Laboratorio #2

Presentado por

Cristian Ballen Gamboa
Breyner Andreit Rincon Quiroga

Documentación

#1 Ejecución del archivo #2

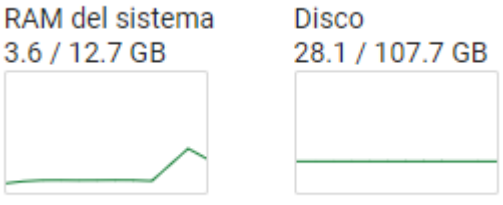
- Tiempo 27 segundos
- ram 4.7 gb

```
[1] import pandas as pd
    # flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
    flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
    # flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
    #flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
    # flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
    # df1 = pd.read_parquet(flights_file1)
    df2 = pd.read_parquet(flights_file2)
    # df3 = pd.read_parquet(flights_file3)
    # df4 = pd.read_parquet(flights_file4)
    # df5 = pd.read_parquet(flights_file5)
```

#2 Ejecución del archivo #3

- Tiempo 14 segundos
- Ram 3.6 gb

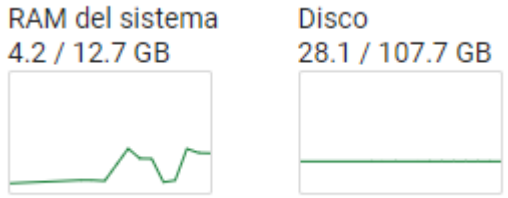
```
import pandas as pd
# flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
# flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
#flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
# flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
# df1 = pd.read_parquet(flights_file1)
# df2 = pd.read_parquet(flights_file2)
df3 = pd.read_parquet(flights_file3)
# df4 = pd.read_parquet(flights_file4)
# df5 = pd.read_parquet(flights_file5)
```



#3 Ejecución del archivo #4

- Tiempo 13 segundos
- Ram 4.2 gb

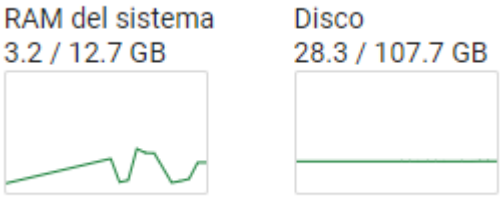
```
import pandas as pd
# flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
# flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
# flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
# flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
# df1 = pd.read_parquet(flights_file1)
# df2 = pd.read_parquet(flights_file2)
# df3 = pd.read_parquet(flights_file3)
df4 = pd.read_parquet(flights_file4)
# df5 = pd.read_parquet(flights_file5)
```



#4 Ejecución del archivo #5

- Tiempo 4 segundos
- Ram 3.2 gb

```
import pandas as pd
# flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
# flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
# flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
# flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
# df1 = pd.read_parquet(flights_file1)
# df2 = pd.read_parquet(flights_file2)
# df3 = pd.read_parquet(flights_file3)
# df4 = pd.read_parquet(flights_file4)
df5 = pd.read_parquet(flights_file5)
```



#5 concatenación del archivo 2 y 3 nos deja visualizar que la Ram no tiene suficiente espacio para realizar la ejecución y sólo se podrá ejecutar un archivo.

```
import pandas as pd
# flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
# flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
# flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
# df1 = pd.read_parquet(flights_file1)
df2 = pd.read_parquet(flights_file2)
df3 = pd.read_parquet(flights_file3)
# df4 = pd.read_parquet(flights_file4)
# df5 = pd.read_parquet(flights_file5)
```

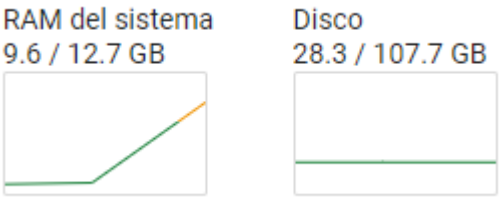
```
df = pd.concat([df2, df3])
#df = df2
```

6 Al realizar la concatenación del archivo 3 y 5 podemos visualizar que la ram cuenta con espacio suficiente para concatenar estos dos archivos, , (dependiendo el tamaño del archivo solo se podrá ejecutar si la ram tiene espacio suficiente).

Debido a que el archivo 3 y 5 nos ayuda a evidenciar que podemos realizar la concatenación de los dos archivos, realizaremos el ejercicio con este ejemplo.

```
[1] import pandas as pd
    # flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
    # flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
    flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
    # flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
    flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
    # df1 = pd.read_parquet(flights_file1)
    # df2 = pd.read_parquet(flights_file2)
    df3 = pd.read_parquet(flights_file3)
    # df4 = pd.read_parquet(flights_file4)
    df5 = pd.read_parquet(flights_file5)
```

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



El código agrupa los datos de vuelo por aerolínea y año, calcula varias estadísticas sobre los retrasos en la salida y llegada, luego guarda los resultados agrupados y calculados en un archivo Parquet llamado "temp_pandas.parquet".

```
# %%timeit

df_agg = df.groupby(['Airline','Year'])[["DepDelayMinutes", "ArrDelayMinutes"]].agg(
    ["mean", "sum", "max"]
)
df_agg = df_agg.reset_index()
df_agg.to_parquet("temp_pandas.parquet")
```

Se evidencia que la ejecución del código se demora 2 segundos.

```
!ls -GFlash temp_pandas.parquet
```

```
12K -rw-r--r-- 1 root 10K Jun 20 00:19 temp_pandas.parquet
```

lista detalladamente el archivo `temp_pandas.parquet` en el directorio actual.
se visualiza la informacion

```
[5] pd.read_parquet('temp_pandas.parquet')
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
1	Air Wisconsin Airlines Corp	2021	16.553045	1290194.0	1421.0	17.327440	1346602.0	1416.0
2	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
3	Alaska Airlines Inc.	2021	8.575051	1594042.0	938.0	9.507075	1761718.0	934.0
4	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0
5	Allegiant Air	2021	19.634153	2200694.0	1995.0	21.274423	2378523.0	1990.0
6	American Airlines Inc.	2020	7.624477	4084097.0	3890.0	7.861155	4202644.0	3864.0
7	American Airlines Inc.	2021	14.527001	10463784.0	3095.0	13.793588	9901465.0	3089.0
8	Capital Cargo International	2020	7.665063	512969.0	1482.0	8.427212	561522.0	1470.0
9	Capital Cargo International	2021	9.970415	978327.0	1808.0	10.777159	1052853.0	1797.0
10	Comair Inc.	2020	10.068723	1798294.0	1919.0	10.686808	1903235.0	1888.0
11	Comair Inc.	2021	10.456981	2305482.0	1838.0	10.883368	2392654.0	1869.0

carga el archivo Parquet `temp_pandas.parquet` en un DataFrame utilizando `pd.read_parquet()`, y luego llama al método `.info()` en ese DataFrame para imprimir un resumen de la estructura y la información básica

```
pd.read_parquet('temp_pandas.parquet').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   (Airline, )             47 non-null     object
 1   (Year, )                47 non-null     int64
 2   (DepDelayMinutes, mean) 47 non-null     float64
 3   (DepDelayMinutes, sum)  47 non-null     float64
 4   (DepDelayMinutes, max)  47 non-null     float64
 5   (ArrDelayMinutes, mean) 47 non-null     float64
 6   (ArrDelayMinutes, sum)  47 non-null     float64
 7   (ArrDelayMinutes, max)  47 non-null     float64
dtypes: float64(6), int64(1), object(1)
memory usage: 3.1+ KB
```

Pandas

Con Pandas logramos evidenciar que el uso de Ram (11.8) y el tiempo (63 segundos).

Playing with pandas

```
[1] import pandas as pd
    # flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
    # flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
    flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
    flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
    # flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
    # df1 = pd.read_parquet(flights_file1)
    # df2 = pd.read_parquet(flights_file2)
    df3 = pd.read_parquet(flights_file3)
    df4 = pd.read_parquet(flights_file4)
    # df5 = pd.read_parquet(flights_file5)
```

```
[2] df = pd.concat([df3, df4])
    #df = df2
```

```
[3] # %%timeit

    df_agg = df.groupby(['Airline','Year'])[["DepDelayMinutes", "ArrDelayMinutes"]].agg(
        ["mean", "sum", "max"]
    )
    df_agg = df_agg.reset_index()
    df_agg.to_parquet("temp_pandas.parquet")
```

```
[4] !ls -GFlash temp_pandas.parquet
```

```
12K -rw-r--r-- 1 root 10K Jun 20 01:31 temp_pandas.parquet
```

```
[5] pd.read_parquet('temp_pandas.parquet')
```

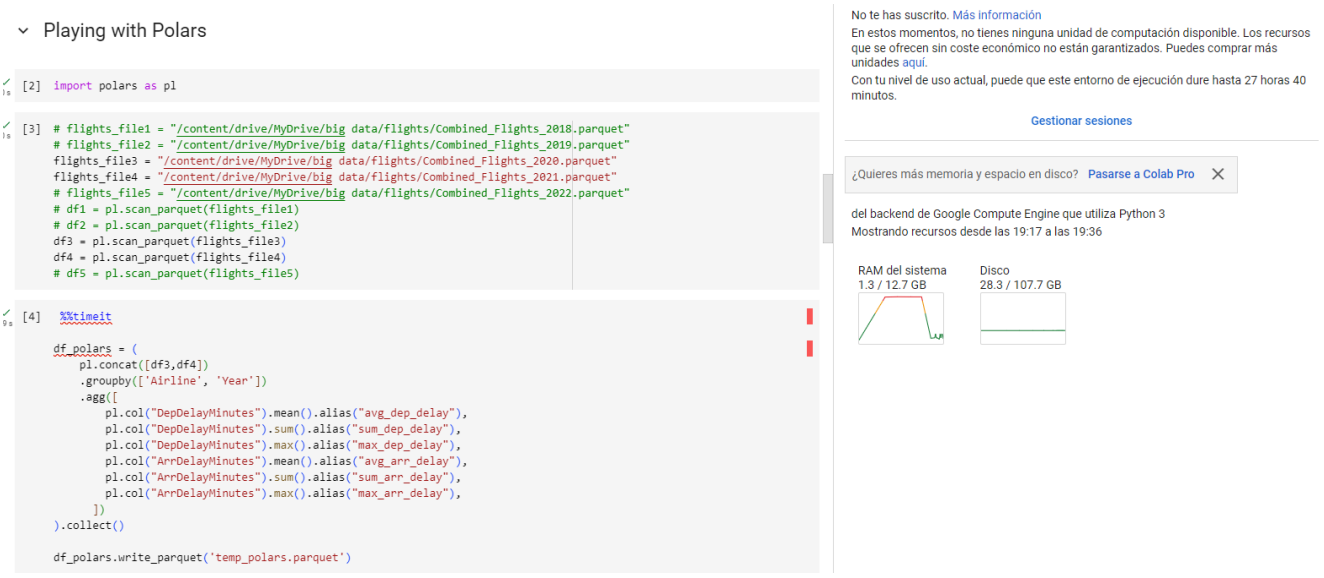
16	Delta Air Lines Inc.	2021	8.852645	6590555.0	1234.0	8.509149	6322706.0	1272.0
17	Empire Airlines Inc.	2020	6.861561	32613.0	274.0	7.028136	33222.0	272.0
18	Empire Airlines Inc.	2021	5.241071	587.0	85.0	5.250000	588.0	81.0
19	Endeavor Air Inc.	2020	5.653603	1156405.0	2579.0	5.877866	1200707.0	2560.0
40	Southwest Airlines Co.	2020	4.116516	3657115.0	902.0	3.972146	2975631.0	897.0
41	Southwest Airlines Co.	2021	14.802173	15413518.0	975.0	12.613220	13102108.0	630.0
42	Spirit Air Lines	2020	7.809889	1032194.0	1339.0	7.758486	1023290.0	1262.0
43	Spirit Air Lines	2021	14.462256	2686740.0	1503.0	14.593450	2703962.0	1519.0
44	Trans States Airlines	2020	13.739364	235108.0	1390.0	14.784864	252008.0	1366.0
45	United Air Lines Inc.	2020	6.925499	1978518.0	1471.0	6.766355	1929514.0	1503.0
46	United Air Lines Inc.	2021	12.196467	5377081.0	1581.0	12.238722	5380803.0	1655.0

```
pd.read_parquet('temp_pandas.parquet').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   (Airline, )              47 non-null     object
 1   (Year, )                 47 non-null     int64
 2   (DepDelayMinutes, mean)  47 non-null     float64
 3   (DepDelayMinutes, sum)   47 non-null     float64
 4   (DepDelayMinutes, max)   47 non-null     float64
 5   (ArrDelayMinutes, mean)  47 non-null     float64
 6   (ArrDelayMinutes, sum)   47 non-null     float64
 7   (ArrDelayMinutes, max)   47 non-null     float64
dtypes: float64(6), int64(1), object(1)
memory usage: 3.1+ KB
```

Polars

Con polars se evidencia que el uso de Ram (1.3gb) y de tiempo (9 segundos).



Playing with Polars

```
[2]  import polars as pl
```

```
[3]  # flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
     # flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
     flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
     flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
     # flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
     # df1 = pl.scan_parquet(flights_file1)
     # df2 = pl.scan_parquet(flights_file2)
     df3 = pl.scan_parquet(flights_file3)
     df4 = pl.scan_parquet(flights_file4)
     # df5 = pl.scan_parquet(flights_file5)
```

```
[4]  %%timeit

     df_polars = (
         pl.concat([df3,df4])
         .groupby(['Airline', 'Year'])
         .agg([
             pl.col("DepDelayMinutes").mean().alias("avg_dep_delay"),
             pl.col("DepDelayMinutes").sum().alias("sum_dep_delay"),
             pl.col("DepDelayMinutes").max().alias("max_dep_delay"),
             pl.col("ArrDelayMinutes").mean().alias("avg_arr_delay"),
             pl.col("ArrDelayMinutes").sum().alias("sum_arr_delay"),
             pl.col("ArrDelayMinutes").max().alias("max_arr_delay"),
           ])
     ).collect()

     df_polars.write_parquet('temp_polars.parquet')
```

No te has suscrito. Más información

En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 27 horas 40 minutos.

Gestionar sesiones

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

del backend de Google Compute Engine que utiliza Python 3

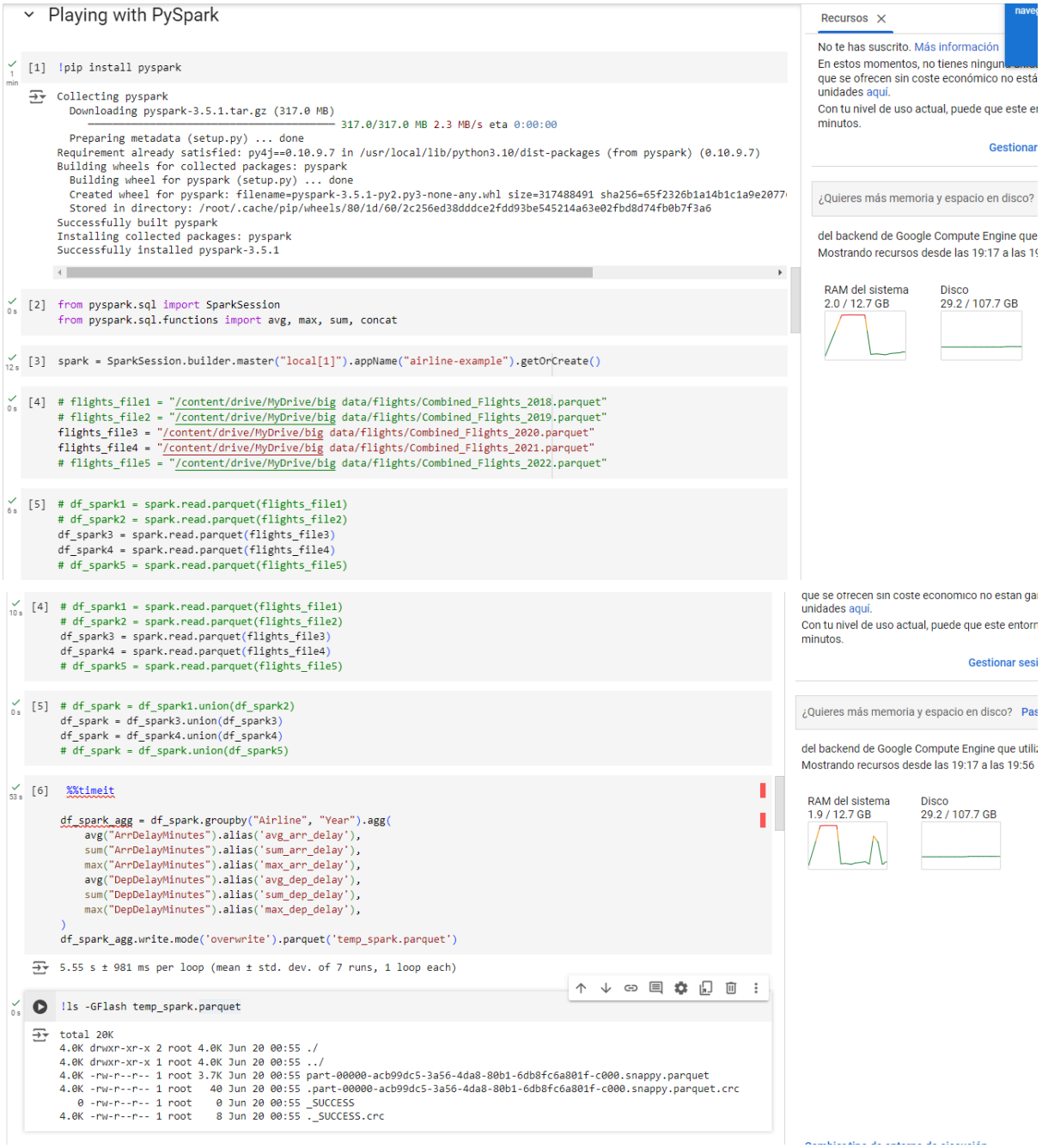
Mostrando recursos desde las 19:17 a las 19:36

RAM del sistema
1.3 / 12.7 GB

Disco
28.3 / 107.7 GB

PySpark

Con pyspark logramos evidenciar que el uso de Ram (2.0) y el tiempo (75 segundos)



dask

Con dask logramos evidenciar que el uso de Ram (7.2) y el tiempo (50 segundos)

Playing with dask

```
[2] import pandas as pd
    import dask.dataframe as dd
    # flights_file1 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2018.parquet"
    # flights_file2 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2019.parquet"
    flights_file3 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2020.parquet"
    flights_file4 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2021.parquet"
    # flights_file5 = "/content/drive/MyDrive/big data/flights/Combined_Flights_2022.parquet"
    # df1 = dd.read_parquet(flights_file1)
    # df2 = dd.read_parquet(flights_file2)
    df3 = dd.read_parquet(flights_file3)
    df4 = dd.read_parquet(flights_file4)
    # df5 = dd.read_parquet(flights_file5)
```

```
[3] df = dd.concat([df3, df4])
```

```
[4] print(df.compute())
```

```
        FlightDate               Airline Origin Dest  Cancelled  Diverted  \
0       2020-09-01          Comair Inc.    PHL  DAY      False     False
1       2020-09-02          Comair Inc.    PHL  DAY      False     False
2       2020-09-03          Comair Inc.    PHL  DAY      False     False
3       2020-09-04          Comair Inc.    PHL  DAY      False     False
4       2020-09-05          Comair Inc.    PHL  DAY      False     False
...            ...                  ...    ...  ...        ...       ...
573774 2021-06-01  Southwest Airlines Co.    BNA  MDW      False     False
573775 2021-06-01  Southwest Airlines Co.    BNA  MDW      False     False
573776 2021-06-01  Southwest Airlines Co.    BNA  MIA      False     False
573777 2021-06-01  Southwest Airlines Co.    BNA  MIA      False     False
573778 2021-06-01  Southwest Airlines Co.    BNA  MKE      False     False

        CRSDepTime  DepTime  DepDelayMinutes  DepDelay  ...  WheelsOff  \
0             1905   1858.0              0.0      -7.0  ...     1914.0
1             1905   1858.0              0.0      -7.0  ...     1914.0
2             1905   1855.0              0.0     -10.0  ...     2000.0
3             1905   1857.0              0.0      -8.0  ...     1910.0
4             1905   1856.0              0.0      -9.0  ...     1910.0
...            ...      ...              ...       ...  ...        ...
573774        1255   1301.0              6.0       6.0  ...     1310.0
573775         730    727.0              0.0      -3.0  ...      740.0
573776         800    757.0              0.0      -3.0  ...      811.0
```

Recursos

No te has suscrito. Más información

Aceptar | No, gracias

En estos momentos, no tienes ningunas unidades de computación disponibles con recursos que se ofrecen sin coste económico no están garantizados. Puedes comprar más unidades aquí.

Con tu nivel de uso actual, puede que este entorno de ejecución dure hasta 27 horas minutos.

Gestionar sesiones

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

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 19:17 a las 19:50

RAM del sistema
7.2 / 12.7 GB

Disco
29.2 / 107.7 GB

Cambiar tipo de entorno de ejecución

```
[5] df = df.compute()
```

34 s

```
[6] # %%timeit

    df_agg = df.groupby(['Airline','Year'])[["DepDelayMinutes", "ArrDelayMinutes"]].agg(
        ["mean", "sum", "max"]
    )
    df_agg = df_agg.reset_index()
    df_agg.to_parquet("temp_dask.parquet")
```

1 s

```
[7] !ls -GFlash temp_pandas.parquet
```

0 s

```
12K -rw-r--r-- 1 root 10K Jun 20 00:19 temp_pandas.parquet
```

```
[8] pd.read_parquet('temp_dask.parquet').info()
```

0 s

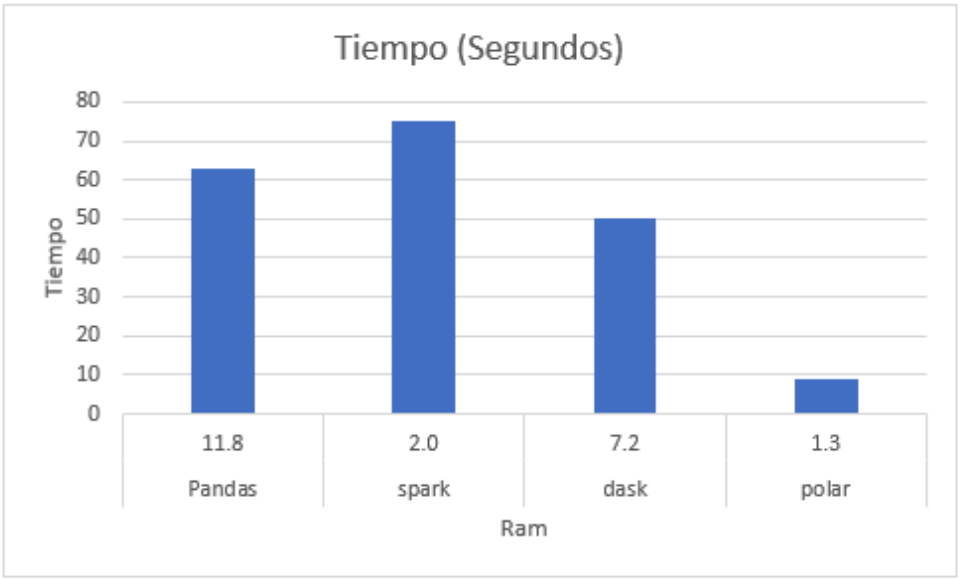
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   (Airline, )               47 non-null     string
 1   (Year, )                  47 non-null     int64
 2   (DepDelayMinutes, mean)   47 non-null     float64
 3   (DepDelayMinutes, sum)    47 non-null     float64
 4   (DepDelayMinutes, max)    47 non-null     float64
 5   (ArrDelayMinutes, mean)   47 non-null     float64
 6   (ArrDelayMinutes, sum)    47 non-null     float64
 7   (ArrDelayMinutes, max)    47 non-null     float64
dtypes: float64(6), int64(1), string(1)
memory usage: 3.1 KB
```

```
pd.read_parquet('temp_dask.parquet')
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
1	Air Wisconsin Airlines Corp	2021	16.553045	1290194.0	1421.0	17.327440	1346602.0	1416.0
2	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
3	Alaska Airlines Inc.	2021	8.575051	1594042.0	938.0	9.507075	1761718.0	934.0
4	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0
5	Allegiant Air	2021	19.634153	2200694.0	1995.0	21.274423	2378523.0	1990.0
6	American Airlines Inc.	2020	7.624477	4084097.0	3890.0	7.861155	4202644.0	3864.0
7	American Airlines Inc.	2021	14.527001	10463784.0	3095.0	13.793588	9901465.0	3089.0
8	Capital Cargo International	2020	7.665063	512969.0	1482.0	8.427212	561522.0	1470.0
9	Capital Cargo International	2021	9.970415	978327.0	1808.0	10.777159	1052853.0	1797.0
10	Comair Inc.	2020	10.068723	1798294.0	1919.0	10.686808	1903235.0	1888.0
11	Comair Inc.	2021	10.456981	2305482.0	1838.0	10.883368	2392654.0	1869.0
12	Commutair Aka Champlain Enterprises, Inc.	2020	12.266858	385670.0	1557.0	13.438158	421340.0	1555.0

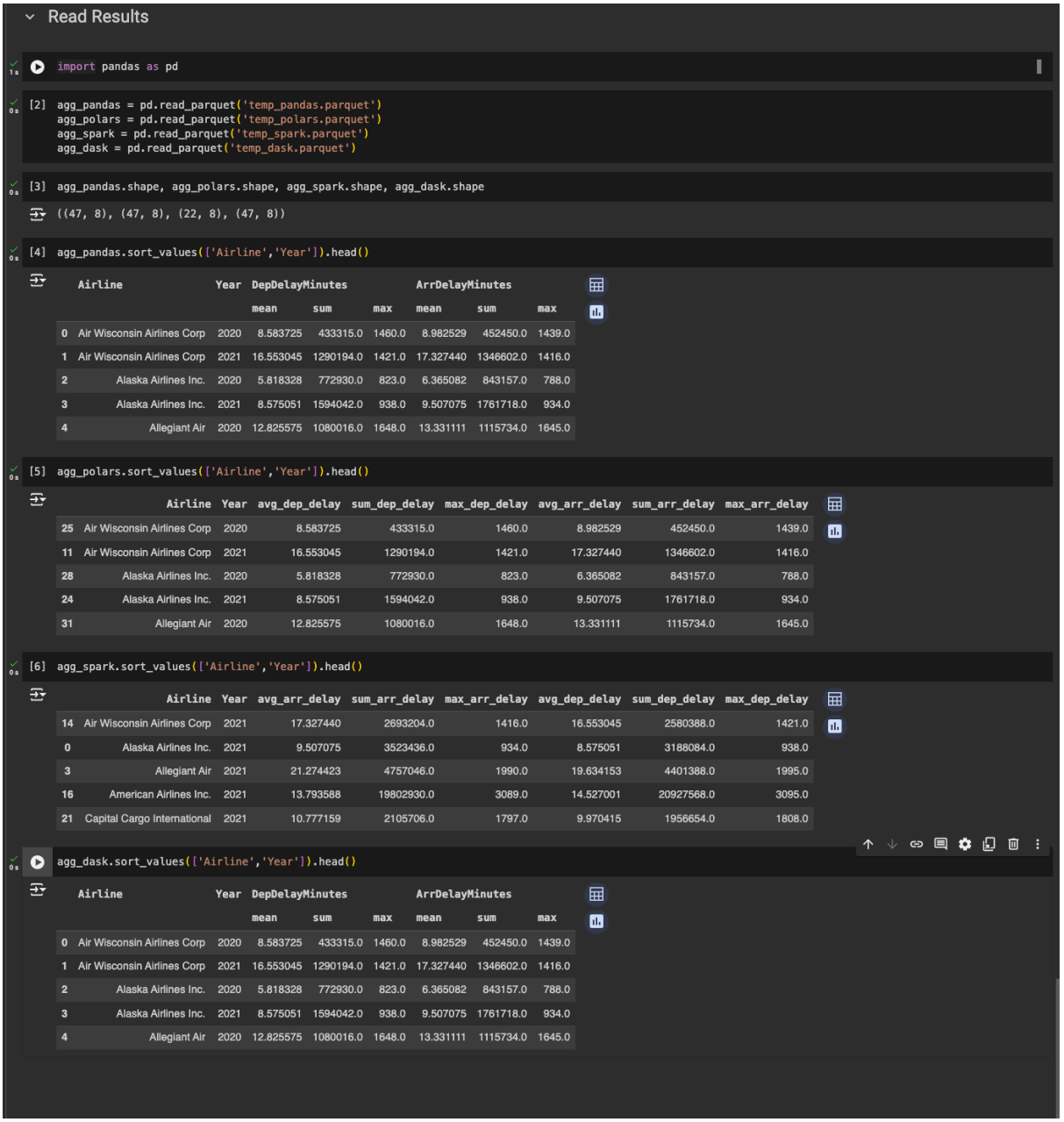
Resultados

Librería	Ram	Tiempo (Segundos)
Pandas	11.8	63
spark	2.0	75
dask	7.2	50
polar	1.3	9



Logramos evidenciar que polars es significativamente mejor para la lectura de los datos dando mejores resultados en tiempo y uso de ram.

Al visualizar la lectura de los archivos podemos observar que el resultado de pandas, polars y dask son iguales, el único que difiere de resultados es spark, de igual manera los encabezados suelen cambiar según la librería



Read Results

```
import pandas as pd
```

```
[2] agg_pandas = pd.read_parquet('temp_pandas.parquet')
    agg_polars = pd.read_parquet('temp_polars.parquet')
    agg_spark = pd.read_parquet('temp_spark.parquet')
    agg_dask = pd.read_parquet('temp_dask.parquet')
```

```
[3] agg_pandas.shape, agg_polars.shape, agg_spark.shape, agg_dask.shape
```

```
((47, 8), (47, 8), (22, 8), (47, 8))
```

```
[4] agg_pandas.sort_values(['Airline','Year']).head()
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
1	Air Wisconsin Airlines Corp	2021	16.553045	1290194.0	1421.0	17.327440	1346602.0	1416.0
2	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
3	Alaska Airlines Inc.	2021	8.575051	1594042.0	938.0	9.507075	1761718.0	934.0
4	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0

```
[5] agg_polars.sort_values(['Airline','Year']).head()
```

	Airline	Year	avg_dep_delay	sum_dep_delay	max_dep_delay	avg_arr_delay	sum_arr_delay	max_arr_delay
25	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
11	Air Wisconsin Airlines Corp	2021	16.553045	1290194.0	1421.0	17.327440	1346602.0	1416.0
28	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
24	Alaska Airlines Inc.	2021	8.575051	1594042.0	938.0	9.507075	1761718.0	934.0
31	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0

```
[6] agg_spark.sort_values(['Airline','Year']).head()
```

	Airline	Year	avg_arr_delay	sum_arr_delay	max_arr_delay	avg_dep_delay	sum_dep_delay	max_dep_delay
14	Air Wisconsin Airlines Corp	2021	17.327440	2693204.0	1416.0	16.553045	2580388.0	1421.0
0	Alaska Airlines Inc.	2021	9.507075	3523436.0	934.0	8.575051	3188084.0	938.0
3	Allegiant Air	2021	21.274423	4757046.0	1990.0	19.634153	4401388.0	1995.0
16	American Airlines Inc.	2021	13.793588	19802930.0	3089.0	14.527001	20927568.0	3095.0
21	Capital Cargo International	2021	10.777159	2105706.0	1797.0	9.970415	1956654.0	1808.0

```
agg_dask.sort_values(['Airline','Year']).head()
```

	Airline	Year	DepDelayMinutes			ArrDelayMinutes		
			mean	sum	max	mean	sum	max
0	Air Wisconsin Airlines Corp	2020	8.583725	433315.0	1460.0	8.982529	452450.0	1439.0
1	Air Wisconsin Airlines Corp	2021	16.553045	1290194.0	1421.0	17.327440	1346602.0	1416.0
2	Alaska Airlines Inc.	2020	5.818328	772930.0	823.0	6.365082	843157.0	788.0
3	Alaska Airlines Inc.	2021	8.575051	1594042.0	938.0	9.507075	1761718.0	934.0
4	Allegiant Air	2020	12.825575	1080016.0	1648.0	13.331111	1115734.0	1645.0

Conclusión

Según la ejecución de todos los archivos y códigos logramos evidenciar que polars es el que tiene mejor rendimiento, ocupando menos ram y menor tiempo en ejecución.