
No3:  id=2332370 value=18
```

图10 系统通过 WebSocket 实时发送结果

Fig. 10 System sends results in real time via WebSocket

3.2.3 流程可视化配置的高效性验证

为了测试系统的可视化流程配置功能的高效性,进行对照实验。实验室邀请 2 位熟悉流数据开发的工程师,并提供河南高速路收费站收费信息数据集,让 2 人分别使用传统的流数据处理框架(已配置好开发环境)和测试系统,在数据集上进行操作,统计过去 1 周每个收费站点的车流量,将处理结果生成 REST 服务。结果发现,使用传统流数据框架的工程师需要 10 min 才能完成数据处理任务,并且还要借助其他开发工具才能生成 REST 服务;而使用本系统的工程师,只需要花费 3 min 时间进行拖拽组件和配置参数,就能达到相同的

开发效果。

由上述测试与验证可知,系统具有高效、灵活、可配置等特点,同时支持高并发执行,在实用性、可用性和伸缩性方面都更有优势。

4 结　语

分析了目前流数据处理工作中,因为流数据的数据量大、实时到达、持续不间断、有序性且数据价值对时间敏感等特征,以及各类流数据处理框架互相独立,导致的业务人员开发效率低的问题,给出了一种基于可视化界面配置处理流数据并生成服务的系统的设计与实现。能够通过简单易懂的拖拽组合连接方式来进行流数据的实时处理和处理结果的服务化,简化开发人员的配置工作。并设计了一套规则引擎将数据处理逻辑映射成到对流数据的基本操作单元的组合连接,同时系统利用现有的 HBase、Kafka 以及 Flink 实现了高吞吐和低延迟的数据处理需求。目前,如何在系统中加入神经网络模型与主动式服务,将深度学习与服务计算结合在一起,是值得研究的问题。

参考文献:

[1] Jang J, Jung I Y, Park J H. An effective handling of secure data stream in IoT[J]. Applied Soft Computing, 2017:S1568494617302739.

[2] Babcock B, Babu S, Datar M, et al. Models and issues in data stream systems [C]. In Proceedings of the twenty-first ACM Sigmod-sigact-sigart symposium on Principles of database systems,2002:1-16.

[3] Laska M, Herle S, Klamma R, et al. A scalable architecture for real-time stream processing of spatiotemporal IoT stream data: performance analysis on the example of map matching[J]. ISPRS International Journal of Geo-Information, 2018, 7(7): 238.

[4] Lee H S, Jin S I. An effective XML-based sensor data stream processing middleware for ubiquitous service[M]. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007:844-857.

[5] 黄廷辉, 王玉良, 汪振, 等. 基于 Spark 的分布式交通流数据预测系统[J]. 计算机应用研究, 2018, 35(2): 405-409, 416.
HUANG Tinghui, WANG Yuliang, WANG Zhen, et al. Distributed traffic flow data prediction system based on spark[J]. Application Research of Computers, 2018, 35(2): 405-409, 416.(in Chinese)

[6] Lee M, Lee M, Hur S J, et al. Load adaptive and fault tolerant distributed stream processing system for explosive stream data[C/OL]. 2016 18th International Conference on Advanced Communication Technology (ICACT). New York, USA: IEEE, 2016 (2016-03-03) [2019-11-10]. https://ieeexplore.ieee.org/document/7423613.

[7] Cao H, Wachowicz M. The design of a streaming analytical workflow for processing massive transit feeds[J/OL]. Computers and Society,2017[2019-09-25].https://arxiv.org/abs/1706.04722.

[8] Tony C, Smith, Eibe Frank. Introducing Machine Learning Concepts with WEKA[J]. Methods in Molecular Biology, 2016, 1418:353-378.

[9] Mierswa I, Wurst M, Klinkenberg R, et al. Yale: Rapid prototyping for complex data mining tasks[C]// Proceedings of the 12th ACM Sigkdd international conference on Knowledge discovery and data mining. New York, USA: ACM Press, 2006: 935-940.

[10] Berthold M R, Cebron N, Dill F, et al. KNIME - the Konstanz information miner[J]. ACM Sigkdd Explorations Newsletter, 2009, 11(1): 26.

[11] Demšar J, Zupan B, Leban G, et al. Orange: from experimental machine learning to interactive data mining[M]. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004,3202: 537-539.

[12] Kranjc J, Ora č R, Podpe čan V, et al. ClowdFlows: Online workflows for distributed big data mining[J]. Future Generation Computer Systems, 2017, 68: 38-58.

[13] González-Jiménez M, de Lara J. Datalyzer: streaming data applications made easy[M]. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018,10845: 420-429.

[14] Zhang B, Yu L, Feng Y B, et al. Application of workflow technology for big data analysis service[J]. Applied Sciences, 2018, 8(4): 591.

[15] Marciani G, Piu M, Porretta M, et al. Real-time analysis of social networks leveraging the flink framework[C]. DEBS'16 Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems. New York, USA: ACM Press, 2016:386-389.

(编辑　陈移峰)