## DataFrames

So far, we've introduced the concept of distributed systems using parallel processing on a single machine with ipyparallel and its scatter/gather functions. We then expanded on this idea by using different nodes (instead of different processors) to distribute our workload with the map/reduce paradigm. In the last lab, we explored how Spark can also handle map/reduce jobs.

In this lab, we will repeat what we've been doing, but this time leverage Spark's [DataFrame](http://spark.apache.org/docs/latest/sql-programming-guide.html) API. This is similar in many ways to Pandas' [DataFrame](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html) only it uses camelCase because it's Spark (so groupBy instead of groupby for example). The way it uses select also takes a little getting used to but will make more sense when we see it in terms of Spark SQL.

Are you tired of Zeppelin yet? That's good because we are going to go back to using Jupyter now to make sure our DataFrames work. (When testing this lab earlier this month, we found that Zeppelin did not interface with Spark correctly when trying to create a DataFrame out of JSON. We're not huge fans of Zeppelin anyway, so rather than fix it, we decided to use Jupyter instead.)

After you have created the EMR Spark cluster, follow these steps:

1. Open the Security Groups for the Master node, and add 2 inbound rules for SSH (anywhere) and Custom TCP Port 8888 (anywhere)

2. SSH to the master node (note that the username is hadoop, not ubuntu).

3. run sudo pip install pyyaml ipython jupyter ipyparallel pandas boto -U

4. You will need to do two things: set PYSPARK\_DRIVER\_PYTHON to the location of jupyter and set PYSPARK\_DRIVER\_PYTHON\_OPTS so that it tells the notebook to accept connections from all IP addresses and without opening a brower. To find out where jupyter is installed, you can use the which command (as in which jupyter). Your PYSPARK\_DRIVER\_PYTHON\_OPTS will look familiar is they are the same options you used in the Distributed Systems lab, namely notebook --no-broweser  --ip=0.0.0.0 ---port=8888. In the end, you will want to append the following two lines to your .bashrc file:

export PYSPARK\_DRIVER\_PYTHON=/usr/local/bin/jupyter

export PYSPARK\_DRIVER\_PYTHON\_OPTS="notebook --no-browser  --ip=0.0.0.0 --port=8888"

Then run source ~/.bashrc followed by pyspark (not jupyter notebook). You should see the familiar standard output telling you that Jupyter Notebook is starting.

Make sure you are allowing connections on port 8888 (in [Security Groups](https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroups:search=ElasticMapReduce-master)) and point your browser to the master node:8888 and don't forget the token. (i.e. your URL should look something like http://ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com:8888/tree?token=000111222333444555666777888999aaabbbcccdddeeefff#)

From there you should see the familiar Jupyter console. As in the previous project, the Spark context (sc) and session (spark) are already given so you can get started right away. Other classes and functions, like Spark SQL's [explode](https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.functions.explode) function, will need to be imported in the usual way.

For this lab you must create at least four (4) DataFrames. The first two will be saved as Parquet files to be used later. The third is an aggregation on the second. All three should be normalized.

The first DataFrame should include atomic data about the event logs. You will need to create an unique id for each event, which will be used as the key. It must include ts and userId. Notice that there is a bit of a problem here if you use select('id', 'ts', 'userId', 'song') because id is ambiguous. Spark will let you do this but it will cause problems later. Instead try selectExpr('id', 'ts', 'userId AS user\_id', 'song'). You may include other data (like song) if you wish.

The second DataFrame should consist of your user data (firstName, lastName, gender, location, level, userId) for this. But be sure to store only distinct users.

The third DataFrame should have as a composite key the id and artist.

[Write](http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrameWriter.parquet) these three to S3 in [Parquet](https://parquet.apache.org/) format for use in the next project.

Finally, using the artist DataFrame, compute an aggrate to show the top 10 artists as before.

**Submission:** Submit your code as a Jupyter Notebook (.ipynb) file.