Hive UDF

Task 1: Code your function

1. Simple UDF

   Purpose:

   One primitive input and one output.

   Example:

   Convert an age to age group information

   Coding Approach:

   Create a class which extends org.apache.hadoop.hive.ql.exec.UDF

   Implement evaluate method

Task 2: Deploy it temporarily

1. Add JAR to classpath
2. Create temporary function
3. Use it

```
hive (default)> add JAR /home/s_kante/IdeaProjects/HiveUDF/out/artifacts/HiveUDF_jar/HiveUDF.jar;
Added [/home/s_kante/IdeaProjects/HiveUDF/out/artifacts/HiveUDF_jar/HiveUDF.jar] to class path
Added resources: [/home/s_kante/IdeaProjects/HiveUDF/out/artifacts/HiveUDF_jar/HiveUDF.jar]
hive (default)> create temporary function isaccepted as 'udf.IsAcceptedNew';
OK
Time taken: 0.005 seconds
hive (default)> select isaccepted(result, cast(5.6 as double), cast(7.0 as double)) from mydb.table2;
OK
_c0
false
Time taken: 4.283 seconds, Fetched: 1 row(s)
hive (default)>
```

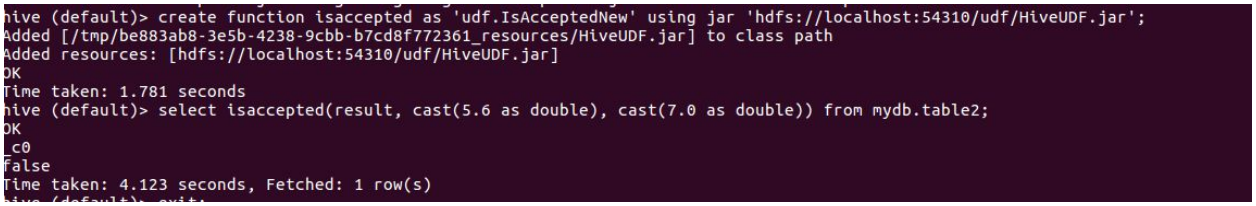
Or just append it into .hiverc file

Following is the content of a sample .hiverc file. It should be within conf directory of hive. If not present then you can create one.

set hive.cli.print.header=true;
set hive.cli.print.current.db=true;
add JAR /home/s_kante/IdeaProjects/HiveUDF/out/artifacts/HiveUDF_jar/HiveUDF.jar;
create temporary function isaccepted as 'udf.IsAcceptedNew';

Task 3: Deploy it permanently

1. Copy JAR file to hdfs file system
2. Register function



Complex data types in Hive

1. **Array**

   create table result(student_id int, bands array<double>) row format delimited fields terminated by '|' collection items terminated by ','

   insert into result select 10 , array(cast(4.5 as double),cast(6.7 as double));

   hive (mydb)> *select * from result;*
   OK
   result.student_id result.bands
   10 [4.5,6.7]
   Time taken: 0.662 seconds, Fetched: 1 row(s)

   hive (mydb)> *select student_id, bands[0] from result;*
   OK
   student_id _c1
   10 4.5
   Time taken: 0.821 seconds, Fetched: 1 row(s)

   Hdfs view
   hduser@shyam:/usr/local/hadoop/etc/hadoop$ *hdfs dfs -cat /user/hive/warehouse/mydb.db/result/000000_0*

   10|4.5,6.7

Hive UDF

Task 1: Code your function

2. User Defined Tabular Function (UDTF)

   Purpose:

   UDTF takes single record as input and generates multiple records in output.

   Example:

   Generate combination of transaction id and product id for a given transaction with all products flattened in single record.

   Coding Approach:

   Create a class which extends
   org.apache.hadoop.hive.ql.udf.generic.GenericUDTF
   Define methods

   initialize: will return the structure information of output record
   process: will be called on each new record
   close: any cleanup tasks to be carried out

3. User Defined Aggregate Function (UDAF) - Simple

   Purpose:

   UDAF takes multiple records with primitive data types as input to generate single record with primitive data type as output.

   Example:

   For a given IELTS bands in denormalized form, decide if student has cleared the exam or not

   Coding Approach:

   Create a class which extends *org.apache.hadoop.hive.ql.exec.UDAF*

   Create a subclass within that class which implements
   org.apache.hadoop.hive.ql.exec.UDAFEvaluator

   Define methods

   init: initialize variables
   iterate: Will be called for each record
   terminatePartial: how to behave when process completes with partial result on one node
   merge: to merge two partial results
   terminate: finally output the result

Debugging Hive CLI

hive -hiveconf hive.log.file=debug_hive_20180403.log -hiveconf hive.log.dir=/tmp/hivedebug/ -hiveconf hive.root.logger=DEBUG,DRFA

Complex UDF

Purpose:

Extended version of UDF class, which allows to deal with complex types as in List, Map, Struct as input and output. It also supports nested types for example List<List<>> with variable number of arguments.

Example:

For a given list of IELTS bands, decide if the score is acceptable or not

Coding Approach:

Three abstract methods that we need to implement

abstract String getDisplayString(String[] children)

Get the String to be displayed in explain.

abstract ObjectInspector initialize(ObjectInspector[] arguments)

called once, before any evaluate() calls. You receive an array of object inspectors that represent the arguments of the function this is where you validate that the function is receiving the correct argument types, and the correct number of arguments.

It returns ObjectInspector for the return type.

ObjectInspector:

Allows to look into the internal structure of complex data types. For example List, Map, Struct

abstract Object evaluate(GenericUDF.DeferredObject[] arguments);

This is similar to evaluate method of the simple API. It takes the actual arguments and returns the result

User Defined Aggregate Function (UDAF) - Complex

Purpose:

UDAF takes multiple records with primitive/complex data type as input to generate single record with primitive/complex data type as output.

Example:

For a given IELTS bands in denormalized form, decide if student has cleared the exam or not

Coding Approach:

Create a class which extends

org.apache.hadoop.hive.ql.udf.generic.AbstractGenericUDAFResolver

Create a subclass within that class which implements *org.apache.hadoop.hive.ql.udf.generic.GenericUDAFEvaluator*

Create a subclass within inner class which implements *AggregationBuffer* and works as buffer to hold temporary result

Define methods within most outer class

getEvaluator: Check on input data types and return corresponding instance of evaluator class based on input parameters

Define methods within inner class working as Evaluator

init: Initialization

iterate: Will be called for each record

getNewAggregationBuffer: return new aggregate buffer instance

reset: reset aggregate buffer

terminatePartial: how to behave when process completes with partial result on one node

merge: to merge two partial results

terminate: finally output the result

Complex data types in Hive

Map

create table resultMap(student_id int, bands map<string,double>) row format delimited fields terminated by '|' collection items terminated by ',' map keys terminated by ':';

Input file

hduser@shyam:~$ *cat tempfile*

10|reading:4.5,speaking:6.7,listening:7.5,writing:7.0
20|reading:5.5,speaking:6.5,listening:6.5,writing:8.0

hduser@shyam:~$ *hdfs dfs -copyFromLocal tempfile /user/hive/warehouse/mydb.db/resultmap/*

hive (mydb)> *select * from resultMap;*
OK
resultmap.student_id resultmap.bands
10 {"reading":4.5,"speaking":6.7,"listening":7.5,"writing":7.0}
20 {"reading":5.5,"speaking":6.5,"listening":6.5,"writing":8.0}

Time taken: 0.126 seconds, Fetched: 2 row(s)

hive (mydb)> *select student_id, bands["listening"] from resultMap;*
OK
student_id _c1
10 7.5
20 6.5
Time taken: 0.107 seconds, Fetched: 2 row(s)

Struct

create table rider(name string, age int, vehicle_conf struct<reg_no:string, top_speed:int, cc:int, brand:string>) row format delimited fields terminated by '|' collection items terminated by ',';

hduser@shyam:~$ *cat tempfile*

Sukhajinder|27|QC123,250,600,YAMAHA
Cesar|35|ON123,300,700,HONDA

hduser@shyam:~$ *hdfs dfs -copyFromLocal tempfile /user/hive/warehouse/mydb.db/rider/*

hive (mydb)> *select * from rider;*
OK
rider.name rider.age rider.vehicle_conf
Sukhajinder 27 {"reg_no":"QC123","top_speed":250,"cc":600,"brand":"YAMAHA"}
Cesar 35 {"reg_no":"ON123","top_speed":300,"cc":700,"brand":"HONDA"}
Time taken: 0.119 seconds, Fetched: 2 row(s)

hive (mydb)> *select vehicle_conf.brand from rider;*
OK
brand
YAMAHA
HONDA
Time taken: 0.111 seconds, Fetched: 2 row(s)

Read XML file in Hive Table

Create TABLE xmltable(xmldata string) STORED AS TEXTFILE;

hduser@shyam:~$ *cat tempfile*

<dependency><groupId>org.apache.hive</groupId><artifactId>hive-exec</artifactId><version>0.8.0</version></dependency>
<dependency><groupId>org.apache.hadoop</groupId><artifactId>hadoop-core</artifactId><version>1.2.1</version></dependency>
<dependency><groupId>junit</groupId><artifactId>junit</artifactId><version>4.5</version><scope>test</scope></dependency>

hduser@shyam:~$ *hdfs dfs -copyFromLocal tempfile /user/hive/warehouse/mydb.db/xmltable/*

hive (mydb)> *select xpath(xmldata, 'dependency/groupId/text()') from xmltable;*
OK
_c0
["org.apache.hive"]
["org.apache.hadoop"]
["junit"]
Time taken: 0.109 seconds, Fetched: 3 row(s)

Reference:

https://blog.matthewrathbone.com/2013/08/10/guide-to-writing-hive-udfs.html
https://cwiki.apache.org/confluence/display/Hive/GenericUDAFCaseStudy
https://cwiki.apache.org/confluence/display/Hive/GenericUDAFCaseStudy#GenericUDAFCaseStudy-WritingGenericUDAFs:ATutorial
https://community.hortonworks.com/content/supportkb/150214/how-to-enable-debug-hive-cli-logging.html

Hive Serde:
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-RowFormat,StorageFormat,andSerDe
https://cwiki.apache.org/confluence/display/Hive/DeveloperGuide#DeveloperGuide-CodeOrganizationandaBriefArchitecture

https://stackoverflow.com/questions/24607685/loading-xml-data-into-hive-table-org-apache-hadoop-hive-ql-metadata-hiveexcepti

Fun to Learn:
https://stackoverflow.com/questions/20208696/hadoop-restart-datanode-and-tasktracker