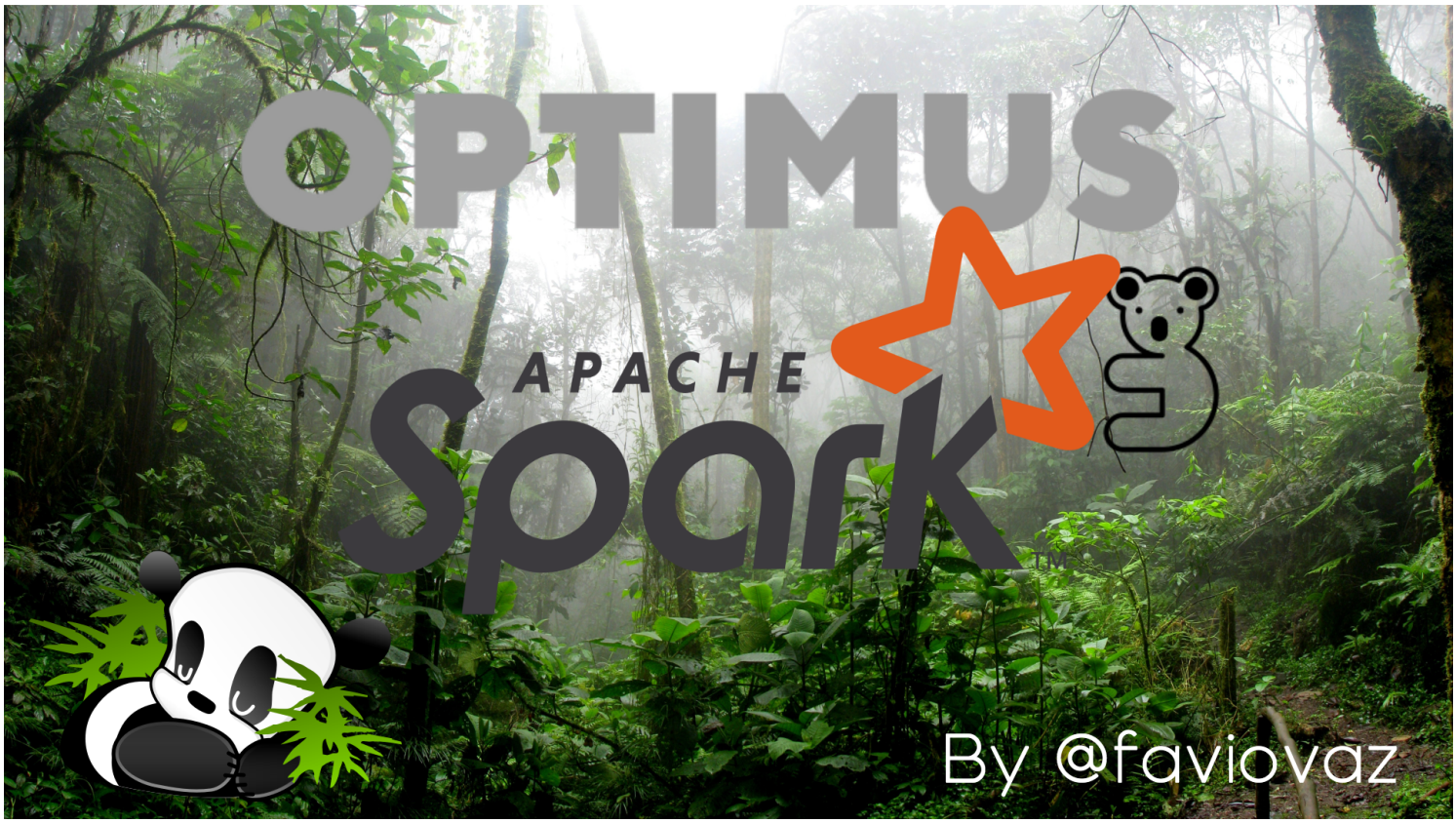
The Jungle of Koalas, Pandas, Optimus and Spark

What to expect from the newest library from Databricks (Koalas), the Optimus framework and Apache Spark 3.x

Favio Vázquez Follow
Apr 25 · 8 min read



If you are as excited about data science as me, you probably know that the Spark+AI latest summit started yesterday (April 24th 2019). And there are great things to talk about. But I will do it with a spin-off.

If you've been following me you now that I co-created a framework called Optimus. If you want to see more about that check these articles:

Data Science with Optimus. Part 1: Intro.
Breaking down data science with Python, Spark and Optimus.

towardsdatascience.com

Data Science with Optimus. Part 2: Setting your DataOps Environment.

Breaking down data science with Python, Spark and Optimus. Today: Data Operations for Data Science. ..::Part 1 here...

towardsdatascience.com

Where I'm explaining a whole data science environment with Optimus and other open source tools. This could have been part 3, but I'm more interested in showing you other stuff.

Optimus APIs

https://github.com/ironmussa/Optimus

In the beginning Optimus started as a project for cleaning big data, but suddenly we realized that there was a lot of more opportunities for the framework. Right now we have created a lot of interesting things and if you are a data scientist using pandas, or spark you should check it out.

Right now we have these APIs:

- Improved versions of Spark Dataframes (much better for cleaning and munging data).
- Easier Machine Learning with Spark.

- Easier Deep Learning with Spark and Spark Deep Learning.
- Plots directly from Spark Dataframes.
- Profiling Spark Dataframes.
- Database connections (like Amazon Redshift) easier.
- Enrich data connecting with external APIs.

And more. You can even read data directly from the internet to Spark.

So you can see we have been trying a lot to improve the world of the data scientist. One of the things we care was creating a simple and usable API, and we didn't love Pandas API or Spark API by itself, but a combination of those with a little touch of awesomeness created what you can call today our framework.

Koalas vs Optimus vs Spark vs Pandas

https://www.freepik.com/

Today Databricks announced the project Koala as a more productive way when interacting with big data, by augmenting Apache Spark's Python DataFrame API to be compatible with Pandas.

If you want to try it check this MatrixDS project:

MatrixDS | The Data Project Workbench

MatrixDS is a place to build, share and manage data projects at any scale.

community.platform.matrixds.com

And this GitHub repo:

FavioVazquez/koalas_optimus_spark

Rocking the world with Spark and friends. . Contribute to FavioVazquez/koalas_optimus_spark development by creating an...

github.com

So instead of boring you with copy-pasting the documentation of Koalas that you can read right now, I created a simple example of the connection between Koalas, Optimus and Spark.

You'll need to install Optimus

```
pip install --user optimuspyspark
```

and Koalas

```
pip install --user koalas
```

I'll be using this dataset for testing:

https://raw.githubusercontent.com/databricks/koalas/master/data/sample_stocks.csv

Let's first read data with Spark vanilla:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

df = spark.read.csv("sample_stocks.csv", header=True)
```

For that I needed to upload the dataset before. Let's see that in Optimus:

```
from optimus import Optimus
op = Optimus()
df =
op.load.url("https://raw.githubusercontent.com/databricks/koalas/mas
ter/data/sample_stocks.csv")
```

That was one step simpler because with Optimus you can read data directly from the web.

What about Koalas?

```
import databricks.koalas as ks

df =
ks.read_csv("https://raw.githubusercontent.com/databricks/koalas/mas
ter/data/sample_stocks.csv")
```

This code will fail, would that happen in Pandas?

```
import pandas as pd

df =
pd.read_csv("https://raw.githubusercontent.com/databricks/koalas/mas
ter/data/sample_stocks.csv")
```

Well no. That would work. That's because you can read data with Pandas from the web directly. Ok so let's make the Koalas code work:

```
import databricks.koalas as ks

df_ks = ks.read_csv("sample_stocks.csv")
```

Well that looks simple enough. By the way, if you want to read the data from the local storage with Optimus it's almost the same:

```
from optimus import Optimus
op = Optimus()
df_op_local = op.load.csv("sample_stocks.csv")
```

But, let's take a look at what happen next. What are the types of this Dataframes?

```
print(type(df_sp))
print(type(df_op))
print(type(df_pd))
print(type(df_ks))
```

And the result is:

```
<class 'pyspark.sql.dataframe.DataFrame'>
<class 'pyspark.sql.dataframe.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'databricks.koalas.frame.DataFrame'>
```

So the only framework that created a Spark DF apart from Spark itself, was Optimus. What does this mean?

Let's see what happens when we want to show the data. For showing data in Spark we normally use the *.show()* method, and for Pandas the *.head()* method.

```
df_sp.show(1)
```

Will work as expected.

```
df_op.show(1)
```

Will work too.

```
df_pd.head(1)
```

Will work as well. But what about out Koalas DF? Well you need to use the pandas API, because that's one of the goals of the library, make the transition easier from pandas. So:

```
df_ks.show(1)
```

Will fail, but

```
df_ks.head(1)
```

Will work.

If you are running this code along with me, if you hit *show* for spark, this is what you saw:

```
+----------+------+------+------+------+----------+----------+------
----+-------+-------+------+--------+----------+------+
|      Date|  Open|  High|   Low| Close|
```

```
Volume|ExDividend|SplitRatio|AdjOpen|AdjHigh|AdjLow|AdjClose|
AdjVolume|Symbol|
+----------+------+------+------+------+----------+----------+------
----+-------+-------+------+--------+----------+------+
|2018-03-27|173.68|175.15|166.92|168.34|38962839.0|       0.0|
1.0| 173.68| 175.15|166.92|  168.34|38962839.0|  AAPL|
+----------+------+------+------+------+----------+----------+------
----+-------+-------+------+--------+----------+------+
only showing top 1 row
```

Which is kinda awful. Everyone prefers those pretty HTML outline tables to see their data, and Pandas has them, so Koalas inherits them from Pandas. But remember, this are not Spark DF. If you really want to see a prettier version of Spark DF with Optimus you can use the *.table()* method.

```
df_op.table(1)
```

and you'll see:

```
: df_op.table(1)
```

Viewing 1 of 5148 rows / 14 columns

1 partition(s)

Date 1 (timestamp) nullable	Open 2 (double) nullable	High 3 (double) nullable	Low 4 (double) nullable	Close 5 (double) nullable	Volume 6 (double) nullable	ExDividend 7 (double) nullable	SplitRatio 8 (double) nullable	AdjOpen 9 (double) nullable	AdjHigh 10 (double) nullable	AdjLow 11 (double) nullable	AdjClose 12 (double) nullable	AdjVolume 13 (double) nullable	Symbol 14 (string) nullable
2018-03-26□ 17:00:00	173.68	175.15	166.92	168.34	38962839.0	0.0	1.0	173.68	175.15	166.92	168.34	38962839.0	AAPL

Viewing 1 of 5148 rows / 14 columns

1 partition(s)

which shows you the data better plus information about it like the types of the columns, the number of rows in the DF, the number of columns and partitions.

Selecting data

Let's do more with our data. Like slicing it.

I'll choose the columns Date, Open, High, Low and Volume with the frameworks. There may be more ways of selecting data, I'm just using the common ones.

With Spark:

```
# With Spark
df_sp["Date","Open","High","Volume"].show(1)

# or
df_sp.select("Date","Open","High","Volume").show(1)
```

With Optimus:

```
df_op["Date","Open","High","Volume"].table(1)

# or
df_op.select("Date","Open","High","Volume").table(1)

# or with indices :)

df_op.cols.select([0,1,2,5]).table(1)
```

With Pandas:

```
df_pd[["Date","Open","High","Volume"]].head(1)

#or

df_pd.iloc[:, [0,1,2,4]].head(1)
```

With Koalas:

```
df_ks[["Date","Open","High","Volume"]].head(1) # will work

df_ks.iloc[:, [0,1,2,4]].head(1) # will fail

df_ks.select("Date","Open","High","Volume") # will fail
```

So as you can see right now we have good support of different things with Optimus, and if you love the [[]] style from Pandas, you can use it with Koalas too, but you can't select by indices, at least not yet.

The difference here is with Koalas and Optimus you are running Spark code underneath, so you don't have to worry about performance. At least not right now.

More advance stuff:

Let's get the frequencies for a column:

Pandas:

```
df_pd["Symbol"].value_counts()
```

Koalas:

```
df_ks["Symbol"].value_counts()
```

They're the same which is very cool.

Spark (some of the bad parts):

```
df_sp.groupBy('Symbol').count().show()
```

Optimus (you can do the same as in Spark):

```
df_op.groupBy('Symbol').count().show()
```

or you can use the *.cols* attribute to get more functions:

```
df_op.cols.frequency("Symbol")
```

Let's transform our data with One-Hot-Enconding:

Pandas:

```
# This is crazy easy
pd.get_dummies(data=df_pd, columns=["Symbol"]).head(1)
```

Koalas:

```
# This is crazy easy too
ks.get_dummies(data=df_ks, columns=["Symbol"]).head(1)
```

Spark (similar enough result but horrible to do):

```
# I hate this
from pyspark.ml.feature import StringIndexer,OneHotEncoderEstimator

indexer = StringIndexer(inputCol="Symbol", outputCol="SymbolIndex")
df_sp_indexed = indexer.fit(df_sp).transform(df_sp)

encoder = OneHotEncoderEstimator(inputCols=["SymbolIndex"],
                                 outputCols=["SymbolVec"])

model = encoder.fit(df_sp_indexed)
df_sp_encoded = model.transform(df_sp_indexed)
df_sp_encoded.show(1)
```

Optimus (a little better but I still prefer Koalas for this):

```
from optimus.ml.feature import string_to_index, one_hot_encoder

df_sp_indexed = string_to_index(df_sp, "Symbol")
df_sp_encoded = one_hot_encoder(df_sp_indexed, "Symbol_index")
df_sp_encoded.show()
```

So in this case the easier way was from Pandas, and luckily it's implemented in Koalas, and this types of functions will increase in the future, but right now this is almost all we have as you can see here:

General functions - Koalas 0.1.0 documentation

Edit description

koalas.readthedocs.io

But they run in Spark so it rocks.

Plots:

Plotting is an important part of data analysis. With Pandas we are use to plot what ever we want very easily, but with Spark it's not that easy. We are happy to announce that in the latest version of Optimus (2.2.2) we have created a way of creating plots directly from your Spark DataFrames, no subsetting needed.

Pandas:

```
df_pd.plot.scatter("Open","Volume")
```

```
df_pd.boxplot("High")
```



```
df_pd.hist("Low")
```

Koalas:

```
df_ks.hist("Low")
```

This will not work.

Spark:

Nope.

Optimus:

```
df_op.plot.boxplot(“High”)
```

```
df_op.plot.hist(“Low”)
```

```
df_op.plot.scatterplot(["Open","Volume"])
```


Apache Spark 3.x:

databricks.com

These are some of the things to expect from Spark 3.x:

- Scala 2.12 support (listing it twice)
- Continuous Processing non-experimental
- Kubernetes support non-experimental
- A more flushed out version of data source API v2
- Hadoop 3.0 support
- Improve usability of Spark Deep Learning Pipelines

And more!

databricks.com

Conclusion

Spark it's growing exponentially, and if you are not using it now, you definitely should. Commonly you will be coming from Pandas or something like that, so make use of great libraries like Koalas and Optimus to improve your life in the Data Science world.

. . .

If you have any questions please write me here:

Favio Vazquez — Founder / Chief Data Scientist — Ciencia y Datos | LinkedIn

Join LinkedIn !!!! Important Note: Due to Linkedin technical limitations, I can now only accept connection requests...

www.linkedin.com

and follow me on twitter:

Favio Vázquez (@FavioVaz) | Twitter

The latest Tweets from Favio Vázquez (@FavioVaz). Data Scientist. Physicist and computational engineer. I have a...

twitter.com

Have fun learning :)

)

Technology Data Science Business Artificial Intelligence Machine Learning

About Help Legal