

Laboratorio 2

Harold Shneider Martínez Tapiero - 77999
Johan Steven Benavides Guarnizo – 88593

Elías Buitrago Bolívar

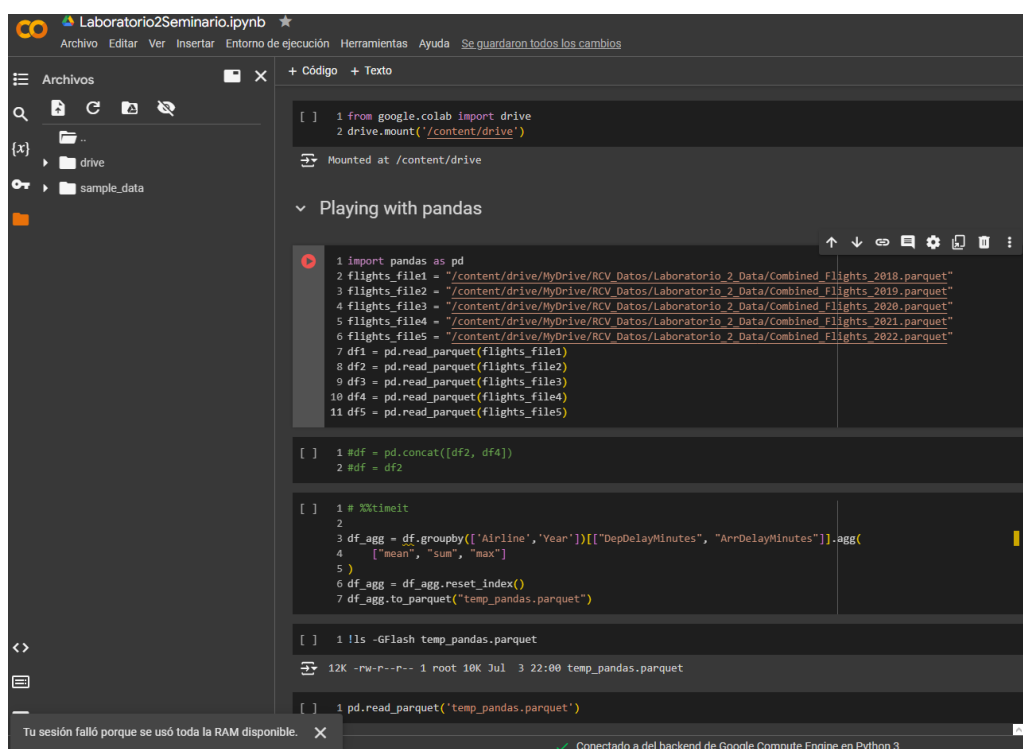
Universidad ECCI
Facultad de Ingeniería
Bogotá
2.024

Introducción

Con el fin de comparar diferentes bibliotecas en cuanto a su rendimiento y efectividad en el procesamiento de archivos .parquet, fue necesario observar más a detalle qué tipo de conjuntos de datos trabajan. Con la siguiente investigación se analizó la capacidad de las diferentes bibliotecas, en escenarios correspondientes, arrojando resultados diferentes y dando unas conclusiones de cuales son mejores, y optimas, hasta otro en los cuales se evidencia una mejor carga, pero a su vez un gasto de recursos más altos.

PANDAS

Prueba 1:



```
[ ] 1 from google.colab import drive
2 drive.mount('/content/drive')

Mounted at /content/drive

Playing with pandas

[ ] 1 import pandas as pd
2 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
3 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
4 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
5 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
6 flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
7 df1 = pd.read_parquet(flights_file1)
8 df2 = pd.read_parquet(flights_file2)
9 df3 = pd.read_parquet(flights_file3)
10 df4 = pd.read_parquet(flights_file4)
11 df5 = pd.read_parquet(flights_file5)

[ ] 1 df1 = pd.concat([df2, df4])
2 df = df2

[ ] 1 # %%timeit
2
3 df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
4     ['mean', 'sum', 'max']
5 )
6 df_agg = df_agg.reset_index()
7 df_agg.to_parquet("temp_pandas.parquet")

[ ] 1 !ls -lG temp_pandas.parquet

12K -rw-r--r-- 1 root 10K Jul 3 22:00 temp_pandas.parquet

[ ] 1 pd.read_parquet("temp_pandas.parquet")

Tu sesión falló porque se usó toda la RAM disponible. X
```

Archivos usados: Se usaron los 5 archivos

RAM: - / 12.7 GB

NOTAS: Arroja error por falta de recursos, en este caso se superan los 12.7 GB de RAM

Ejecución: 1 min Aprox

Prueba 2:

Playing with pandas

```
1 import pandas as pd
2 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
3 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
4 #flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
5 #flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
6 flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
7 df1 = pd.read_parquet(flights_file1)
8 df2 = pd.read_parquet(flights_file2)
9 #df3 = pd.read_parquet(flights_file3)
10 #df4 = pd.read_parquet(flights_file4)
11 df5 = pd.read_parquet(flights_file5)
```

```
1 df = pd.concat([df1, df2, df5])
2 #df = df2
```

```
[ ] 1 # %%timeit
```

Archivos usados: 1, 2 y 5

RAM: - / 12.7 GB

NOTAS: Se ejecuta el error al momento de concatenar por falta de recursos.

Ejecución: 39 SEG

Prueba 3:

```
[1] 1 import pandas as pd
2 #flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
3 #flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
4 #flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
5 #flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
6 flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
7 #df1 = pd.read_parquet(flights_file1)
8 #df2 = pd.read_parquet(flights_file2)
9 df3 = pd.read_parquet(flights_file3)
10 #df4 = pd.read_parquet(flights_file4)
11 df5 = pd.read_parquet(flights_file5)

[2] 1 df = pd.concat([df3, df5])
2 #df = df2

[3] 1 # %%timeit
2
3 df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
4     ["mean", "sum", "max"]
5 )
6 df_agg = df_agg.reset_index()
7 df_agg.to_parquet("temp_pandas.parquet")

[5] 1 !ls -l temp_pandas.parquet

12K -rw-r--r-- 1 root 9.9K Jul 17 15:20 temp_pandas.parquet

[6] 1 pd.read_parquet("temp_pandas.parquet")
```

¿Quieres más memoria y espacio en el disco? [Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3
Se muestran los recursos desde el 10:19 a.m. hasta el 10:22 a.m.

RAM de sistema	Disco
9.6 / 12.7 GB	28.4 / 107.7 GB

Cambiar tipo de entorno de ejecución

Archivos usados: 3 y 5

RAM: 9.6 / 12.7 GB

NOTAS: Se ejecutaron los comandos sin errores

Ejecución: 27 Seg

Prueba 4:

```
1 import pandas as pd
2 #flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
3 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
4 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
5 #flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
6 #flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
7 #df1 = pd.read_parquet(flights_file1)
8 df2 = pd.read_parquet(flights_file2)
9 df3 = pd.read_parquet(flights_file3)
10 #df4 = pd.read_parquet(flights_file4)
11 #df5 = pd.read_parquet(flights_file5)

1 df = pd.concat([df2, df3])
2 #df = df2

[ ] 1 # %%timeit
2
3 df_agg = df.groupby(['Airline', 'Year'])[['DepDelayMinutes', 'ArrDelayMinutes']].agg(
4     ["mean", "sum", "max"]
5 )
6 df_agg = df_agg.reset_index()
7 df_agg.to_parquet("temp_pandas.parquet")
```

Archivos usados: 2 y 3

RAM: - / 12.7 GB

NOTAS: Se ejecutaron los comandos arrojando error en la concatenación por falta de RAM

POLARS

Prueba 1:

```
[1] 1 import polars as pl

1 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
6 df1 = pl.scan_parquet(flights_file1)
7 df2 = pl.scan_parquet(flights_file2)
8 df3 = pl.scan_parquet(flights_file3)
9 df4 = pl.scan_parquet(flights_file4)
10 df5 = pl.scan_parquet(flights_file5)

[3] 1 %%timeit
2
3 df_polars = (
4     pl.concat([df1, df2, df3, df4, df5])
5     .groupby(['Airline', 'Year'])
6     .agg([
7         pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
8         pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
9         pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
10        pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
11        pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
12        pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
13    ])
14 ).collect()
15
16 df_polars.write_parquet("temp_polars.parquet")

<magic-timeit>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
10.4 s ± 788 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 llis -GFlash temp_polars.parquet

12K -rw-r--r-- 1 root 8.1K Jul 17 15:29 temp_polars.parquet
```

Archivos usados: Todos

RAM: 1.4 / 12.7 GB

NOTAS: Se ejecutaron todos los comandos correctamente

Ejecución: 1min

Prueba 2:

```
[1] 1 import polars as pl

[5] 1 #flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 #flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
6 #df1 = pl.scan_parquet(flights_file1)
7 df2 = pl.scan_parquet(flights_file2)
8 #df3 = pl.scan_parquet(flights_file3)
9 df4 = pl.scan_parquet(flights_file4)
10 df5 = pl.scan_parquet(flights_file5)

1 %timeit
2
3 df_polars = (
4     pl.concat([df2, df4, df5])
5     .groupby(['Airline', 'Year'])
6     .agg([
7         pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
8         pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
9         pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
10        pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
11        pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
12        pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
13    ])
14 ).collect()
15
16 df_polars.write_parquet('temp_polars.parquet')

<magic-timetr>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
6.42 s ± 721 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 lls -GFlash temp_polars.parquet

8.0K -rw-r--r-- 1 root 6.3K Jul 17 15:31 temp_polars.parquet
```

Recursos

No tienes una suscripción. [Más información](#)

En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades [aquí](#).

En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 84 horas.

[Administrar sesiones](#)

¿Quieres más memoria y espacio en el disco? [Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3

Se muestran los recursos desde el 10:24 a.m. hasta el 10:31 a.m.

RAM de sistema	Disco
3.3 / 12.7 GB	28.4 / 107.7 GB

Archivos usados: 2, 4 y 5

RAM: 1.3 / 12.7 GB

NOTAS: Se ejecutaron todos los comandos correctamente

Ejecución: 50 SEG

Prueba 3:

```
[1] 1 import polars as pl

[10] 1 #flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 #flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
6 #df1 = pl.scan_parquet(flights_file1)
7 df2 = pl.scan_parquet(flights_file2)
8 df3 = pl.scan_parquet(flights_file3)
9 df4 = pl.scan_parquet(flights_file4)
10 #df5 = pl.scan_parquet(flights_file5)

1 %timeit
2
3 df_polars = (
4     pl.concat([df2, df3, df4])
5     .groupby(['Airline', 'Year'])
6     .agg([
7         pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
8         pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
9         pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
10        pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
11        pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
12        pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
13    ])
14 ).collect()
15
16 df_polars.write_parquet('temp_polars.parquet')

<magic-timetr>:3: DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'group_by'.
6.38 s ± 758 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 lls -GFlash temp_polars.parquet

8.0K -rw-r--r-- 1 root 6.4K Jul 17 15:34 temp_polars.parquet
```

Recursos

No tienes una suscripción. [Más información](#)

En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades [aquí](#).

En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 84 horas.

[Administrar sesiones](#)

¿Quieres más memoria y espacio en el disco? [Actualizar a Colab Pro](#)

del backend de Google Compute Engine en Python 3

Se muestran los recursos desde el 10:24 a.m. hasta el 10:34 a.m.

RAM de sistema	Disco
1.4 / 12.7 GB	28.4 / 107.7 GB

Archivos usados: 2, 3 y 4

RAM: 1.4 / 12.7 GB

NOTAS: Se ejecutaron todos los comandos correctamente

Ejecución: 51 SEG

Prueba 4:

```
[1] 1 import polars as pl

1 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 #flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 #flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 flights_files = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"
6 df1 = pl.scan_parquet(flights_file1)
7 #df2 = pl.scan_parquet(flights_file2)
8 #df3 = pl.scan_parquet(flights_file3)
9 df4 = pl.scan_parquet(flights_file4)
10 df5 = pl.scan_parquet(flights_files)

[13] 1 %timeit
2
3 df_polars = (
4     pl.concat([df1, df4, df5])
5     .groupby(['Airline', 'Year'])
6     .agg([
7         pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
8         pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
9         pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
10        pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
11        pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
12        pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
13    ])
14 ).collect()
15
16 df_polars.write_parquet('temp_polars.parquet')

<magic-timeit>:3: DeprecationWarning: `groupby` is deprecated. It has been renamed to `group_by`.
5.69 s ± 644 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 !ls -Gflash temp_polars.parquet

8.0K -rw-r--r-- 1 root 6.3K Jul 17 15:37 temp_polars.parquet
```

Archivos usados: 1, 4 y 5

RAM: 1.4 / 12.7 GB

NOTAS: Se ejecutaron todos los comandos correctamente

Ejecución: 46 SEG

PYSPARK

Prueba 1:

```
[16] 1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg, max, sum, concat

[17] 1 spark = SparkSession.builder.master("local[1]").appName("airline-example").getOrCreate()

[18] 1 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 flights_files = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"

[19] 1 df_spark1 = spark.read.parquet(flights_file1)
2 df_spark2 = spark.read.parquet(flights_file2)
3 df_spark3 = spark.read.parquet(flights_file3)
4 df_spark4 = spark.read.parquet(flights_file4)
5 df_spark5 = spark.read.parquet(flights_files)

[20] 1 df_spark = df_spark1 # Initialize df_spark with the first Dataframe
2 df_spark = df_spark.union(df_spark2)
3 df_spark = df_spark.union(df_spark3)
4 df_spark = df_spark.union(df_spark4)
5 df_spark = df_spark.union(df_spark5)

[21] 1 %timeit
2
3 df_spark_agg = df_spark.groupby("Airline", "Year").agg(
4     avg("ArrDelayMinutes").alias("avg_arr_delay"),
5     sum("ArrDelayMinutes").alias("sum_arr_delay"),
6     max("ArrDelayMinutes").alias("max_arr_delay"),
7     avg("DepDelayMinutes").alias("avg_dep_delay"),
8     sum("DepDelayMinutes").alias("sum_dep_delay"),
9     max("DepDelayMinutes").alias("max_dep_delay"),
10 )
11 df_spark_agg.write.mode("overwrite").parquet("temp_spark.parquet")

9.28 s ± 1.31 s per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 !ls -Gflash temp_pyspark.parquet
```

Archivos usados: Todas

RAM: 2.0 / 12.7 GB

NOTAS: Se ejecutaron todos los comandos exitosamente

Ejecución: 2.18 MIN

Prueba 2:

```
[16] 1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg, max, sum, concat

[17] 1 spark = SparkSession.builder.master("local[1]").appName("airline-example").getOrCreate()

[23] 1 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 # flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 # flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 # flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"

[24] 1 df_spark1 = spark.read.parquet(flights_file1)
2 df_spark2 = spark.read.parquet(flights_file2)
3 # df_spark3 = spark.read.parquet(flights_file3)
4 # df_spark4 = spark.read.parquet(flights_file4)
5 # df_spark5 = spark.read.parquet(flights_file5)

[25] 1 df_spark = df_spark1 # Initialize df_spark with the first Dataframe
2 df_spark = df_spark.union(df_spark2)
3 # df_spark = df_spark.union(df_spark3)
4 # df_spark = df_spark.union(df_spark4)
5 # df_spark = df_spark.union(df_spark5)

[26] 1 %timeit
2
3 df_spark_agg = df_spark.groupby("Airline", "year").agg(
4     avg("ArrDelayMinutes").alias("avg_arr_delay"),
5     sum("ArrDelayMinutes").alias("sum_arr_delay"),
6     max("ArrDelayMinutes").alias("max_arr_delay"),
7     avg("DepDelayMinutes").alias("avg_dep_delay"),
8     sum("DepDelayMinutes").alias("sum_dep_delay"),
9     max("DepDelayMinutes").alias("max_dep_delay"),
10 )
11 df_spark_agg.write.mode("overwrite").parquet("temp_spark.parquet")

4.49 s ± 478 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 !ls -l temp_spark.parquet

ls: cannot access 'temp_spark.parquet': No such file or directory
```

Archivos usados: 1 y 2

RAM: 2.0 / 12.7 GB

NOTAS: Se compilo de manera exitosa

Ejecución: 1.45 MIN

Prueba 3:

```
Stored in directory: /root/.cache/pip/wheels/11/80/3d/68/2c25ed30d08cc2f8d930e54521d4630e2f0d8674f0807f306
[15] Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

[16] 1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg, max, sum, concat

[17] 1 spark = SparkSession.builder.master("local[1]").appName("airline-example").getOrCreate()

[38] 1 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2018.parquet"
2 # flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2019.parquet"
3 # flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2020.parquet"
4 # flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2021.parquet"
5 # flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2_Data/Combined_Flights_2022.parquet"

[39] 1 df_spark1 = spark.read.parquet(flights_file1)
2 # df_spark2 = spark.read.parquet(flights_file2)
3 # df_spark3 = spark.read.parquet(flights_file3)
4 # df_spark4 = spark.read.parquet(flights_file4)
5 # df_spark5 = spark.read.parquet(flights_file5)

[40] 1 df_spark = df_spark1 # Initialize df_spark with the first Dataframe
2 # df_spark = df_spark.union(df_spark2)
3 # df_spark = df_spark.union(df_spark3)
4 # df_spark = df_spark.union(df_spark4)
5 # df_spark = df_spark.union(df_spark5)

[41] 1 %timeit
2
3 df_spark_agg = df_spark.groupby("Airline", "year").agg(
4     avg("ArrDelayMinutes").alias("avg_arr_delay"),
5     sum("ArrDelayMinutes").alias("sum_arr_delay"),
6     max("ArrDelayMinutes").alias("max_arr_delay"),
7     avg("DepDelayMinutes").alias("avg_dep_delay"),
8     sum("DepDelayMinutes").alias("sum_dep_delay"),
9     max("DepDelayMinutes").alias("max_dep_delay"),
10 )
11 df_spark_agg.write.mode("overwrite").parquet("temp_spark.parquet")

4.65 s ± 1.06 s per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 !ls -l temp_spark.parquet

ls: cannot access 'temp_spark.parquet': No such file or directory
```

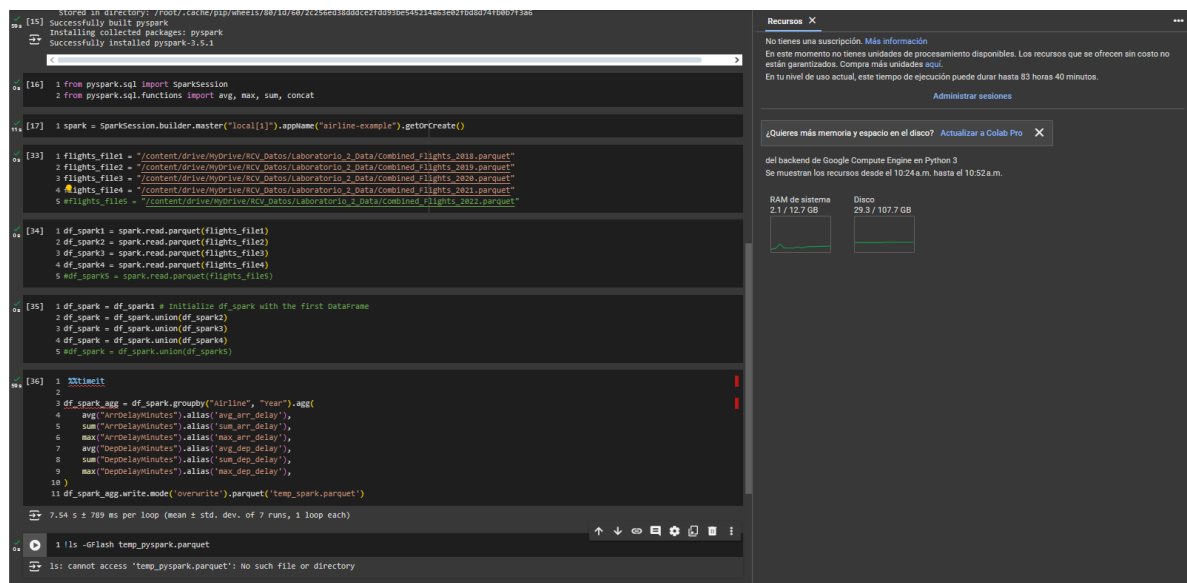
Archivos usados: 1, 3 y 5

RAM: 2.1 / 12.7 GB

NOTAS: Ejecución sin problemas

Ejecución: 1.48 MIN

Prueba 4:



```
[15] stores in directory: /root/.cache/pip/wheels/0b/10/0b/4250a3d0dce10030e945144b30b100d0/4f90b713d0
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1

[16] 1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import avg, max, sum, concat

[17] 1 spark = SparkSession.builder.master("local[1]").appName("airline-example").getOrCreate()

[23] 1 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2018.parquet"
2 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2019.parquet"
3 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2020.parquet"
4 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2021.parquet"
5 flights_files = [flights_file1, flights_file2, flights_file3, flights_file4]

[34] 1 df_spark1 = spark.read.parquet(flights_file1)
2 df_spark2 = spark.read.parquet(flights_file2)
3 df_spark3 = spark.read.parquet(flights_file3)
4 df_spark4 = spark.read.parquet(flights_file4)
5 df_sparks = spark.read.parquet(flights_files)

[35] 1 df_spark = df_spark1 # initialize df_spark with the first dataframe
2 df_spark = df_spark.union(df_spark2)
3 df_spark = df_spark.union(df_spark3)
4 df_spark = df_spark.union(df_spark4)
5 df_spark = df_spark.union(df_sparks)

[36] 1 %timeit
2
3 df_spark_agg = df_spark.groupby("Airline", "year").agg(
4     avg("ArrDelayMinutes").alias("avg_arr_delay"),
5     sum("ArrDelayMinutes").alias("sum_arr_delay"),
6     max("ArrDelayMinutes").alias("max_arr_delay"),
7     avg("DepDelayMinutes").alias("avg_dep_delay"),
8     sum("DepDelayMinutes").alias("sum_dep_delay"),
9     max("DepDelayMinutes").alias("max_dep_delay"),
10 )
11 df_spark_agg.write.mode('overwrite').parquet('temp_spark.parquet')

7.54 s ± 789 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

1 !ls -l temp_spark.parquet
ls: cannot access 'temp_spark.parquet': No such file or directory
```

Recursos

No tienes una suscripción. Más información

En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades aquí.

En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 83 horas 40 minutos.

Administrar sesiones

¿Quieres más memoria y espacio en el disco? Actualizar a Colab Pro

del backend de Google Compute Engine en Python 3

Se muestran los recursos desde el 10:24 a.m. hasta el 10:52 a.m.

RAM de sistema 2.1 / 12.7 GB

Disco 29.3 / 107.7 GB

Archivos usados: 1, 2, 3 y 4

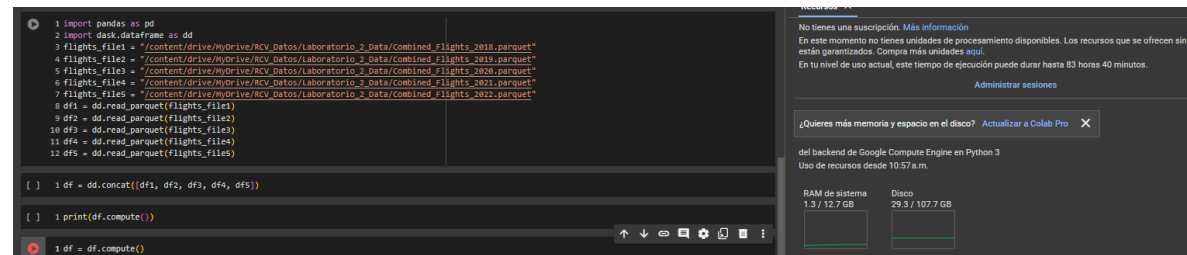
RAM: 2.1 / 12.7 GB

NOTAS: Ejecución sin problemas

Ejecución: 2.09 MIN

DASK

Intento 1:



```
1 import pandas as pd
2 import dask.dataframe as dd
3 flights_file1 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2018.parquet"
4 flights_file2 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2019.parquet"
5 flights_file3 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2020.parquet"
6 flights_file4 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2021.parquet"
7 flights_file5 = "/content/drive/MyDrive/RCV_Datos/Laboratorio_2/Data/combined_flights_2022.parquet"
8 df1 = dd.read_parquet(flights_file1)
9 df2 = dd.read_parquet(flights_file2)
10 df3 = dd.read_parquet(flights_file3)
11 df4 = dd.read_parquet(flights_file4)
12 df5 = dd.read_parquet(flights_file5)

1 df = dd.concat([df1, df2, df3, df4, df5])

1 print(df.compute())

1 df = df.compute()
```

Recursos

No tienes una suscripción. Más información

En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades aquí.

En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 83 horas 40 minutos.

Administrar sesiones

¿Quieres más memoria y espacio en el disco? Actualizar a Colab Pro

del backend de Google Compute Engine en Python 3

Uso de recursos desde 10:57 a.m.

RAM de sistema 1.3 / 12.7 GB

Disco 29.3 / 107.7 GB

Archivos usados: Todas

RAM: - / 12.7 GB

NOTAS: Se interrumpe en el df.compute por falta de recursos

Prueba 2:


```

1 import pandas as pd
2 import dask.dataframe as dd
3 flights_file1 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2018.parquet"
4 flights_file2 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2019.parquet"
5 flights_file3 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2020.parquet"
6 flights_file4 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2021.parquet"
7 flights_files = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2022.parquet"
8 df1 = dd.read_parquet(flights_file1)
9 df2 = dd.read_parquet(flights_file2)
10 df3 = dd.read_parquet(flights_file3)
11 df4 = dd.read_parquet(flights_file4)
12 df5 = dd.read_parquet(flights_files)

1 df = dd.concat([df1, df2, df4, df5])

1 print(df.compute())

1 df = df.compute()

1 * %timeit
2
3 df_agg = df.groupby(["Airline", "year"])[["DepDelayMinutes", "ArrDelayMinutes"]].agg(
4     ["mean", "sum", "max"]
5 )
6 df_agg = df_agg.reset_index()
7 df_agg.to_parquet("temp_dask.parquet")

```

Recursos

No tienes una suscripción. Más información

En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades aquí.

En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 83 horas 30 minutos.

Administrar sesiones

¿Quieres más memoria y espacio en el disco? Actualizar a Colab Pro

No hay conexión con el entorno de ejecución.

Cambiar tipo de entorno de ejecución

Archivos usados: 1, 2, 4 y 5

RAM: - / 12.7 GB

NOTAS: Se desbordó el código con esa cantidad

Ejecución: N/A

Prueba 3:

```

1 import pandas as pd
2 import dask.dataframe as dd
3 flights_file1 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2018.parquet"
4 flights_file2 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2019.parquet"
5 flights_file3 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2020.parquet"
6 flights_file4 = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2021.parquet"
7 flights_files = "/content/drive/MyDrive/RCV Datos/Laboratorio 2 Data/Combined_Flights_2022.parquet"
8 df1 = dd.read_parquet(flights_file1)
9 df2 = dd.read_parquet(flights_file2)
10 df3 = dd.read_parquet(flights_file3)
11 df4 = dd.read_parquet(flights_file4)
12 df5 = dd.read_parquet(flights_files)

1 df = dd.concat([df1, df2, df5])

1 print(df.compute())

1 df = df.compute()

1 * %timeit
2
3 df_agg = df.groupby(["Airline", "year"])[["DepDelayMinutes", "ArrDelayMinutes"]].agg(
4     ["mean", "sum", "max"]
5 )
6 df_agg = df_agg.reset_index()
7 df_agg.to_parquet("temp_dask.parquet")

```

Recursos

No tienes una suscripción. Más información

En este momento no tienes unidades de procesamiento disponibles. Los recursos que se ofrecen sin costo no están garantizados. Compra más unidades aquí.

En tu nivel de uso actual, este tiempo de ejecución puede durar hasta 83 horas 30 minutos.

Administrar sesiones

¿Quieres más memoria y espacio en el disco? Actualizar a Colab Pro

No hay conexión con el entorno de ejecución.

Cambiar tipo de entorno de ejecución

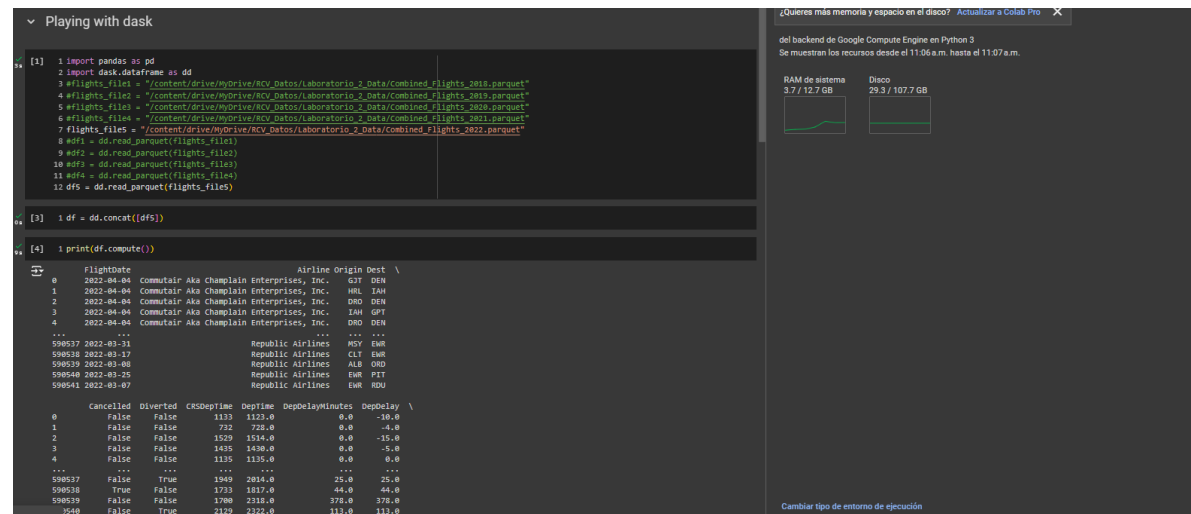
Archivos usados: 1, 2 y 5

RAM: - / 12.7 GB

NOTAS: Se desbordó el código con esa cantidad

Ejecución: N/A

Prueba 4:



Archivos usados: 5

RAM: 3.7 / 12.7 GB

NOTAS: Ejecución exitosa

Ejecución: 19 SEG

Conclusiones

- En la librería Pandas, en la mayoría de ejecuciones que se realizaban colapsaba por el tema de la RAM, pues en promedio ejecutaba hasta un máximo de 350 MB
- El tiempo de ejecución en todas las librerías era muy parecido, pero en pandas y en Dask en la mayoría de casos el código reventaba.
- El correr más de 2 conjuntos de datos en Pandas y Dask, el Colab está colapsando.
- El correr con DASK varios datos puede provocar en promedio más de 10 GB de consumo por ejecución
- Pandas es la opción más usada a pesar de las dificultades en la carga de datos y las limitaciones.
- Si se desea no exceder la RAM, es factible usar cualquiera que no sea pandas, ya que disminuye la cantidad de recursos consumidos operativos.