**Transferring data from Oracle Golden Gate to Impala**

The aim of the integration is capture transactions in Oracle databases (source) and apply it to an Impala database (target). The idea is replicate the source database in almost real time or in a very short time.

**Architecture**

The architecture can be divided in two systems: source and target. The source system is a machine where are the Oracle source database, the Golden Gate instance and the Impala loader. On the other side, we have the target system which involves a cluster a machines running Impala and HDFS.

The deployment of this architecture can be shown in the image below.

**Oracle database**

Source

Log

Trail

Adapter for Flat Files

Flat Files

**Oracle Golden Gate**

**Source machine**

**Target cluster**

Node 1

NameNode

Impala layer

HDFS

layer

Node 2

DataNode

Node N

DataNode

**…**

New

Flat files

SQL: creates and inserts

**How it works**

On the Golden Gate side, we use the adapter for Flat Files which generates CSV files with the transactional data. The generated files are transferred to the file system (HDFS) that Impala uses. An implemented loader check periodically if there is new data to transfer, and if there is new data, the loader follow several steps:

1. Move local new data to HDFS.
2. Create staging (external) table in Impala with the new data located in HDFS.
3. Insert all data of the staging table into final Impala table.
4. Delete staging table.
5. Delete new local data.

A flow diagram has been created to understand better how the loader works.

No

Yes

Copy local new data to HDFS

Create Impala staging table

Insert data to final table

Delete staging data

Start

New flat

files?

Wait N seconds

**Configuration and installation**

On one hand, we should configure Oracle Golden Gate in order to produce the Flat Files that the loader will read. On the other hand, we should deploy and configure the loader to import data into target Impala table.

* Installation and configuration of the adapter for Flat Files

(by Lorena)

* Deployment and configuration of the Impala loader

The loader is a custom client of HDFS and Impala that reads the flat files generated by Golden Gate and inserts that data into Impala. We must follow some steps to get a functional loader. That commands must be executed with the same user that Oracle Golgen Gate uses (e.g. oracle)

1. Clone and build the project where the loader have been implemented.

$> cd <ogg\_directory>

$> git clone https://github.com/dlanza1/ogg-impala.git

$> cd ogg-impala

$> mvn install

1. Copy JAR with dependencies to a new directory located where OGG is installed.

$> cd <ogg\_directory>

$> mkdir impala-loader

$> cp ogg-impala/target ogg-impala-\*-jar-with-dependencies.jar impala-loader

1. Create properties file for logging

$> cd <ogg\_directory>/impala-loader

$> nano log4j.properties

# An example of this file could be:

# Root logger option

log4j.rootLogger=INFO, stdout

# Redirect log messages to console

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

1. Create parameters file for configure the loader.

$> cd <ogg\_directory>/impala-loader

$> nano config.params

# An example of this file could be:

impala.host = itrac925.cern.ch

ogg.data.folder = ./out

ogg.control.file.name = GGUSER.DATA\_NUMERICcontrol

hdfs.staging.directory = ogg/staging

1. Create configuration file for HDFS client.

$> cd <ogg\_directory>/impala-loader

$> nano core-site.xml

# An example of this file could be:

<?xml version="1.0" encoding="UTF-8"?>

<!--Autogenerated by Cloudera Manager-->

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://itrac925.cern.ch:8020</value>

<description>NameNode host and port</description>

</property>

</configuration>

Current state of impala-loader directory must be:

$> ls -l <ogg\_directory>/impala-loader

config.params

core-site.xml

log4j.properties

ogg-impala-\*-jar-with-dependencies.jar

Now we should start the loader or in foreground (first point) or in background (second point).

* 1. Build CLASSPATH and start loader in foreground.

$> cd <ogg\_directory>

$> export CLASSPATH=./impala-loader/:./impala-loader/\*

$> java ch.cern.impala.ogg.datapump.ImpalaDataLoader ./impala-loader/config.params

Output should be similar to:

2015-04-20 13:49:23 INFO ImpalaDataLoader:22 - inicializing loader (properties file = ./impala/config.params)

2015-04-20 13:49:23 INFO ImpalaDataLoader:26 - reading control data from ./out/GGUSER.DATA\_NUMERICcontrol

2015-04-20 13:49:23 WARN PropertiesE:71 - the number of seconds between batches has been set to the default value (10 seconds)

2015-04-20 13:49:23 INFO ImpalaDataLoader:47 - there is no data to process

2015-04-20 13:49:33 INFO ImpalaDataLoader:47 - there is no data to process

* 1. Run loader in background.

$> cd <ogg\_directory>

$> export CLASSPATH=./impala-loader/:./impala-loader/\*

$> nohup java ch.cern.impala.ogg.datapump.ImpalaDataLoader ./impala-loader/config.params &

# To check the putput of the command

$> cat nohup.out

# It will show the some output that we run in foreground