Creating a Data Reservoir with Oracle GoldenGate

by Mark Rittman

Replicate and stream data from Oracle Database 12c to Oracle Big Data Appliance.

Many organizations are looking to extend their Oracle data warehouses with Hadoop and NoSQL technology, adding Oracle Big Data Appliance engineered systems to their existing Oracle Exadata Database Machines to create what has been termed a "data reservoir." Hadoop- and NoSQL-based data reservoirs built on Oracle Big Data Appliance extend the storage capacity of an Oracle Exadata–based data warehouse. They can be used to hold unstructured and semistructured data sets and are often used as an initial landing and preprocessing area in a data warehouse.

Typically, data loading into Oracle Big Data Appliance occurs in the form of real-time streams of data taken from file and event sources. Oracle GoldenGate for Big Data 12c can extend this capability to transactional data loading directly into Apache Hive and Hadoop Distributed File System (HDFS) data structures. Oracle GoldenGate can also interface with Apache Flume, the Oracle Big Data Appliance's standard real-time ingestion service, to provide a single service that streams incoming real-time data to a data reservoir.

In this article, I'll look at what's involved in configuring Oracle GoldenGate for Big Data 12c along with Oracle GoldenGate 12c for Oracle Database to capture database transactions and replicate them to a Hadoop-based data reservoir. You can run through the steps in this article by using Oracle Big Data Lite Virtual Machine, which you can download from Oracle Technology Network. Oracle GoldenGate for Big Data 12c and Oracle GoldenGate 12c (for Oracle Database) have already been installed on the virtual machine, along with Oracle Database 12c, providing an environment similar to what you'd find on an Oracle Big Data Appliance but without the need to install the individual software components I'll be demonstrating in this article.

How Oracle GoldenGate Streams Database Transactions into a Hadoop Environment

Oracle GoldenGate can be used to capture and transport database transactions between heterogeneous environments running on distributed server platforms. For example, you can capture database transactions from a MySQL database running on a Linux server and transport them to Oracle Database running on Oracle Exadata. Oracle GoldenGate uses a decoupled architecture, where a capture or extract process on the source database server monitors database logs and writes activity data to a trail file, which is then routed to target platforms in a compressed and encrypted form by the pump process. Next, a collector process on the target platform(s) takes this trail file data and passes it to the replicate process, which then writes the transactions to the target platform.

When you're working with sources of big data, Oracle GoldenGate for Big Data 12c uses the same capture, trail file, and pump processes to collect transaction activity on the source database server. But in this case, Oracle GoldenGate Adapter for Apache Flume then takes log activity from the trail file and sends it via an Apache Flume channel to a Flume agent in Avro remote procedure call (RPC) message format. The Flume agent receiving these flume events then writes the database transaction activity and the replicated data to files in the HDFS file system, as shown in **Figure 1**. This information is then used as the datasource for an Apache Hive table. Configuration files provide parameter values for the pump, Oracle GoldenGate Adapter for Apache Flume, and Apache Flume processes, with all configuration performed with the Oracle GoldenGate software command-line interface (GGSCI) shell. Note that in this example, all the components are contained within the same virtual machine environment, so you won't be using the pump process, because Oracle GoldenGate Adapter for Apache Flume can read directly from the trail file produced by the database extract process.

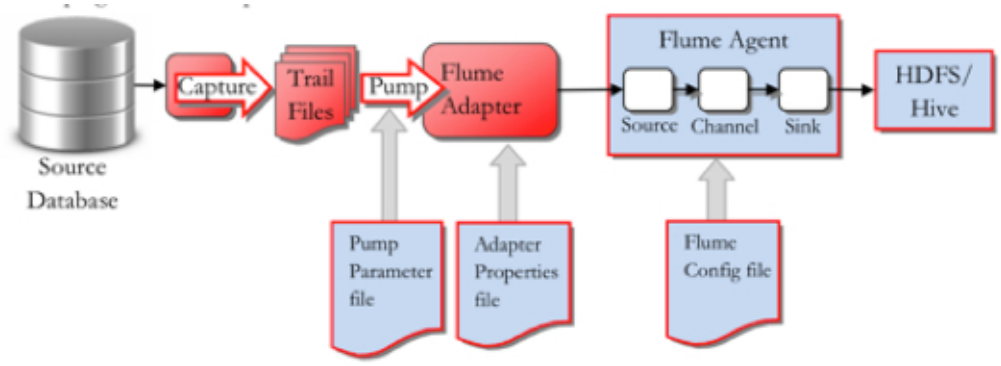

Figure 1: Oracle GoldenGate Adapter for Apache Flume Data Flow

Configuring and enabling Oracle GoldenGate for Big Data 12c to replicate data between an Oracle Database 12c table and a Hive table running on Hadoop requires the following tasks to be completed in order (I'll discuss the stages in detail shortly):

1: Configure the Oracle Database 12c environment for Oracle GoldenGate integrated capture.

2: Create a datasource definition file for the source database schema and table.

3: Configure Oracle GoldenGate for Oracle Database 12c, and create the extract process to capture the database transactions, using the datasource definition file generated in the previous step.

4: Configure and start a Flume agent to listen for Avro RPC message data from Oracle GoldenGate Adapter for Apache Flume.

5: Configure and start up Oracle GoldenGate Adapter for Apache Flume (again using the datasource definition file created in step 2), which will read from the trail file and send its log events to the Flume agent via Avro RPC.

6: Using the same table structure as the source Oracle Database table, create an external Hive table over the HDFS directory location where the Flume agent writes its incoming data, and then do a SELECT against this table to check that transactions from the source are being replicated successfully.

Configuring Oracle Database 12c for Integrated Capture and Creating the Extract Process

Oracle Database 12c on the Oracle Big Data Lite Virtual Machine needs to be configured for ARCHIVELOG mode and then enabled for Oracle GoldenGate for Oracle Database 12c integrated capture mode and supplemental logging. To do this, open the terminal application from the Oracle Linux desktop toolbar, via **Applications** -> **System Tools** -> **Terminal**. Enter the following commands to connect to SQL*Plus as the SYS user:

```
sqlplus sys/welcome1 as sysdba
shutdown immediate
startup mount;
alter database archivelog;
alter database open;
alter system set enable_goldengate_replication=true scope=both;
alter database add supplemental log data;
alter database force logging;
alter system switch logfile;
exit;
```

To create the schema and the table used in this example, download the create_example_db_data.sql script, save it to the default /home/oracle/Downloads directory, and run it with SQL*Plus. At the terminal command-line prompt, enter

```
cd $HOME/Downloads
sqlplus system@orcl/welcome1 @./create_example_db_data.sql
```

From the terminal prompt, enter the following commands to generate a source data definition file to use as part of the Flume agent configuration process later:

```
cd /u01/ogg
vi dirprm/defgg_test.prm

DEFSFILE ./dirdef/gg_test_hive.def,PURGE
userid system, password welcome1
SOURCECATALOG orcl
TABLE gg_test.logs;

./defgen PARAMFILE ./dirprm/defgg_test.prm
```

This creates a definition (DEF) file that describes the schema and table(s) the extract process is using in the trail file. Normally, you'd be able to use this file as is, but because you are extracting data from an Oracle Database 12c container database, you'll need to edit the file to remove the reference to the container database (the target Hive database uses SCHEMA.TABLE naming). Do this with the terminal application. Open the DEF file for editing:

```
vi ./dirdef/gg_test_hive.def
```

Then change the line that currently reads

```
Definition for table ORCL.GG_TEST.LOGS
```

to instead read

Definition for table `GG_TEST.LOGS`

Next, create an Oracle GoldenGate extract process called ORAEXT. The process will capture changes from the new table and map the resulting rows to a Hive table called LOGS in the gg_test Hive database. From the terminal prompt, enter the following command to create the oraext.prm parameter file for the ORAEXT process:

```
cd /u01/ogg
vi dirprm/oraext.prm

extract oraext
userid system, password welcome1
discardfile ./dirrpt/oraext.dsc, purge
targetdefs ./dirdef/gg_test_hive.def
exttrail ./dirdat/et
sourcecatalog orcl
table gg_test.logs target gg_test.logs;
```

While still using the terminal interface, employ the GGSCI shell to enable supplemental schema logging for the schema created earlier and then create and start the ORAEXT extract process:

```
./ggsci
start mgr
dblogin userid system@orcl password welcome1
add schematrandata gg_test allcols
dblogin userid system password welcome1
register extract oraext database container (orcl)
add extract oraext, integrated tranlog, begin now
add exttrail ./dirdat/et, extract oraext
start oraext
info all
```

After typing the last command, you should see the following GGSCI output:

```
Program     Status      GROUP       Lag at Chkpt      Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     ORAEXT      00:00:03          00:00:09
```

Note that the status will initially appear as STARTING but should change to RUNNING after a short while. If all is correct, enter exit to exit the GGSCI command-line shell.

Configuring Apache Flume and Oracle GoldenGate Adapter for Apache Flume

Now that your Oracle Database source is writing database transactions to the trail file, using Oracle GoldenGate 12c for Oracle Database integrated capture mode, you can configure Oracle GoldenGate Adapter for Apache Flume and the Flume agent that will receive and process these transactions for appending to the Hive table.

Configure the Flume agent that will receive flume events in Avro format from Oracle GoldenGate Adapter for Apache Flume, and then write them to the HDFS file system in the format Hive requires for table data storage. Again, from the terminal command line, create a configuration file—this time as the place the Flume agent should receive and store the data passed to it from Oracle GoldenGate Adapter for Apache Flume, as shown in **Listing 1**.

Code Listing 1: Creating the configuration file for the Flume agent

```
a1.channels = c1
a1.sources = r1
a1.sinks = k2
a1.channels.c1.type = memory
a1.sources.r1.channels = c1
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 1000

a1.sources.r1.type = avro
a1.sources.r1.bind = bigdatalite
a1.sources.r1.port = 4545
a1.sinks.k2.type = hdfs
a1.sinks.k2.channel = c1
a1.sinks.k2.hdfs.path = /user/oracle/gg/%{SCHE-
MA_NAME}/%{TABLE_NAME}
a1.sinks.k2.hdfs.filePrefix = %{TABLE_NAME}_
a1.sinks.k2.hdfs.writeFormat=Writable
a1.sinks.k2.hdfs.rollInterval=0
a1.sinks.k2.hdfs.hdfs.rollSize=1048576
a1.sinks.k2.hdfs.rollCount=0
a1.sinks.k2.hdfs.batchSize=1000
a1.sinks.k2.hdfs.fileType=DataStream
```

Next, start the Flume agent with this configuration file (using the Linux nohup command to ensure that the agent continues running even after you have exited the terminal command-line session):

```
nohup /usr/lib/flume-ng/bin/flume-ng agent --conf conf -f /usr/lib/
flume-ng/conf/gg-flume.conf -n a1 2>&1 > /tmp/gg-flume.out &
```

The next step is to create a properties file and a parameter file for Oracle GoldenGate Adapter for Apache Flume, using another extract process to read data from the trail file created by the ORAEXT process you set up earlier (as previously mentioned, you don't need an Oracle GoldenGate pump process in this example, because the database extract process and the Oracle GoldenGate Adapter for Apache Flume extract process are on the same Linux file system).

When this second extract process is running, it will take database log activity and event activity from the trail file and turn them into Flume events. These Flume events will be sent to the Flume agent, using the Avro RPC data exchange format over port 4545. At the terminal prompt, enter the code in **Listing 2** to create the Apache Flume properties file.

Code Listing 2: Creating the Apache Flume properties file

```
cd /u01/ogg
vi ./dirprm/flume.props

gg.handlerlist=ggflume

gg.handler.ggflume.type=com.goldengate.delivery.handler.-
flume.FlumeHandler
gg.handler.ggflume.host=bigdatalite

gg.handler.ggflume.port=4545
gg.handler.ggflume.rpcType=avro
gg.handler.ggflume.delimiter=\u0001

gg.handler.ggflume.mode=tx
gg.handler.ggflume.includeOpType=false
# Indicates if the operation time stamp should be included as part of
output in
the delimited separated values
# true - Operation time stamp will be included in the output
# false - Operation time stamp will not be included in the output
# Default :- true
gg.handler.ggflume.includeOpTimestamp=false

# Optional properties to use the transaction grouping functionality
gg.handler.ggflume.maxGroupSize=1000
gg.handler.ggflume.minGroupSize=1000

### native library config ###
goldengate.userexit.nochkpt=TRUE
goldengate.userexit.timestamp=utc
goldengate.log.logname=cuserexit
goldengate.log.level=INFO
goldengate.log.tofile=true
goldengate.userexit.writers=javawriter

gg.log=log4j
gg.log.level=info
```

```
gg.report.time=30sec
#gg.classpath=AdapterExamples/big-data/flume/target/flume-lib/*
#gg.classpath=AdapterExamples/big-data/flume/bin/*
gg.classpath=AdapterExamples/big-data/flume/bin/*:/usr/lib/ha-
doop/*:/usr/
lib/hadoop/lib/*:/usr/lib/hadoop/client/*:/etc/hadoop/-
conf.bigdatalite:/u01/
connectors/*:/usr/lib/flume-ng/lib/*

javawriter.stats.full=TRUE
javawriter.stats.display=TRUE
javawriter.bootoptions=-Xmx32m -Xms32m -Djava.class.path=ggjava/ggjava.jar
-Dlog4j.configuration=file:///u01/ogg/cfg/log4j.properties
```

Next, create the parameters file, which references the new properties file as well as the DEF file created earlier. At the terminal command-line prompt, create the file by entering

```
vi ./dirprm/flume.prm

EXTRACT flume
SETENV ( GGS_USEREXIT_CONF = "dirprm/flume.props")
CUSEREXIT libggjava_ue.so CUSEREXIT PASSTHRU INCLUDEUPDATEBEFORES
GETUPDATEBEFORES
NOCOMPRESSUPDATES
-- The definition file, generated before
SOURCEDEFS ./dirdef/gg_test_hive.def
DISCARDFILE ./dirrpt/flume.dsc, purge

TABLE gg_test.logs;
```

Finally, create and start up the Oracle GoldenGate Adapter for Apache Flume extract process, using the GGSCI command-line interface:

```
./ggsci
add extract flume, exttrailsource ./dirdat/et
start flume
info all
```

After entering `info all`, you should see the two extract processes and the Oracle GoldenGate Manager process running successfully:

```
Program     Status      GROUP       Lag at Chkpt     Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     FLUME       00:00:00         00:00:02
EXTRACT     RUNNING     ORAEXT      00:00:10         00:00:02
```

After entering `info all`, you should see the two extract processes and the Oracle GoldenGate Manager process running successfully:

Creating the Apache Hive Target Table and Testing the End-to-End Process

Oracle GoldenGate Adapter for Apache Flume should now be running and waiting for database transactions to take place for the GG_TEST .LOGS Oracle Database table, which will then be sent to the Flume agent and written into the HDFS within the Hadoop environment. To view this data in the form of a Hive table, you must create an external Hive table and specify the directories the Flume agent is writing to for the table's LOCATION clause. At the terminal command-line prompt, start the Hive shell and enter the following commands:

```
hive
create database IF NOT EXISTS gg_test;
drop table IF EXISTS gg_test.logs;
create external table IF NOT EXISTS gg_test.logs( log_id int,
log_message string, log_date string)  row format delimited fields
terminated by '\u0001' lines terminated by '\n' location '/user/
oracle/gg/gg_test/logs/';
exit;
```

Load some data into the source GG_TEST Oracle Database table, using a procedure installed earlier (when you ran the script to create the source schema and table). From the terminal command-line prompt, execute the procedure from SQL*Plus by entering

```
sqlplus gg_test@orcl/welcome1
begin P_GENERATE_LOGS(100); end;
/
```

Finally, return to the Hive command-line interface to see whether these new rows have been transported to the Hadoop environment. From the terminal command-line prompt, enter the following commands:

```
hive
select CONCAT('Rows loaded from gg_Test.logs into HDFS via Flume: ',
count(*)) from gg_test.logs;
```

and check that the output matches the following:

```
…
Rows loaded from gg_Test.logs into HDFS via Flume: 100
```

Congratulations! You've now successfully set up and enabled real-time replication between your Oracle Database and Hadoop environment on the Oracle Big Data Lite Virtual Machine.

Conclusion

As I've shown in this article, Oracle GoldenGate for Big Data 12c makes it possible to stream database transactions from Oracle Database and other database platforms into a Hadoop-based data reservoir, using the same Apache Flume ingest system for handling log and event sources. With Oracle GoldenGate for Oracle Database 12c and its integrated capture mode, it's simple and easy to enable this type of replication and stream data from Oracle Database 12c into your Oracle Big Data Appliance.