

KYLIN-5417 实时自定义解析器Dev Design

背景描述

实时功能默认的Kafka消息解析器，对于多层嵌套的Kafka消息，解析器的字段命名是固定的，不能重命名成具有业务含义的名称，缺乏灵活性。有的Kafka消息不一定是以json形式发送的，可能是逗号分隔的字符串，也可能是竖线分隔的字符串，自定义Kafka消息解析器也能够支持这类自定义的Kafka消息的解析。

本功能在默认的 Kafka 解析器的基础上，修改扩展点，支持客户基于这个扩展点，自助解析消息解析器。

Dev Design

解析器抽象类设计

定义了一个抽象类 `AbstractDataParser`

用户可以自定义Parser类，继承 `AbstractDataParser` ，覆盖 `parse` 方法，用于处理单条kafka消息。

`AbstractDataParser` 会统一把用户返回 `resultMap` 处理成 `Map<String, Object>` 数据以供后续使用。

- 用户如有全局初始化的参数可以在构造函数中初始化
- 外部统一调用`process(I Input)`方法获取解析后数据

目前泛型输入数据类型只支持ByteBuffer。

元数据结构设计

DataParserInfo

```
1  resource: org.apache.kylin.parser.JsonDataParser1
2  {
3    "uuid" : "ac3fd5e4-94a4-4ca4-ae4b-307c3c957f37",
4    "last_modified" : -1,
5    "create_time" : 1622187324509,
6    "version" : "4.0.0.0",
7    "project": "streaming_test",
8    // parser name
9    "class_name": "org.apache.kylin.parser.JsonDataParser1",
10   // parser所属jar name
11   "jar_name": "custom_parser1.jar",
12   // 当前parser都有哪些table在使用
13   "streaming_tables": [
14     "DEFAULT.SSB_STREAMING",
```

```
15     "DEFAULT.SSB_TOPIC",
16     "SSB.P_LINEORDER",
17     "SSB.P_LINEORDER_STR",
18     "SSB.P_LINEORDER_STREAMING"
19 ]
20 }
```

每个project都有一个默认的Parser，结构如下

```
1 该Parser无法被删除
2 {
3   "uuid" : "ac3fd5e4-94a4-4ca4-ae4b-307c3c957f37",
4   "last_modified" : -1,
5   "create_time" : 1622187324509,
6   "version" : "4.0.0.0",
7   "project": "streaming_test",
8   "class_name": "org.apache.kylin.parser.TimedJsonStreamParser",
9   "jar_path": "default",
10  "streaming_tables": []
11 }
```

Jar管理元数据设计

```
1 resourcename: STREAMING_CUSTOM_PARSER_custom_parser_test.jar
2
3 {
4   "uuid" : "ac3fd5e4-999a-4ca4-ae4b-307c3c957f37",
5   "last_modified" : -1,
6   "create_time" : 1622187328509,
7   "version" : "4.0.0.0",
8   "project": "streaming_test",
9   // jar 文件名
10  "jar_name": "custom_parser_test.jar",
11  // jar 文件路径
12  "jar_path": "/streaming_test/jar/custom_parser_test.jar",
13  // 枚举类，以后如有其他模块需要使用Jar管理，可以新增枚举类
14  "jar_type": "STREAMING_CUSTOM_PARSER"
15 }
```

Jar包加载通用类设计

- 自定义ClassLoader: **ParserClassLoader**
- ClassLoader管理类: **ParserClassLoaderState**
- Jar加载: **AddToClassPathAction**
- 工具类: **ClassLoaderUtils**

ParserClassLoader

自定义ClassLoader，继承自URLClassLoader

AddToClassPathAction

参考Hive的UDF Jar动态加载

```
org.apache.hadoop.hive ql.exec.AddToClassPathAction
```

该类用于将一组Jar URL集合加载到ClassPath，并将传入的ParentClassLoader转换成

```
ParserClassLoader
```

ParserClassLoaderState

单例对象，用于管理不同Project的 `ParserClassLoader`

单例对象初始化时会使用 `Thread.currentThread().getContextClassLoader()` 传入空URL集合，生成 `ParserClassLoader`。

1. 后续所有需要使用ClassLoader的地方都需要从 `ParserClassLoaderState` Get。
2. 所有load/unload jar之后生成的新 `ParserClassLoader` 都需要在 `ParserClassLoaderState` Set

Jar 存储

- 考虑到读集群存储了索引数据，变动成本高，所以选择将Jar保存在读集群HDFS中
- 在多活环境下，各个KE共享元数据，并且都可以拿到读集群的HDFS地址
- 在读写分离模式下，实时任务启动时通过 `addJar` 的形式将读集群的Jar load 到Spark Application

Jar存储在HDFS的路径设计

```
1 {kylin.env.hdfs-working-dir}/{MetadataUrlPrefix}/custom/jars/{project}/{JarType}
2
3 {JarType}: 标记当前Jar使用的范围，目前只有 STREAMING_CUSTOM_PARSER 一种。
4 后面有扩展别的用途可以新增Type
5
6
```

SDK 设计

- 新增Module `kylin-streaming-sdk`
- 以下两个Module依赖于 `kylin-streaming-sdk`
 - `kylin-assembly` 实时任务需要使用解析器抽象类
 - `kylin-streaming` 实时的数据解析需要使用解析器
- 用户使用SDK步骤
 - a. 只需将 `kylin-streaming-sdk.jar` 引入自建项目的lib中
 - b. 新建解析器类， `extends AbstractDataParser<ByteBuffer>`
 - c. `Override parser(ByteBuffer input)` 方法即可
 - d. 如需有初始化动作需要在实例化解析器类时完成，请在无参构造中完成

- e. 如需在每条数据解析前，有初始化动作，请 `Override before()`
- f. 如需在处理数据时，明确表示当前数据有误，需要忽略不进行构建，请在 `parse()` 方法中抛出异常
- g. 如需在每条数据处理后对数据做检查，请 `Override after()`
- h. 新建如下文件

i. `src/main/resources/META-INF/services/org.apache.kylin.parser.AbstractDataParser;`
写入自定义解析器类的全路径。

- 打Jar包
- 通过Upload Jar API，上传解析器Jar，上传成功即可使用

SDK使用全流程

[目实时自定义解析器使用手册](#)

新增参数

| 参数 | 默认值 | 参数说明 | |
|-------------------------------------|------|----------------|----|
| kylin.streaming.custom-parser-limit | 50 | 项目内可上传解析器上限。 | 系统 |
| kylin.streaming.custom-jar-size | 20mb | 单个 jar 上传大小上限。 | 系统 |

报错场景

| Error Code | MSG-CN | |
|--------------|--|-----------------------------------|
| KE-010042201 | 无效的文件格式。请使用 Jar 格式文件。 | The file format i |
| KE-010042202 | 文件 “%s” 已存在。请检查后重试。 | The file "%s" al |
| KE-010042203 | 当前无法连接 HDFS。请检查网络后重试。 | Can't connect to |
| KE-010042204 | 文件 “%s” 中检测不到任何消息解析器。请添加后重试。 | Can't detect any |
| KE-010042205 | 文件 “%s” 中的消息解析器 “%s” 已存在。请检查后重试。 | The message pa |
| KE-010042206 | 无法删除，当前文件中的解析器正在被表 “%s” 调用。请解除后重试。 | Can't delete the |
| KE-010042207 | 无法删除，当前解析器正在被表 “%s” 调用。请解除后重试。 | Can't delete the |
| KE-010042208 | 无效的解析列名。请以字母开头，并只使用字母、数字、下划线。 | The parsed colu |
| KE-010042209 | 无法上传，解析器数量超限，当前系统内解析器数量：“%s”，新增解析器数量：“%s”，解析器数量上限：“%s”。请检查后重试。 | Unable to uploa number of newl |

| | | |
|--------------|--|-------------------|
| KE-010042210 | 无法删除默认消息解析器。 | Can't delete the |
| KE-010042211 | 加载Jar文件失败。请检查后重试。 | Failed to load th |
| KE-010042212 | 解析器 “%s” 不存在。 | Parser "%s" doe |
| KE-010042213 | 上传Jar文件超出大小限制 “%s” 。 | Uploading Jar fi |
| KE-010042214 | Jar文件 “%s” 不存在。 | Jar "%s" does n |
| KE-010042215 | 解析器 “%s” 已存在。 | Parser "%s" alre |
| KE-010042216 | Jar文件 “%s” 已存在。 | Jar "%s" alread |
| KE-010035202 | 使用解析器 “%s” 解析Topic “%s” 的消息时发生异常，请检查后重试。 | An exception oc |

流程

Upload Jar

1. 读取Jar文件名，元数据查找改Jar名是否重复。重复则报错
2. 上传Jar到HDFS: /{working-dir}/{metadata_identifier}/custom/jars/{project}/{JarType}/
3. Load HDFS Jar
4. 获取新增Parser Class集合，元数据查找Class是否重复，重复则报错然后UnLoad。
5. Load Jar 到 Meta & ClassPath

Remove Jar

1. 元数据查找Jar的所有Parser是否有Table在使用，有则报错
2. 无引用则卸载Jar，删除Meta，删除ClassPath

Remove Parser

1. 元数据查找Parser是否有Table在使用，有则报错
2. 无引用则在元数据层面删除Parser，Jar包内的其他Parser还可以使用

Get Parser

获取Project中元数据中缓存的所有Parser

API设计

1. Upload Jar

- 检查Jar文件是否合规
 - ".jar"结尾
 - 元数据中是否有同名Jar
- 上传Jar到HDFS `/{working-dir}/{metadata_identifier}/custom/jars/{project}/{JarType}/`
- 加载Jar并获得Jar内的解析器类，校验元数据中是否存在同名解析器，校验解析器数量是否超过 `kylin.streaming.custom-parser-limit`（默认50）。
- 加载Jar元数据
- 加载解析器元数据
- 返回成功加载的解析器List

```
1 curl --location --request POST 'http://localhost:7070/kylin/api/custom/jar' \  
2 --header 'Accept: application/vnd.apache.kylin-v4+json' \  
3 --header 'Accept-Encoding: cn' \  
4 --header 'Accept-Language: gzip, deflate, br' \  
5 --header 'Content-Type: multipart/form-data' \  
6 --header 'Authorization: Basic QURNSU46SGpsaGpscXFacXExOTk2' \  
7 --header 'Cookie:  
2668b148e20b2b3001c3ac8230655f3e7667197a5039dcbf102f8e8fb4217fe8=NTM4MGEwNDQtNm  
JhYy00MGVlLTg1YWItMTM1MzMxYmQyNTFi' \  
8 --form 'file=@"/Users/jiale.he/Desktop/解析器/my_parser-custom_2.jar"' \  
9 --form 'project="Test4"' \  
10 --form 'jar_type="STREAMING_CUSTOM_PARSER"'
```

Request参数说明

1. file [必填]: 上传的Jar路径
2. project [必填]: 项目名
3. jar_type [必填]: Jar包用途分类，目前只有 `STREAMING_CUSTOM_PARSER` 用于自定义解析器

Response 参数说明

```
1 {  
2   "code": "000",  
3   "data": [  
4     // 本次上传Jar成功加载的解析器列表  
5     "org.apache.kylin.parser.JsonDataParser1",  
6     "org.apache.kylin.parser.CsvDataParser2",  
7     "org.apache.kylin.parser.CustomDataParser2"  
8   ],  
9   "msg": ""  
10 }
```

2. Delete Jar

- 检查要删除的Jar内的所有解析器是否有表引用
- 删除Jar元数据、删除解析器元数据、重置ClassLoader


```

1 curl --location --request DELETE 'http://localhost:7070/kylin/api/custom/jar?
  project=Test4&jar_name=my_parser-custom_2.jar&jar_type=STREAMING_CUSTOM_PARSER'
  \
2 --header 'Accept: application/vnd.apache.kylin-v4+json' \
3 --header 'Accept-Encoding: cn' \
4 --header 'Accept-Language: gzip, deflate, br' \
5 --header 'Content-Type: application/json;charset=utf-8' \
6 --header 'Authorization: Basic QURNSU46SGpsaGpscXFacXEx0Tk2' \
7 --header 'Cookie:
  2668b148e20b2b3001c3ac8230655f3e7667197a5039dcbf102f8e8fb4217fe8=OGI20GRhNjQtYz
  BjMy00NDFlLTk4MTEtZTd1ZTcwM2Q2NjJm' \
8 --data-raw ''

```

Request参数说明

1. project [必填]: 项目名
2. jar_name [必填]: 要删除的目标Jar文件名
3. jar_type [必填]: Jar包用途分类, 目前只有 `STREAMING_CUSTOM_PARSER` 用于自定义解析器

Response参数说明

```

1 {
2   "code": "000",
3   // 成功删除的Jar文件名
4   "data": "my_parser-custom_2.jar",
5   "msg": ""
6 }

```

3. Get Parsers

- 从元数据中获取当前项目的所有Parser

```

1 curl --location --request GET 'http://localhost:7070/kylin/api/kafka/parsers?
  project=Test4' \
2 --header 'Accept: application/vnd.apache.kylin-v4+json' \
3 --header 'Accept-Encoding: cn' \
4 --header 'Accept-Language: gzip, deflate, br' \
5 --header 'Content-Type: application/json;charset=utf-8' \
6 --header 'Authorization: Basic QURNSU46SGpsaGpscXFacXEx0Tk2' \
7 --header 'Cookie:
  2668b148e20b2b3001c3ac8230655f3e7667197a5039dcbf102f8e8fb4217fe8=OGI20GRhNjQtYz
  BjMy00NDFlLTk4MTEtZTd1ZTcwM2Q2NjJm' \
8 --data-raw ''

```

Request参数说明

1. project [必填]: 项目名

Response参数说明

```

1 {
2   "code": "000",

```

```
3     "data": [  
4         // 当前项目内存在的解析器List  
5         "org.apache.kylin.parser.CsvDataParser1",  
6         "org.apache.kylin.parser.CustomDataParser1",  
7         "org.apache.kylin.parser.TimedJsonStreamParser"  
8     ],  
9     "msg": ""  
10 }
```

4. Delete Parser

- 检查要删除的解析器是否有表引用
- 删除解析器元数据

```
1 curl --location --request DELETE 'http://localhost:7070/kylin/api/kafka/parser?p  
2 --header 'Accept: application/vnd.apache.kylin-v4+json' \  
3 --header 'Accept-Encoding: cn' \  
4 --header 'Accept-Language: gzip, deflate, br' \  
5 --header 'Content-Type: application/json;charset=utf-8' \  
6 --header 'Authorization: Basic QURNSU46SGpsaGpscXFACXExOTk2' \  
7 --header 'Cookie: 2668b148e20b2b3001c3ac8230655f3e7667197a5039dcbf102f8e8fb4217f  
8 --data-raw ''
```

Request参数说明

1. project [必填]: 项目名
2. class_name [必填]: 解析器名

Response参数说明

```
1 {  
2     "code": "000",  
3     // 被删除的解析器名  
4     "data": "org.apache.kylin.parser.CsvDataParser1",  
5     "msg": ""  
6 }
```

注意

项目隔离

jar包应项目隔离。

Project 与 Project 之间无法互相获取对方jar内的解析器。

想要使用需重新在新的project中上传jar

目前只支持数据 1 - 1

只支持数据1进1出，不支持1进n出

解析器复杂度

由于自定义解析器决定了实时表的schema

所以在实时构建任务中，kafka数据流入的第一时间就会经过解析器。如果解析器复杂度过高，或者代码质量有问题，在一定程度上会影响实时构建任务效率。