Run Pandas as Fast as Spark

Why the Pandas API on Spark is a total game changer

 Adrián González Carpintero Nov 27 · 4 min read

Photo by Clayton Holmes on Unsplash

That's it. It's out. Spark now has a Pandas API.

It seems that, every time you want to work with Dataframes, you have to open a messy drawer where you keep all the tools, and carefully look for the right one.

If you work with structured data, you need SQL. Pandas is always there too. Spark is indispensable for Big Data. There is a tool for every need in the toolbox. But you don't need the toolbox anymore, because Spark has become the ultimate Swiss Army Knife.

It all started at the 2019 Spark + AI Summit. Koalas, an open source project that enables the use of Pandas on top of Spark, was launched. At the beginning, it only covered a small part of the Pandas functions, but it gradually grew. Two years have passed, and now, in the new Spark 3.2 release, Koalas has been merged into PySpark. And the result is wonderful.

Spark now integrates a Pandas API so you can run Pandas on top of Spark. You just have to change one line of your code:

```
import pyspark.pandas as ps
```

Isn't it great?

Thanks to this, we can obtain a wide range of benefits:

- If you use Pandas but you are not familiar with Spark, you can work with Spark right away, with no learning curve.
- You can have a single codebase for everything: small data and big data. A single machine and distributed machines.
- You can run your Pandas code faster.

This last point is especially remarkable.

On the one hand, you can apply distributed computing to your code in Pandas. But the benefits don't end there. Thanks to the Spark engine, your code will be faster even in a single machine!

And in case you are wondering, yes, it looks like Pandas-on-Spark is also faster than Dask.

I'm really excited about this breakthrough so, why don't we get down to business? Let's do some code with the Pandas API on Spark!

Switching between Pandas, Pandas-on-Spark, and Spark

The first thing we need to know is what exactly we are working with. When working with Pandas, we use the class `pandas.core.frame.DataFrame`. When working with the pandas API in Spark, we use the class `pyspark.pandas.frame.DataFrame`. Both are similar, but not the same. The main difference is that the former is in a single machine, whereas the latter is distributed.

We can create a Dataframe with Pandas-on-Spark and convert it to Pandas, and vice-versa:

```
# import Pandas-on-Spark
import pyspark.pandas as ps

# Create a DataFrame with Pandas-on-Spark
ps_df = ps.DataFrame(range(10))

# Convert a Pandas-on-Spark Dataframe into a Pandas Dataframe
pd_df = ps_df.to_pandas()

# Convert a Pandas Dataframe into a Pandas-on-Spark Dataframe
ps_df = ps.from_pandas(pd_df)
```

Note that if you are using multiple machines, when converting a Pandas-on-Spark Dataframe into a Pandas Dataframe, data is transferred from multiple machines to a single one, and vice-versa (see PySpark guide).

We can also convert a Pandas-on-Spark Dataframe into a Spark DataFrame, and vice-versa:

```
# Create a DataFrame with Pandas-on-Spark
ps_df = ps.DataFrame(range(10))

# Convert a Pandas-on-Spark Dataframe into a Spark Dataframe
spark_df = ps_df.to_spark()

# Convert a Spark Dataframe into a Pandas-on-Spark Dataframe
ps_df_new = spark_df.to_pandas_on_spark()
```

What happens with data types?

When working with Pandas-on-Spark and Pandas, the data types are basically the same. When converting a Pandas-on-Spark DataFrame to a Spark DataFrame, data types are casted to the appropriate type automatically (see PySpark guide)

Replicating Spark functions with Pandas-on-Spark

The aim of this section is to provide a cheatsheet with the most used functions for managing DataFrames in Spark and their analogues in Pandas-on-Spark. Note that the only difference in syntax between Pandas-on-Spark and Pandas is just the `import pyspark.pandas as ps` line.

You will see how, even if you are not familiar with Spark, working with it is straightforward thanks to the Pandas API.

IMPORT LIBRARIES

```
# For running Spark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Spark").getOrCreate()

# For running Pandas on top of Spark
import pyspark.pandas as ps
```

READ DATA

Let's use the old dog iris dataset as example.

```
# SPARK
sdf = spark.read.options(inferSchema='True',
              header='True').csv('iris.csv')

# PANDAS-ON-SPARK
pdf = ps.read_csv('iris.csv')
```

SELECT

```
# SPARK
sdf.select("sepal_length","sepal_width").show()
```

```
# PANDAS-ON-SPARK
pdf[["sepal_length","sepal_width"]].head()
```

DROP COLUMNS

```
# SPARK
sdf.drop('sepal_length').show()

# PANDAS-ON-SPARK
pdf.drop('sepal_length').head()
```

DROP DUPLICATES

```
# SPARK
sdf.dropDuplicates(["sepal_length","sepal_width"]).show()

# PANDAS-ON-SPARK
pdf[["sepal_length", "sepal_width"]].drop_duplicates()
```

FILTER

```
# SPARK
sdf.filter( (sdf.flower_type == "Iris-setosa") & (sdf.petal_length > 1.5)
).show()

# PANDAS-ON-SPARK
pdf.loc[ (pdf.flower_type == "Iris-setosa") & (pdf.petal_length > 1.5)
].head()
```

COUNT

```
# SPARK
sdf.filter(sdf.flower_type == "Iris-virginica").count()

# PANDAS-ON-SPARK
pdf.loc[pdf.flower_type == "Iris-virginica"].count()
```

DISTINCT

```
# SPARK
sdf.select("flower_type").distinct().show()

# PANDAS-ON-SPARK
pdf["flower_type"].unique()
```

SORT

```
# SPARK
sdf.sort("sepal_length", "sepal_width").show()

# PANDAS-ON-SPARK
pdf.sort_values(["sepal_length", "sepal_width"]).head()
```

GROUP BY

```
# SPARK
sdf.groupBy("flower_type").count().show()

# PANDAS-ON-SPARK
pdf.groupby("flower_type").count()
```

REPLACE

```
# SPARK
sdf.replace("Iris-setosa", "setosa").show()

# PANDAS-ON-SPARK
pdf.replace("Iris-setosa", "setosa").head()
```

JOIN

```
# SPARK
sdf.union(sdf)
```

```
# PANDAS-ON-SPARK
pdf.append(pdf)
```

Conclusion

From now on, you will be able to use Pandas in top of Spark. This leads to an increase in Pandas speed, a decrease in the learning curve when emigrating to Spark, and the merging of single machine computing and distributed computing in the same codebase.