基于 Flink 实时计算的自动化流控制算法

樊春美,朱建生,单杏花,杨立鹏,李 雯
(中国铁道科学研究院,北京 100081)

摘 要: 随着现在各种业务系统的复杂多样化,数据分析的实效性要求也变得越来越高,过去的离线分析很多已经不适用于当前的生产需要,针对于大数据的实时分析变得越来越重要。以当前热门的 Flink 流处理架构为解析平台,构建了分布式实时采集解析流数据处理架构,针对于不同的数据流,只需要更新配置就可以实现业务数据的解析,极大地减少了代码开发量。为了实现该架构的配置文件更新,重点分析了当前分布式解析架构实时更新配置文件时存在的问题,提出了通过流控制的方法来更改 Flink 的解析逻辑。流控制的方法能够灵活地实时改变代码的解析逻辑,减少程序重启更新的次数,提高了应用效率。通过完成同样的日志解析入库对比了是否使用流控制算法的效果,实验结果表明加入自动流控制算法的解析架构用更少的时间完成解析结构逻辑的开发和程序部署,并且可以大大地减少延迟入库的日志量,从而最大程度地保证了流的实时性。

关键词: Flink; 流处理; Spark; 大数据; 分布式
中图分类号: TP301 **文献标识码:** A **文章编号:** 1673-629X(2020)08-0066-07
doi: 10.3969/j.issn.1673-629X.2020.08.011

Automatic Flow Control Algorithm Based on Flink Real-time Computation

FAN Chun-mei,ZHU Jian-sheng,SHAN Xing-hua,YANG Li-peng,LI Wen
(China Academy of Railway Sciences,Beijing 100081,China)

Abstract: With the complexity and diversification of various business systems,the requirement of effectiveness of data analysis is becoming higher and higher. Offline analysis in the past is no longer suitable for current production needs,and real-time analysis for big data is becoming more and more important. Taking the current popular Flink stream processing architecture as the parsing platform,a distributed and real-time processing architecture of collecting and parsing data stream is constructed. For different data streams,business data is analyzed by updating the configuration file for the architecture,which will reduce the amount of code development. In order to realize the configuration file update of this architecture,the problems existing in the current distributed parsing architecture when updating the configuration file in real time are analyzed emphatically,and then the parsing logic of Flink is proposed by flow control method. The method of flow control can flexibly change the parsing logic of the code in real time,reduce the number of program restarting and updating,and improve the efficiency of the application. By completing the same log parsing and storing,the effect of whether to use flow control algorithm is compared. The experiment shows that the analytical framework with automatic flow control algorithm takes less time to complete parsing logic structure of the development and application deployment,and greatly reduces the delay of log volume,thus ensuring the real-time performance of the flow to the greatest extent.

Key words: Flink; stream processing; Spark; big data; distributed

0 引 言

随着铁路售票系统业务的多样化,来自各个业务系统的日志解析变得越来越复杂。对于日志数据进行实时的采集和解析是当前需要解决的重要问题。鉴于大数据的应用比较广泛,尤其是分布式的架构和实时计算框架,如 Flink[1]、Spark[2] 计算平台变得越来越热门,因此现在大部分互联网公司都会采用实时计算来完成日志的采集和存储。而对于复杂的铁路售票系统,不仅需要针对不同服务层进行日志的采集,还需要对不同业务场景的不同类型的日志进行解析,为了业务数据分析的需要,可能同一份数据还要存储到不同的渠道中;与此同时系统会随着生产的需求而不断完

收稿日期: 2019-10-14 **修回日期:** 2020-02-21
基金项目: 中国国家铁路集团有限公司 2018 系统性重大项目(P2018X002);中国国家铁路集团有限公司 2019 重大项目(K2019X008)
作者简介: 樊春美(1986-),女,博士研究生,研究方向为 Web 安全;朱建生,博导,研究员,研究方向为铁路信息化。

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

善,因此也会导致日志存储格式的变动,从而需要不停地调整解析程序来适应新的需求。在每一次变动中,都需要测试整套解析程序,开发效率较慢,甚至对于一个数据流的解析要启动多个程序,造成资源的浪费,维护多个程序也变得更加复杂。于是可以采用现在常用的改变配置文件的方式来实时更新解析相关的配置。但是分布式的计算框架刷新配置文件存在着两个问题,一个是配置文件的存放问题,一般分布式的计算框架,真正的计算程序是在每个节点上执行的,配置文件如果在 master 节点上,而每个执行节点是不能读取 master 节点的配置文件的,如果每个执行节点都放一份配置文件,也无法保证每个执行节点的路径是一致的,只要其中的一个节点路径不一致就会导致程序出错,无法执行; 另外一个问题是,假设能够保证每台机器的路径都是一致的,配置文件可以放在每台机器上,这时程序每解析一条数据都要读取一次配置文件,频繁的读取必然会影响解析的效率。由于对于分布式的计算框架刷新配置存在着上述两个问题,文中提出了一种通过自动化流控制的方法,实现配置文件的实时刷新。

1　相关工作

目前针对大型数据的分析框架主要有 Hadoop、Spark、Flink 等[3-5]。以 Hadoop 为代表的大数据技术的出现,可以很好地解决大量静态数据集的数据处理与分析,但是很多数据都是实时产生的,用户希望可以实时地处理这些数据,这就需要使用流计算技术来实时处理这些数据,及时产出应用价值。

而 Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎,是一种与 Hadoop 相似的开源集群计算环境,其拥有 Hadoop MapReduce 所具有的优点,但与 MapReduce 存在的不同是任务中间输出结果可以保存在内存中,从而不再需要读写 HDFS,因此 Spark 除了能够提供交互式查询外,还可以优化迭代工作,能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。

随着 Spark 计算引擎的出现,促进了上层应用快速发展,如对各种迭代计算的性能以及对流计算和 SQL 等的支持,Flink 就在这样的背景下诞生了。Flink 的突出优势是对流计算的支持及更一步的实时性。为了对流式引擎处理有更深刻的认识,Sanket Chintapalli 等人[6]基于常用的流式计算构建了三种不同的流引擎,进行对比分析。其中 Spark streaming 是将流划分解成一系列短小的批处理作业,Flink 才是真正的流处理,批处理只是流数据的一个极限特例而已。Flink 不仅可以支持本地的快速迭代,以及一些环形的迭代任务,而且 Flink 可以定制化内存管理,其并没有将内存完全交给应用层,因此 Flink 处理大数据速度快,能够更好地满足大数据背景下应用实时计算平台的需求。

由于 Flink 框架的优势,现在有很多关于 Flink 应用的相关研究。蔡鲲鹏[7]研究了 Flink 的概念、生态系统和相关技术等理论基础并对 Hadoop 和 Flink 在处理大批量数据上的耗时和准确率进行了对比分析,针对不同的流式处理平台,分析总结了 Flink 所面临的一些挑战,为 Flink 的进一步研究提供了参考。麦冠华等[8]基于 Flink 的计算框架,设计了对大规模轨迹数据进行实时运动模式检测的算法,弥补了对于当前大规模轨迹数据只能做范围查询、近邻查询的简单处理的不足,很好地应用了 Flink 实时计算的优势。Marciani G[9]利用 Flink 框架对社交网络进行实时分析,系统架构的设计重点在于利用 Flink 的并行性和内存效率,以便能够在分布式基础设施上有效地处理大容量数据流。不仅 Flink 的应用比较广泛,对其优化的相关研究也比较多。Verbitskiy I 等[10]分析了 Flink 的执行效率,通过各种实验评估表明 Apache Flink 的性能是高度依赖于问题的; 李梓杨等人[11]通过针对大数据流式计算平台中输入数据流急剧上升所导致的计算延迟升高问题进行优化,有效地提高了现在 Flink 框架集群的吞吐量; 文献[12-13]对基于多查询和状态管理进行了优化,研究了 Flink 的可扩展性等等。但是对于 Flink 使用过程中数据解析逻辑的控制研究相对较少。

文中主要从对 Flink 进行实时解析时逻辑的更改角度进行优化,通过使用流控制的方式,减少代码的开发量,提高 Flink 应用实时解析的效率。

2　实时计算架构

随着业务越来越复杂,需要采集和存储的数据越来越多,由于存在着不同的业务系统,日志的存储格式多种多样。为了对不同的日志进行解析,同时能够根据不同的需求将解析的数据输出到相应的存储空间,需要开发一套满足灵活地适配各种日志格式的数据解析架构,从而减少同类解析代码的开发,将不同的数据解析进行集中式的管理。文中基于队列和分布式流处理架构构建了大数据的实时采集计算和存储平台。数据处理架构如图 1 所示。

1.数据采集。

日志数据都是实时产生的,在采集的过程中,也是在不断生成的,因此数据采集模块需要完成实时采集。目前应用较多的有 Tcollector、Filebeat[14]等采集工具。

其中 Filebeat 具有两个较大的优势:

(1) 性能稳健。

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

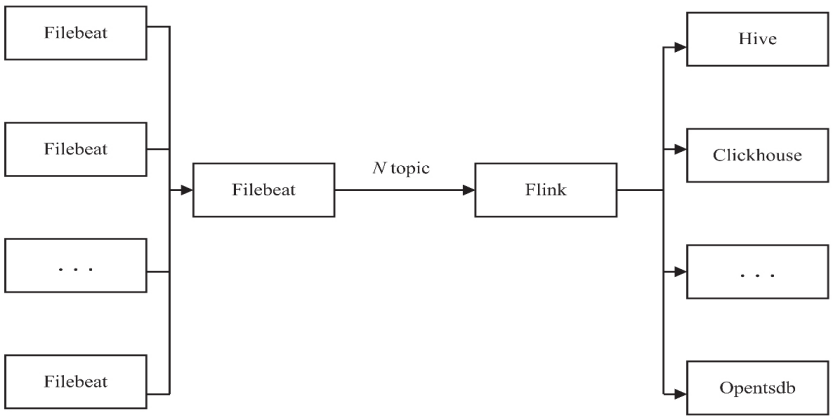

图 1 数据处理架构

无论什么样的应用都可能存在程序中断的情况，Filebeat 能够读取并转发日志行，如果出现中断，还会在一切恢复正常后，从中断前停止的位置继续开始。

(2) 部署简单。

Filebeat 内置有多种模块(Apache、System、MySQL 等等)，可以针对常见格式的日志大大简化收集、解析和可视化过程。

基于 Filebeat 的优势，在构建数据的采集平台时采用该服务进行日志的实时采集。在部署采集程序时，将不同的业务发送到不同的 topic 数据流中；通过 Filebeat 的配置文件实现数据采集的机器、日志文件、采集的路径、数据的输出端的配置。

2.数据传输。

数据传输采用 Kafka[15] 队列，每个 topic 队列作为一个单独的数据流，并与数据的采集和解析构成完整的数据处理流。除了采集业务数据的数据流，这里增加一个空流，用来进行流控制。Kafka 采集的各个数据流如图 2 所示。

Topic	# Partitions	# Brokers	Brokers Spread %	Brokers Skew %	Brokers Leader Skew %	# Replicas	Under Replicated %	Producer Message/Sec	Summed Recent Offsets
[redacted]	12	9	225	0	0	1	0	0.00	55851
[redacted]	24	9	225	0	0	1	0	1 326.13	53153799978
[redacted]	16	9	225	0	0	1	0	19.67	209476849
service01	1	1	25	0	0	1	0	0.00	0
service02	1	1	25	0	0	1	0	0.00	0
service03	1	1	25	0	0	1	0	0.00	0
service04	1	1	25	0	0	1	0	0.00	0
service05	1	1	25	0	0	1	0	0.00	0
temp	1	1	25	0	0	1	0	0.00	0

Showing 31 to 39 of 39 entries Previous 1 2 3 4 Next

图 2 采集的数据流

3.数据解析。

数据解析部分需要获取多种数据的配置，如系统配置、数据源配置、数据解析逻辑配置、数据存储配置、监控配置等。数据解析模块主要是通过使用 Flink 的各种算子组合完成业务数据解析逻辑的。该模块是整个实时采集计算和存储平台的核心部分，对数据的实时计算能力要求较高。该模块不仅需要完成对数据流的实时解析，同时还要支持对数据流解析的实时更改。例如一个流能够解析多个 topic 的数据，一个 topic 能够通过解析程序分流到不同的存储路径，对一个流能够实时地更改解析逻辑，而不需要重启。

4.数据存储。

日志数据不仅用来进行业务的分析，还需要对各种业务的指标进行监控，所以同一份日志的数据需要存储到不同的存储介质，因此数据流的输出结果也会有多种，如 hdfs、hive、clickhouse、opentsdb 等多个存储渠道。

3 自动化流控制算法

基于第 2 节介绍的实时计算架构，提出了使用更新算子的方式来改变数据流的解析逻辑。

3.1 流迭代算法

现在通用的数据流处理方式是流 stream1 处理完，将得到的结果作为 stream2 的输入，在 stream2 流的处理中完成对 stream1 的结果的处理，将 stream 流的解析通过不同的 map 逻辑依次处理，直到得到想要的输出

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

结果。解析逻辑如图 3 所示。

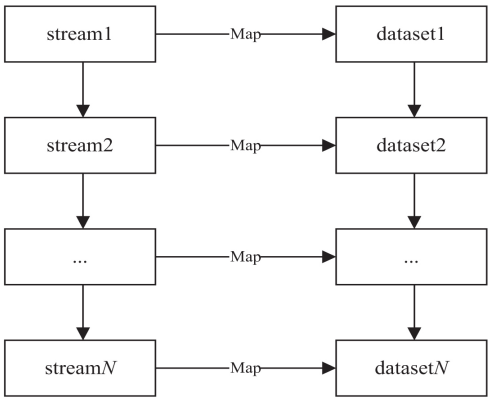

图 3　解析逻辑

而第 2 节介绍的实时计算架构,不再使用这种方式进行业务逻辑的处理,而是通过配置文件来实现,为了满足通过更新配置文件代替代码开发完成日志解析的需求,提出流迭代的算法,具体算法逻辑如下:

输入: 需要解析的数据流 stream。

输出: 解析结果。

Step1: 将每个要处理的数据流的名称通过 hashmap 进行存储,假设 < key, value > = < stream1, datastream1 >,同时将需要对数据流处理的算子存储到 list 中;

Step2: 按照对 datastream1 的流处理算子得到流处理结果 dataset1;

Step3: 更新 hashmap 中 stream1 的 value 值为 dataset1;

Step4: 遍历下一个需要处理的算子,直接读取 key = stream1 的 value 值,对 stream1 的 value 值执行相应的解析逻辑得到数据集 dataset2;

Step5: 更新 stream1 的 value 值为 dataset2;

Step6: 依次迭代对数据流处理的各个算子,直到完成所有的解析逻辑,最后结果依然保存在 stream1 中。

通过流迭代的方法,每次算子执行的时候都是对同一个 stream1 进行处理,只需要遍历定义好的算子即可,这样很多算子在不同的数据流解析中可以共用,不仅可以减少代码的开发量,还可以把开发的重点放在业务逻辑处理中,解析日志的程序开发变得更加简单。在算子中还可以添加复制的算子,将一个数据流复制成多个数据流,再针对不同的数据流配置不同的日志解析算子,实现分流的效果。

3.2　流控制算法

程序的重新启动会中断正在运行的解析逻辑,有些数据实时性要求较高,中间重启程序会造成一些数据的缺失。同时针对 3.1 介绍的流切换算法中的 stream 值更新的时候也需要实时地传入,因此文中设计了免更新、免重启的流控制算法。

输入: 解析算子 γ,算子 γ 是可以实现数据流选择和各种解析业务逻辑的配置,里面通过设置一个参数 source,实现对不同数据流的解析逻辑控制。

输出: 按照算子指定的执行逻辑输出结果。

Step1: 假设需要解析的数据流为 dataA,在现有需要解析的数据流中增加一个空的数据流 temp,该数据流开始时不存储任何数据,同时增加一个内部类的变量用来存储解析的算子 γ;

Step2: 在实时的代码解析逻辑中,增加一个对 temp 流的解析;

Step3: 在需要更新解析逻辑时,通过注入的方式将最新的解析逻辑注入到 temp 流中;

Step4: 通过解析 temp 流中的数据,获取针对当前数据流的解析逻辑,并更新为 γ 的值;

Step5: 再次解析数据流 dataA 的时候,就会使用最新的解析逻辑来处理数据,从而实现解析逻辑的实时控制。

算法的实现逻辑如图 4 所示,数据流处理主要分为数据的采集和解析,业务数据流主要是从各个业务

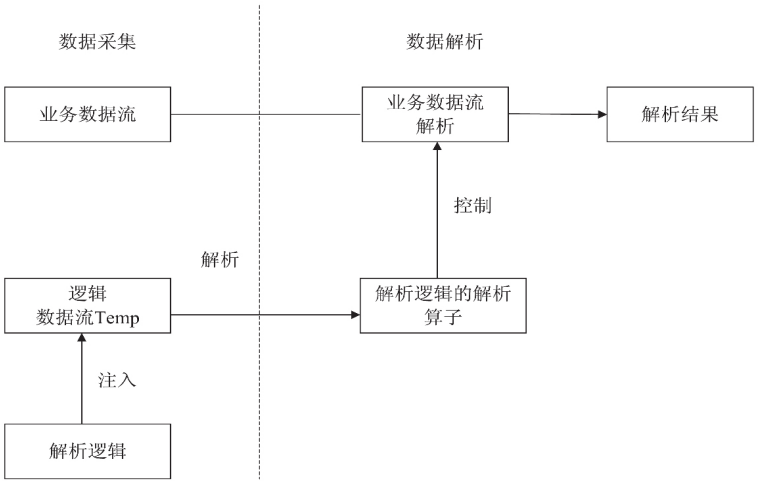

图 4　业务数据流处理流程

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

系统实时采集对应的数据，而逻辑数据流是在需要解析某个业务数据时，传入业务流对应的解析逻辑；在数据解析环节首先获取解析逻辑的解析算子，从而实现对业务数据流解析的控制。

3.3 temp 流中的解析算子 γ

注入 temp 数据流的解析算子 γ 配置如下：

```
#系统配置
#数据源
#业务逻辑( 任务解析)
#配置输出
#监控配置
```

通过 java 生成相应的文件，注入到 temp 数据流中。在整个架构中有一个控制类，该类通过读取 XML 文件，解析一个配置类，配置的成员包括系统配置类、数据源类、业务逻辑算子类、配置输出 sink 类等，其中算子类在实现的时候会继承一个基类，这样不同类型的算子都可以组成一个基类算子的 list 列表。

4 算法应用实例

以从 Kafka 队列接收数据，使用 Flink 解析，输出到 hdfs、Opentsdb 存储数据中为例，介绍该算法在具体的实时流计算框架中的应用。

4.1 配置文件实例

下面给出了注入 temp 的主要配置信息。

```
<? xml version="1.0" encoding="UTF-8"? >
<root>
<! —系统配置 -->
    <system>
        <jobname>service-to-hdfs</jobname>
    </system>
<! --配置数据源--><! —在该架构中使用的 kafka 队列作为传输通道-->
    <source>
        <kafka>
            <stream_name>stream1</stream_name>
            <topic>log-service</topic>
        </kafka>
    </source>
<! --配置解析逻辑 -->
<operator>
<! —第一个解析算子 map 算子，按照日志指定分隔符分隔-->
        <map stream_name="stream1">
            <json>
            <! —获取日志类型、采集机器 IP 地址、具体日志信息等内容，然后根据统一的分隔符“|;”进行拼接-->
            <getField alias="A">fields>ip</getField>
            <getField alias="B ">message</getField>
            <output connector="|;">A,B </output>
            </json>
<! —指定日志解析的分隔符“|;”，输出格式的分隔符为“;”-->
            <split separator="\|;" connector=",">
                <json field="3">
<getField alias="1" cutString="0-14">time</getField><! —截取时间串-->
                    <getField alias="2">info</getField>
                    <! —输出的拼接结果，按照 “;”分隔符输出拼接后的结果-->
                    <output connector=",">1,2 </output>
                    </json>
                </split>
            </map>
<! —第二个解析算子，对日志按照指定条件过滤，这里可实现字段个数的判断 -->
        <filter stream_name="stream1">
            <! —按照 “;”分隔符进行分隔，size 为划分后的字段个数，alias= ‘1’ 为可任意指定的名字-->
        <getField separator="," alias="1">size</getField>
            <int alias="2">24</int>
    <string alias="3">xxx01 </string>
            <judge>
                <basic condition="==" >1,2</basic>
    <or condition="==">1,3</or>
                <! —basic 为一个算子，判断 alias=1 和 2 的两个值是否相等，符合条件的数据流流入下一个处理逻辑，or 也是算子条件是 1 或者 3 成立-->
            </judge>
        </filter>
<! —第三个解析算子，自定义的解析算子，这里通过反射机制实现，class01 对应相应的类，method01 对应类中的方法 -->
            <custom stream_name="stream1" >class01 >method01 </custom>
<! —第四个解析算子，实现统计汇总的功能。flatMap 算子将字符串的字段转成 tuple 类型，设置时间窗口大小、和 sum 操作等不再展示->
            <flatMap stream_name="stream1">
                <int>1</int>
                <getField>all</getField>
            </flatMap>
            <keyBy stream_name="stream1">1</keyBy>
            <timeWindow stream_name="stream1">2</timeWindow>
            <sum stream_name="stream1" >0</sum>
<! —为支持后续处理算子，需要将 tuple 类型转成字符串->
            <tupleToStr stream_name="stream1" connector= "," >f0,f1</tupleToStr>
        <! --分流逻辑的实现，将一个流分成两个流一个写入
```

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

```
hdfs,一个到 Opentsdb -->
    <! 一下面的配置可以实现,将 stream1 流分成 stream2 和
stream3 流,还可以实现流筛选,这里不再显示。-->
  <sideOutput stream_name="stream1">
            <getStream sidOutputTag="stream2">
                <getField>all</getField>
            </getStream>
            <getStream sidOutputTag="stream3">
                <getField>all</getField>
            </getStream>
        </sideOutput>
  <! 一分流后的 stream2,stream3 流的逻辑处理类似,不再赘
述。->
      </operator>
    <! --配置输出-->
      <sink>
          <hdfs stream_name="stream2">
              <path>* * * * * * * * * * * * * * * * *
* * * </path>
          </hdfs>
          <http stream_name="stream3">
              <path>* * * * * * * * * * * * * * * * * *
</path>
          </http>
      </sink>
  <! --配置监控-->
      <monitor>
          <opentsdb>
              <URL></URL>
              <metric></metric>
              <tag>
                  <type></type>
              </tag>
          </opentsdb>
      </monitor>
  </root>
```

4.2 Flink 代码解析

Flink 是一种典型的分布式计算,对于内部类外的变量会在程序启动后存储在 master 节点上,且后续都不能改变,而对于内部类中的变量在每次代码执行时都会执行,因此采用 3.2 介绍的流控制方法在内部类中添加一个变量,用来存储解析算子,在对不同业务数据进行处理时,更新这个值就可以达到解析的目的。

伪代码逻辑如下:

```
创建一个临时变量 str=temp;
#解析数据流 DataA
{
#根据 str,实现 map 的解析
DataStream<String> hiveStream = datastream1. map( new map
(str));
}
#解析数据流 temp
{
Parse(temp_map)
输入新的解析算子 new_temp
If(Source=stream1){ str=new_temp;}
else{Return;}
```

4.3 实验对比

目前生产上共配置了 5 台 CPU16 核,内存为64 G 的服务器,搭建了实时解析架构平台,每秒的日志处理流量大概 15 W 左右,处理业务日志种类多达 50 个,随着业务的变动,实时平台的调整也会比较频繁。

完成一个在线运行的业务更改逻辑过程对比如下:

不使用文中提出的自动化流控制算法的业务处理过程:

(1) 从 Filebeat 采集的每条数据中获取需要的字段;

(2) 对指定的日志数据写代码进行解析,调试;

(3) 查看解析结果,是否满足业务的需求;

(4) 写入库的 sink 代码,调试;

(5) 查看入库的结果;

(6) 将代码打包、上传 jar 包到各台服务上;

(7) 重启各个进程,查看程序是否正常启动。

使用文中提出的自动化流控制算法的业务处理过程:

(1) 在配置文件中实现所有的逻辑;

(2) 查看解析和入库的结果;

(3) 将配置文件注入到 topic 流中;

(4) 自动生效,完成逻辑的更改。

消耗和用时结果对比如表 1 所示。

表 1　是否使用流处理的对比结果

是否使用自动化流控制算法	修改解析逻辑及测试	生产部署	延迟入库数据量
否	3 个小时	15 分钟	13 500 万条日志
是	0.5 个小时	1 分钟(执行一个注入命令即可)	900 万条日志

从实现的流程可以看到,使用文中提出的自动化流控制算法可以很大程度地减少代码的开发与测试,减少生产部署的工作量,最大程度地保证了生产数据的实时性。

5 结束语

通过 Filebeat、Kafka 队列和 Flink 流式处理架构,构建了一套实时数据流解析的平台,不仅能够针对大

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

量的数据进行实时解析，还能够满足实时更新解析逻辑的策略，极大地提高了生产数据的采集和分析效率。通过该实时解析架构平台，可以实现多流合并，单流分流，及业务逻辑实时更新等多个功能，提高了分布式流处理架构Flink的应用性能，为当前各个互联网公司复杂业务逻辑的大数据处理提供了解决方案，具有一定的现实意义。后续将会对Flink使用的资源做进一步的优化，提高数据解析平台的资源利用率。

参考文献：

[1] MIKA P.Flink: semantic web technology for the extraction and analysis of social networks [J].Journal of Web Semantics, 2005,3(2):211-223.

[2] ZAHARIA M, CHOWDHURY M, FRANKLIN M J,et al. Spark: cluster computing with working sets [C]//Proceedings of the 2nd USENIX conference on hot topics in cloud computing.Boston,MA: [s.n.],2010.

[3] 代明竹,高嵩峰.基于Hadoop、Spark及Flink大规模数据分析的性能评价 [J].中国电子科学研究院学报,2018,13(2):149-155.

[4] 赵 娟,程国钟.基于Hadoop、Storm、Samza、Spark及Flink大数据处理框架的比较研究 [J].信息系统工程,2017(6):117.

[5] 周志阳,陈 飞.大数据实时计算平台技术综述 [J].中国新通信,2017,19(4):47.

[6] CHINTAPALLI S,DAGIT D,EVANS B,et al.Benchmarking streaming computation engines: Storm, Flink and Spark streaming [C]//2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW).Chicago: IEEE,2016.

[7] 蔡鲲鹏.基于Flink平台的应用研究 [J].现代工业经济和信息化,2017,7(2):99-101.

[8] 麦冠华.基于Flink的实时轨迹运动模式检测 [D].杭州:浙江大学,2018.

[9] MARCIANI G,PIU M,PORRETTA M,et al.Real-time analysis of social networks leveraging the Flink framework [C]//ACM international conference on distributed & event-based systems.Irvine: ACM,2016.

[10] VERBITSKIY I,THAMSEN L,KAO O.When to use a distributed dataflow engine: evaluating the performance of Apache Flink [C]//Ubiquitous intelligence & computing,advanced & trusted computing, scalable computing & communications, cloud & big data computing,internet of people,& smart world congress.Toulouse: IEEE,2017.

[11] 李梓杨,于 炯,卞 琛,等.基于流网络的流式计算动态任务调度策略 [J].计算机应用,2018,38(9):2560-2567.

[12] SAHAL R,KHAFAGY M,OMARA F A E.Big data multi-query optimisation with apache Flink [J].International Journal of Web Engineering and Technology,2018,13(1):78-97.

[13] CARBONE P,EWEN S,FÓRA G,et al.State management in Apache Flink? [J].Proceedings of the VLDB Endowment, 2017,10(12):1718-1729.

[14] 翟雅荣,于金刚.基于Filebeat自动收集Kubernetes日志的分析系统 [J].计算机系统应用,2018,27(9):81-86.

[15] 王仲生.基于kafka消息队列的文本处理技术研究 [J].软件导刊:教育技术,2016,15(12):87-89.

(上接第65页)

of incomplete data [J].IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics,2001,31(5):735-744.

[6] FREY B J, DUECK D. Clustering by passing messages between data points [J].Science,2007,315(5814):972-976.

[7] 刘述昌,张忠林.基于中心向量的多级分类KNN算法研究 [J].计算机工程与科学,2017,39(9):1758-1764.

[8] 赵 亮,陈志奎,张清辰.基于分布式减法聚类的不完整数据填充算法 [J].小型微型计算机系统,2015,36(7):1409-1414.

[9] 冷泳林,陈志奎,张清辰,等.不完整大数据的分布式聚类填充算法 [J].计算机工程,2015,41(5):19-25.

[10] 梁家政,薛 质.网络数据归一化处理研究 [J].信息安全与通信保密,2010(7):47-48.

[11] 刘思谦,陈志奎,蒋昆佑,等.一种混杂的多核估计数据填充方法 [J].小型微型计算机系统,2017,38(7):1523-1527.

[12] SUN L, GUO C. Incremental affinity propagation clustering based on message passing [J].IEEE Transactions on Knowledge and Data Engineering,2014,26(11):2731-2744.

[13] YANG S,JIAN H,DING Z,et al.IKNN: informative K-nearest neighbor pattern classification [C]//The 11th European conference on principles and practice of knowledge discovery in databases.Warsaw: [s.n.],2007.

[14] ZHANG S,LI X,ZONG M,et al.Efficient kNN classification with different numbers of nearest neighbors [J].IEEE Transactions on Neural Networks & Learning Systems,2018,29(5):1774-1785.

[15] LU W,ZHANG L,LIU X,et al.A human-computer cooperation fuzzy c-means clustering with interval-valued weights [J].International Journal of Intelligent Systems,2015,30(2):81-98.

(C)1994-2020 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net