

SEGUNDO TALLER

Presentado por:

VALENTINA PIEDRAHITA GIL

JHOJAN STIVEN SANCHEZ PALADINES

Presentado a:

Ing ELIAS BUITRAGO BOLIVAR

Universidad ECCI

Ingeniería en Sistemas

Seminario Big Data & Gerencia de datos

Bogotá

2024

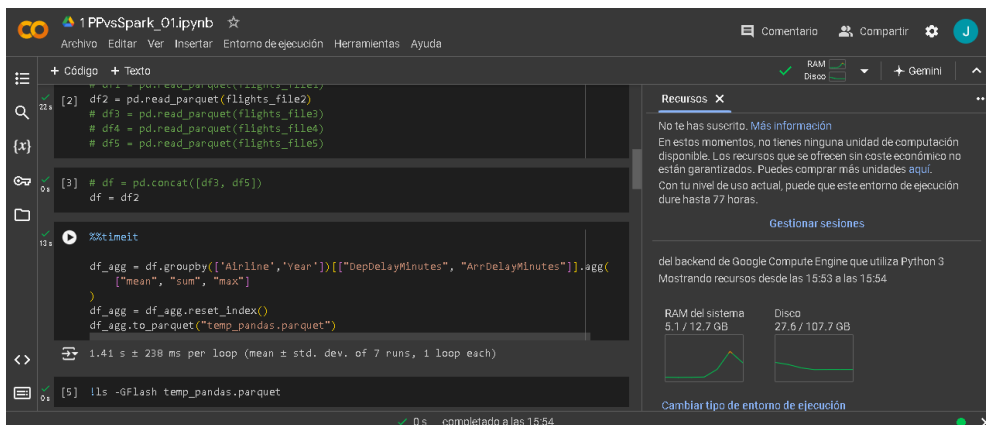
INTRODUCCIÓN:

En este taller, llevaremos a cabo un experimento utilizando las librerías Pandas, Polars, Spark y Dask para procesar cinco archivos de datos. Nuestro objetivo principal es determinar cuál de estas librerías ofrece un rendimiento más eficiente en términos de procesamiento de datos.

CONTENIDO:

1. Cargaremos el primer archivo más pesado para analizar el comportamiento y rendimiento de cada una de las librerías.

PANDAS



```
df1 = pd.read_parquet('flights_file1.parquet')
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

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')
```

1.41 s ± 238 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

0 s completado a las 15:54

Recursos

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 77 horas.

Gestionar sesiones

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 15:53 a las 15:54

RAM del sistema 5.1 / 12.7 GB

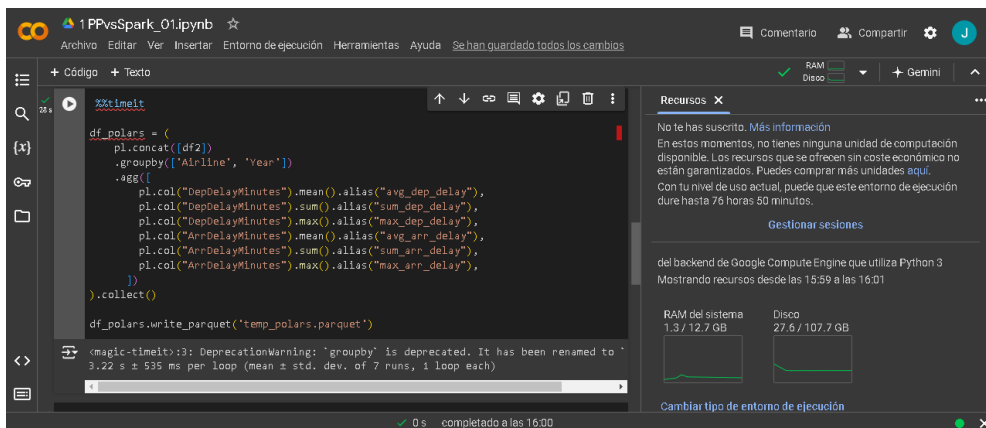
Disco 27.6 / 107.7 GB

Cambiar tipo de entorno de ejecución

Tiempo: 1.41 s ± 238 ms

RAM: 5.1 GB

POLARS



```
df_polars = (
    pl.concat([df2])
    .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')
```

(magic-timet): DeprecationWarning: 'groupby' is deprecated. It has been renamed to 'with_row_index'. 3.22 s ± 535 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

0 s completado a las 16:00

Recursos

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 76 horas 50 minutos.

Gestionar sesiones

del backend de Google Compute Engine que utiliza Python 3

Mostrando recursos desde las 15:59 a las 16:01

RAM del sistema 1.3 / 12.7 GB

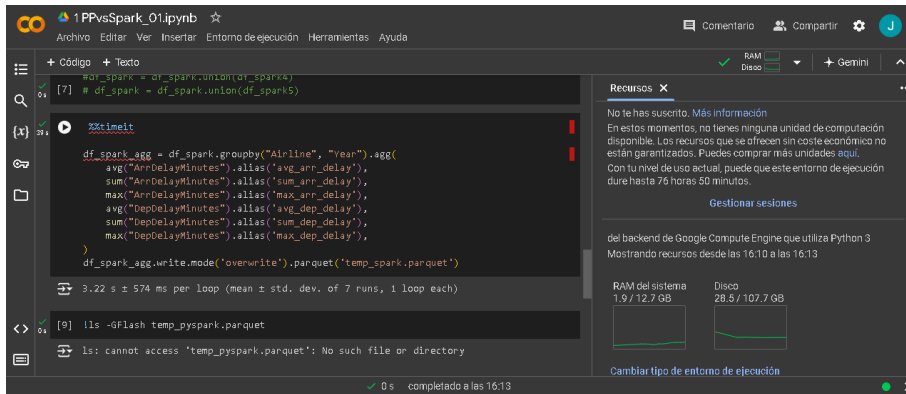
Disco 27.6 / 107.7 GB

Cambiar tipo de entorno de ejecución

Tiempo: 3.22 s ± 535 ms

RAM: 1.3 GB

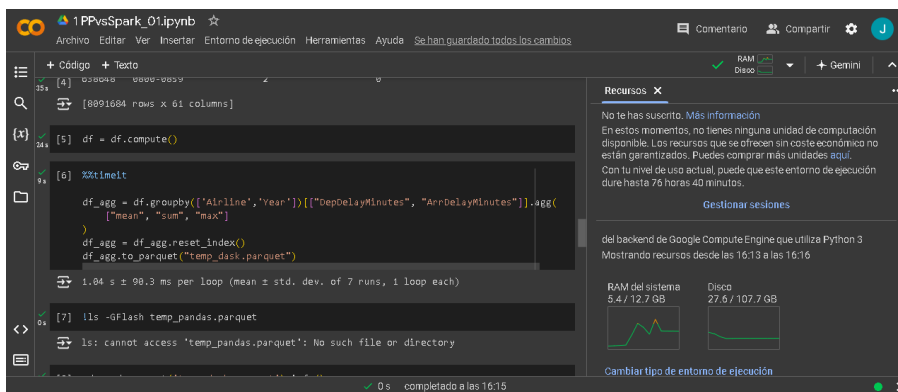
PYSPARK



Tiempo: 3.22 s ± 574 ms

RAM: 1.9 GB

DASK

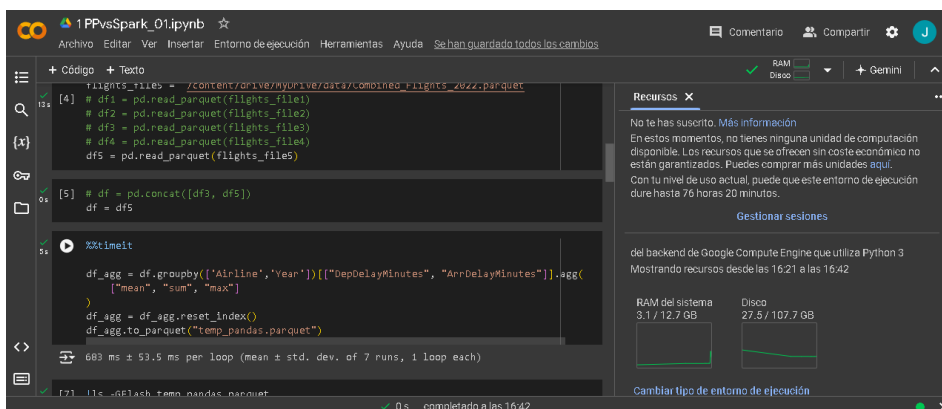


Tiempo: 1.04 s ± 90.3 ms

RAM: 5.4 GB

2. Ahora cargaremos el archivo menos pesado.

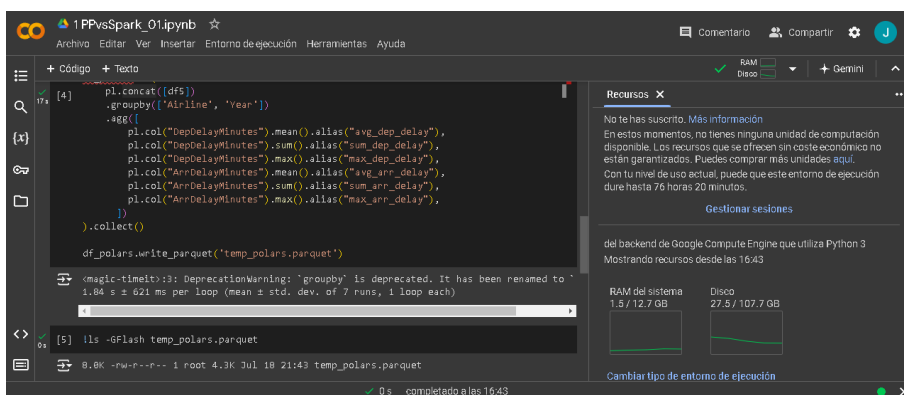
PANDAS



Tiempo: 683 ms ± 53.5 ms

RAM: 3.1 GB

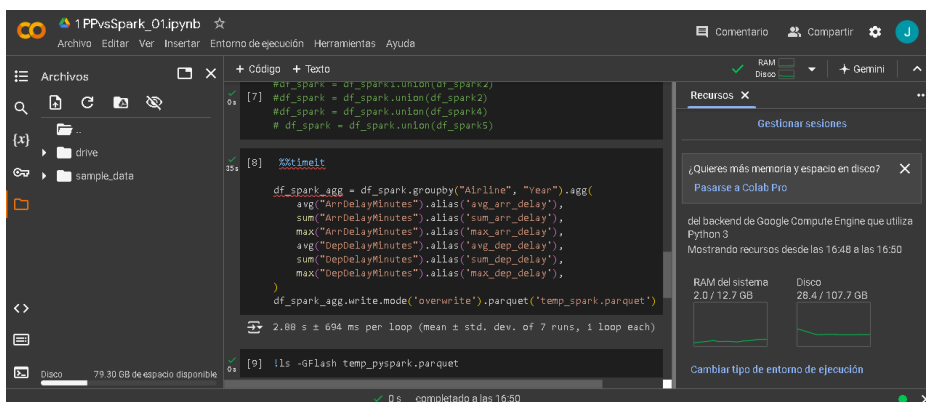
POLARS



Tiempo: 1.84 s ± 621 ms

RAM: 1.5 GB

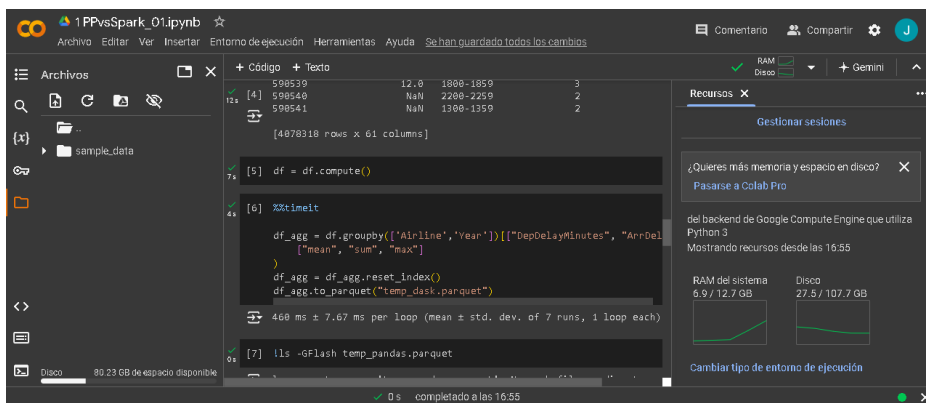
PYSPARK



Tiempo: 2.88 s ± 694 ms

RAM: 2.0 GB

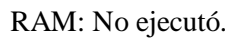
DASK



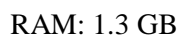
Tiempo: 460 ms ± 7.67 ms

RAM: 6.9 GB

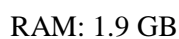
PANDAS



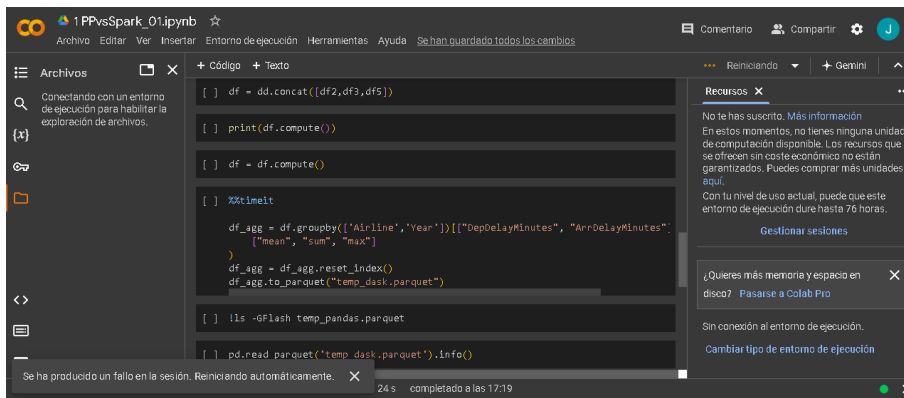
POLARS



PYSPARK



DASK

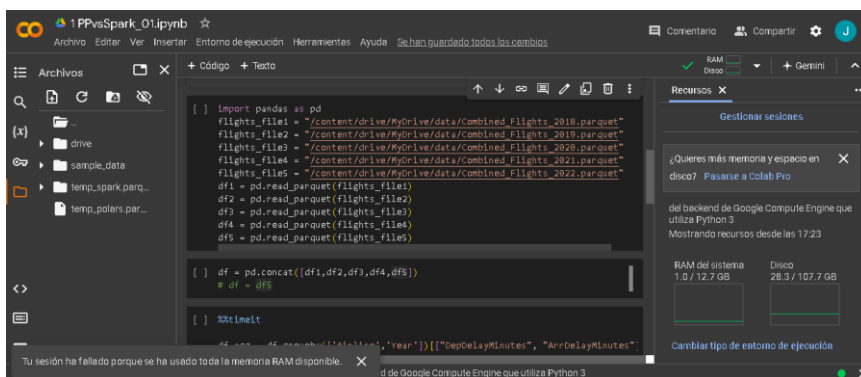


Tiempo: No ejecutó.

RAM: No ejecutó.

4. Probaremos con los 4 archivos.

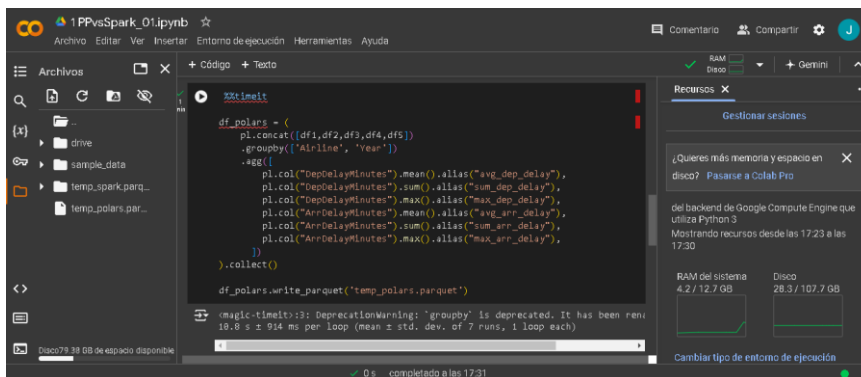
PANDAS



Tiempo: No ejecutó.

RAM: No ejecutó.

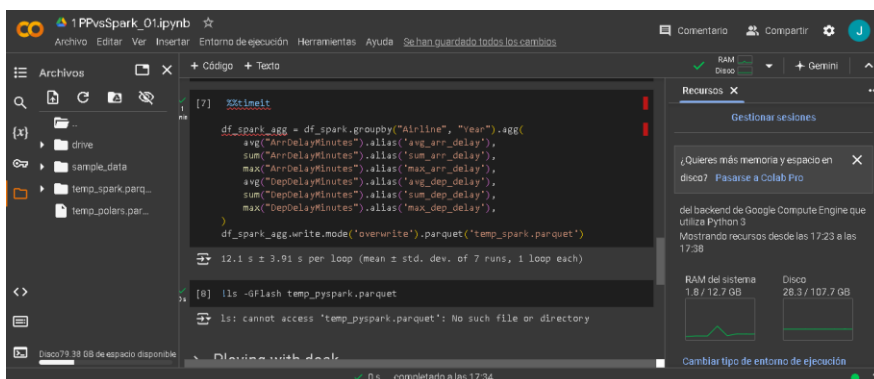
POLARS



Tiempo: 10.8 s ± 914 ms

RAM: 4.2 GB

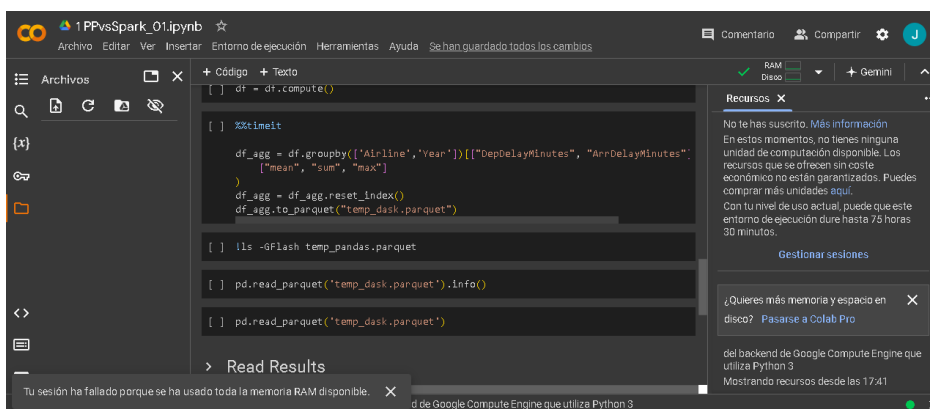
PYSPARK



Tiempo: 12.1 s ± 3.91 s

RAM: 1.8 GB

DASK



Tiempo: No ejecutó.

RAM: No ejecutó.

CONCLUSIONES

1. Panda fue eficiente al procesar archivos de menor tamaño, mostrando tiempos de carga rápidos y uso moderado de RAM. Sin embargo, su rendimiento disminuyó con archivos grandes, siendo incapaz de completar la ejecución cuando se probaron tres o más archivos simultáneamente.
2. Polars demostró un uso de RAM consistentemente bajo en comparación con las otras librerías. Aunque los tiempos de procesamiento fueron más lentos que Pandas y Dask para archivos pequeños, Polars fue capaz de manejar múltiples archivos y archivos más grandes sin problemas.
3. PySpark mostró tiempos de procesamiento estables y un uso de RAM bajo. Fue capaz de procesar archivos de todos los tamaños, incluyendo múltiples archivos simultáneamente. PySpark demostró ser una opción robusta para el procesamiento de grandes volúmenes de datos.
4. Dask fue la librería más rápida en términos de tiempo de carga para archivos individuales, tanto pequeños como grandes. Sin embargo, su uso de RAM fue significativamente mayor, y al igual que Pandas, no pudo completar la ejecución cuando se probaron múltiples archivos simultáneamente.