Data Engineer Technical Test

Google DataFlow (Apache Beam)

Overview

This test is performed to assess hands-on coding, software engineering and troubleshooting skills of candidates for Data Engineer position at Universal Music Group.

The Data Engineer position assumes high level of independence from the candidate, therefore the instructions given are minimal. We will point you to the documentation and will expect you do build a working solution on your own.

If you are stuck, you may have 30 minutes Q&A session with UMG Data Architect, but please use this opportunity wisely and when all the other options are exhausted.

Project Background

UMG has recently adopted Google BigQuery as enterprise-wide analytics platform and currently has a number of projects for creating data pipelines which get data from various sources, perform necessary transformations and output data to BigQuery. The volume of data is assessed at 100s of terabytes.

The chosen technology for these data pipelines is Google DataFlow which has been recently open-sourced by Google and re-branded as Apache Beam and currently at the incubating stage at Apache Foundation.

There are two Google DataFlow/Apache Beam SDKs: Java SDK and Python SDK. Java SDK is stable and production-ready and Python SDK is in Beta, but also pretty stable. The SDK of choice for UMG going forward will be Python, because of faster delivery, more concise code and its appeal to Data Science teams. However, there are pipelines already developed using Java SDK, so we expect you to be able to work with both.

Test

The test is a typical de-normalization task that is performed frequently when loading data to BigQuery. The test itself doesn't require interaction with BigQuery, as we find that final output of transformed

data to BigQuery is the easy part. The transformations in Google DataFlow are more complex and this is what we would like you to do.

You'll be given 3 files in gzip-archived JSON format that we receive from Spotify API: streams, tracks and users. Your job is to develop two pipelines in Google DataFlow (one in Java and one in Python) to denormalize these three files into one flat output JSON file.

Source Files

Schema of each source file is provided below. For simplicity, consider all the fields of type "string".

streams.gz

One record example:

{"message":"APIStreamData","version":"2","user_id":"9302111e473e4b76da615a47a133097a","track_id":"7718fc9605964e8698bc108bf9008864","length":"201","cached":"","source":"others_playlist","source_uri":"spotify:user:12150809594:playlist:60XVOmMeSorE8ZgLVpqDTN","device_type":"desktop","os":"Windows","stream_country":"CO","timestamp":"2016-10-19 19:00:00 UTC","report_date":"2016-10-19 00:00:00 UTC"}

Field
message
version
user_id
track_id
length
cached
source
source_uri
device_type
os
stream_country
timestamp
report_date

tracks.gz

One record example:

{"message":"APITrackData","version":"2","track_id":"7ce9b6b3b51e46b09f5698b53ab37ea9","uri":"spotify:track:3NHR4rVd31G2wZ2m6JNM3D","isrc":"SEUM70500101","album_code":"00602537010196","album_artist":"Abba","track_name":"Should I Laugh Or Cry - Intro Version","album_name":"The Visitors","track_artists":"ABBA"}

Field
message
version
track_id

uri
Isrc
album_code
album_artist
track_name
album_name
track_artists

users.gz

One record example:

{"message":"APIUserData","version":"2","user_id":"47ca39a988516984267a922d586fcccf","zip_code":"1","region":"","country":"UY","gender":"female","birth_year":"1996","referral":"","partner":"","product":"","type":"ad","access":"free"}

Field
message
version
user_id
zip_code
region
country
gender
birth_year
referral
partner
product
type
access

Output File

streams_denorm.json

One record example:

{"user_id":"aba7aebaa4972d3139a268444f1ca009","cached":"","timestamp":"2016-10-19 16:45:00 UTC","source_uri":"","track_id":"c7ff4b8cc72345bbaeec1ff6eb9bfee9","source":"other","length":"215","version":"2","device_type":"tablet”,"message":"APIStreamData","os":"iOS","stream_country":"DO","report_date":"2016-10-19 00:00:00 UTC","isrc":"USUL10400965","album_code":"00602498827840","product":"","country":"DO","region":"","zip_code":"12","access":"free","gender":"male","partner":"","referral":"","type":"ad","birth_year":"1986"}

Field	**Source file**
user_id	streams

cached	streams
timestamp	streams
source_uri	streams
track_id	streams
source	streams
length	streams
version	streams
device_type	streams
message	streams
os	streams
stream_country	streams
report_date	streams
isrc	tracks
album_code	tracks
product	users
country	users
region	users
zip_code	users
access	users
gender	users
partner	users
referral	users
type	users
birth_year	users

Instructions

Preparation

1. Create account on Google Cloud Platform. To get one, go to cloud.google.com and login with your Gmail email account. If you don't have Gmail email account, register for one. You'll be given a trial period and $300 in free spending.
2. Create a new bucket on Google Cloud Storage and upload provided source files into it.
3. Create a new bucket on Google Cloud Storage for staging files. You will need it to run DataFlow jobs
4. Create two solutions one for Java SDK and one for Python SDK. It's up to you what IDE and configuration to use. At UMG we are using Intellij IDEA + Gradle for Java SDK and Intellij PyCharm IDE for Python.
5. Try examples provided with SDKs to make sure that your IDE environments is configured properly. Try in in both DirectPipelineRunner and DataflowPipelineRunner (Java)/BlockingDataflowPipelineRunner (Python).
    a. Java SDK examples: https://github.com/GoogleCloudPlatform/DataflowJavaSDK/tree/master/examples
    b. Python SDK examples:

i. https://github.com/apache/incubator-beam/tree/python-sdk/sdks/python
ii. https://github.com/apache/incubator-beam/tree/python-sdk/sdks/python/apache_beam/examples

Java SDK Tips

1. User TextIO.Read to read files from Google Cloud Storage
2. Join files with CoGroupByKey transformation. Example of this transformation is provided in TfIdf example under https://github.com/GoogleCloudPlatform/DataflowJavaSDK/blob/master/examples/src/main/java/com/google/cloud/dataflow/examples/complete/TfIdf.java
3. Use TextIO.Write to write resulting denormalized file to Google Cloud Storage

Python SDK Tips

1. Configure your Python module same way as in JuliaSet example: https://github.com/apache/incubator-beam/tree/python-sdk/sdks/python/apache_beam/examples/complete/juliaset
2. Use Read(io.TextFileSource()) to read files from Google Cloud Storage
3. Use CoGroupByKey() transformation to join files together. Example of CoGroupByKey for Python is provided in https://github.com/apache/incubator-beam/blob/python-sdk/sdks/python/apache_beam/examples/complete/tfidf.py
4. Write demormalized file with Write(io.TextFileSink()) to Google Cloud Storage

Approximate number of code lines for Java solution is 350 lines and for Python solution around 100 lines.

There will be some routine tasks such as converting from/to JSON, dealing with key/value pairs for joining data, etc. We leave this to your expertise to figure out how to do it.

Good luck!