PROYECTO : *Recomendador de App*
ASIGNATURA: *Cloud y Big Data*

Alessio Cimino, Luca Neve

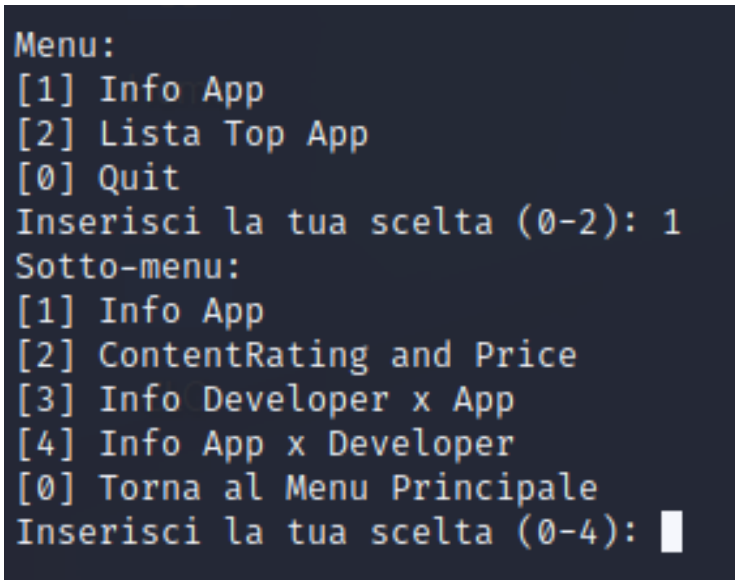
El proyecto que desarrollamos durante este curso tiene como objetivo proporcionar a los usuarios una herramienta que les permita filtrar las aplicaciones especificando parámetros según la solicitud. Cada día, de hecho, se crean y añaden nuevas aplicaciones con una siempre más creciente facilidad.

Hemos utilizado un archivo que contiene más de 2,3 millones de aplicaciones, desarrolladas para el sistema operativo Android y descargables a través de la plataforma Google Play.
El archivo está actualizado hasta junio de 2021 y representa un conjunto de datos de Big Data indispensable para procesar una gran cantidad de información, como en este caso específico.

Este archivo contiene, para cada línea, las siguientes 23 características:

- App Name
- App Id
- Category
- Rating
- Rating Count
- Installs
- Minimum Installs
- Maximum Installs
- Free
- Price
- Currency
- Size
- Minimum Android
- Developer Id
- Developer Website
- Developer Email
- Released
- Privacy Policy
- Last Updated
- Content Rating
- Ad Supported
- In app purchases
- Editor Choice

Decidimos desarrollar una interfaz gráfica simple e intuitiva,a causa de la enorme cantidad de datos. Por esta razón, dividimos los métodos en dos partes, como se muestra en la figura:



La primera parte de los métodos genera una salida directamente en el monitor, seleccionando una de las opciones del submenú que se muestran.

La segunda opción, en cambio, genera, una vez ejecutada, un archivo .csv con un número de líneas que debe ser especificado por el usuario en un rango que va de 1 a 1000. El usuario puede especificar directamente las subclases de interés, aplicando filtros antes de lanzar la consulta. Hemos tomado esta decisión tanto para desarrollar de manera óptima las líneas de código implementadas, como para hacer más eficiente la ejecución, evitando ocupar innecesariamente recursos computacionales y espacio de memoria física.

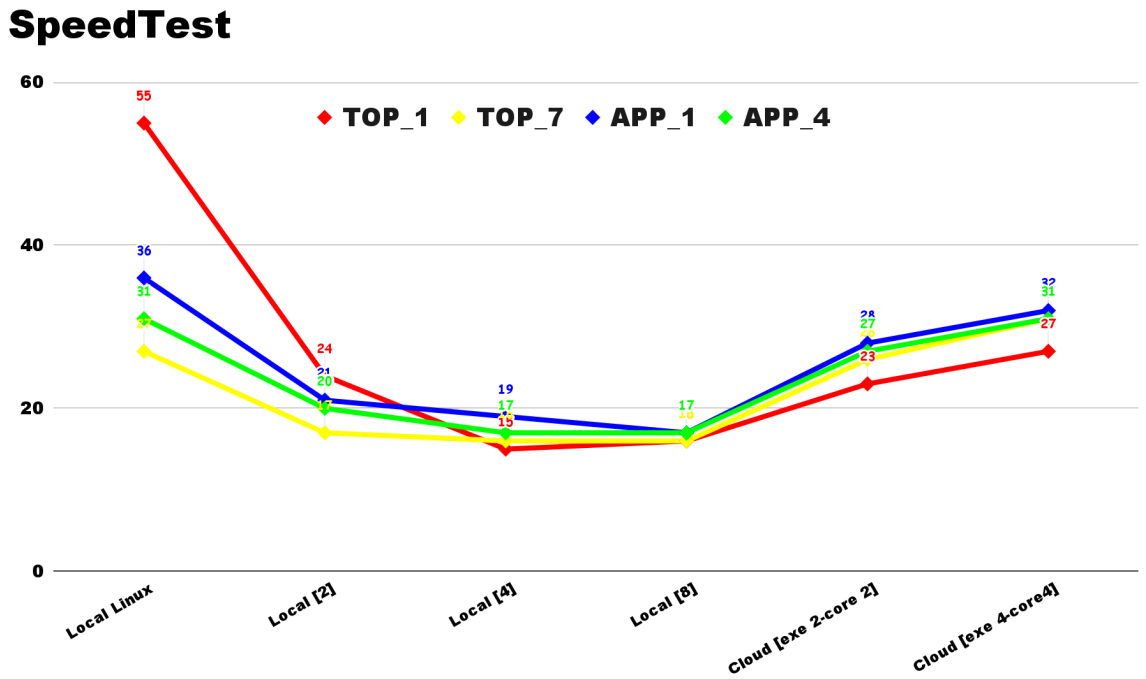
Los métodos fueron probados inicialmente en una máquina virtual Kali Linux en Oracle instalada en un computador portátil con las siguientes características hardware:

- Procesador AMD 3020e (2C / 2T, 1.2 / 2.6GHz, 1MB L2 / 4MB L3)
- Memoria RAM de 8GB SO-DIMM DDR4-2400
- Almacenamiento SSD de 256GB M.2 2242 PCIe NVMe 3.0x2
- Tarjeta gráfica integrada AMD Radeon Graphics
- Sistema operativo: Windows 10

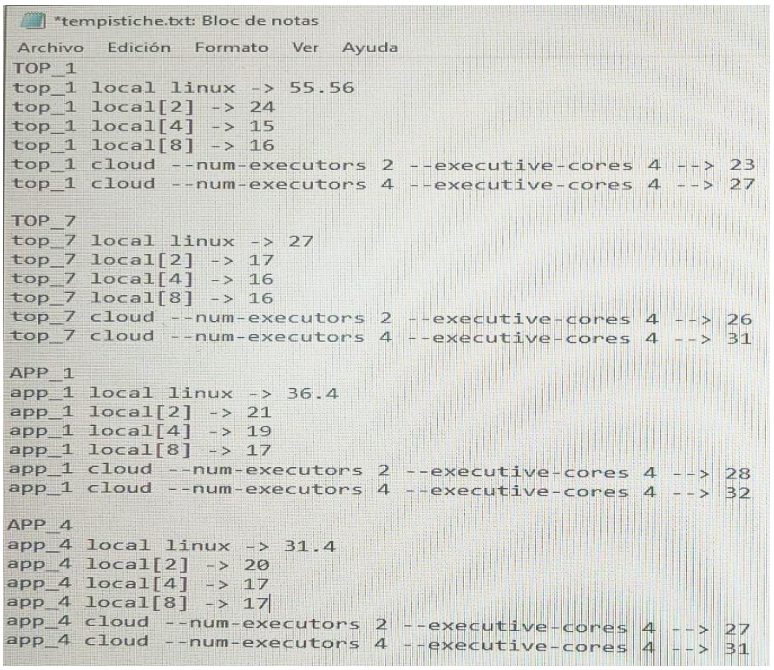
Posteriormente, utilizamos las herramientas presentadas durante el curso para ejecutar los diversos métodos directamente en modo Cloud; para lograrlo, tuvimos que modificar una pequeña parte del código, eliminando la parte de menú interactivo.

Rendimiento:

tiempos de ejecución - Local Virtual Machine y Google Cloud



Como se puede observar en el gráfico presentado, los tiempos de ejecución en la forma local disminuyen significativamente a medida que aumenta el número de núcleos utilizados. Naturalmente, los tiempos representados por la ejecución en la nube son más altos en comparación con el local debido al tiempo necesario para cargar los datos.



*tempistiche.txt: Bloc de notas

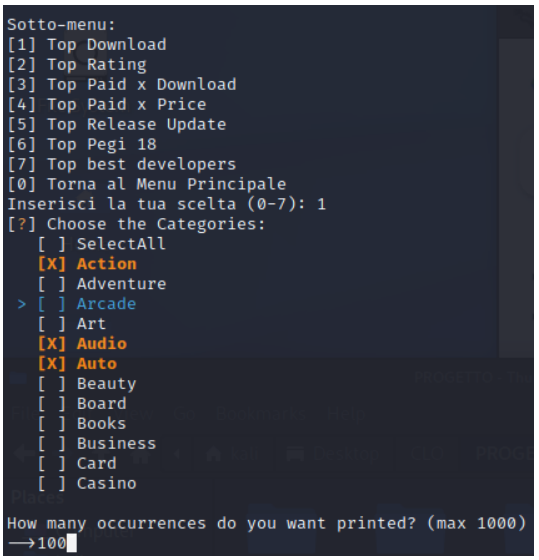
Archivo Edición Formato Ver Ayuda

```
TOP_1
top_1 local linux -> 55.56
top_1 local[2] -> 24
top_1 local[4] -> 15
top_1 local[8] -> 16
top_1 cloud --num-executors 2 --executive-cores 4 --> 23
top_1 cloud --num-executors 4 --executive-cores 4 --> 27

TOP_7
top_7 local linux -> 27
top_7 local[2] -> 17
top_7 local[4] -> 16
top_7 local[8] -> 16
top_7 cloud --num-executors 2 --executive-cores 4 --> 26
top_7 cloud --num-executors 4 --executive-cores 4 --> 31

APP_1
app_1 local linux -> 36.4
app_1 local[2] -> 21
app_1 local[4] -> 19
app_1 local[8] -> 17
app_1 cloud --num-executors 2 --executive-cores 4 --> 28
app_1 cloud --num-executors 4 --executive-cores 4 --> 32

APP_4
app_4 local linux -> 31.4
app_4 local[2] -> 20
app_4 local[4] -> 17
app_4 local[8] -> 17
app_4 cloud --num-executors 2 --executive-cores 4 --> 27
app_4 cloud --num-executors 4 --executive-cores 4 --> 31
```

La infraestructura que aprovechamos para desarrollar el proyecto incluye varias herramientas informáticas, como PySpark con muchas bibliotecas esenciales para analizar diferentes tipos de datos (cadenas de texto, enteros, fechas, boolean, etc.), Google Cloud y, por supuesto, Excel para verificar la precisión de los datos extraídos.

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, unix_timestamp, datediff, current_date
from pyspark.sql.types import TimestampType

import sys

def generate_top_lastUpdated_csv(listaGeneros, output_path, number_choice):
    spark = SparkSession.builder.appName("ToplastUpdated").getOrCreate()

    file_path = "Google-Playstore.csv"
    df = spark.read.option("header", "true").csv(file_path)

    filtered_df = df.filter(col("Category").isin(listaGeneros))
    filtered_df = filtered_df.filter(col("Last Updated").isNotNull() & (col("Last Updated") ≠ ""))
    date_format = "MMM d, yyyy"
    filtered_df = filtered_df.withColumn("Last Updated", unix_timestamp(col("Last Updated"), date_format).cast(TimestampType()))
    filtered_df = filtered_df.withColumn("DateDifference", datediff(current_date(), col("Last Updated")))
    sorted_df = filtered_df.orderBy("DateDifference")

    top_df = sorted_df.limit(number_choice)
    result_df = top_df.select(
        "App Name",
        "Released",
        col("Last Updated").cast("date").alias("Last Updated"),
        "DateDifference"
    )

    result_df.write.option("header", "true").csv(output_path, mode="overwrite")

    spark.stop()
    print(f"File CSV '{output_path}' generato con successo.")

print("Argomenti di input:", sys.argv)
listaGeneros = sys.argv[1].split(',')
number_choice = int(sys.argv[2])

if not listaGeneros:
    print("Specifica almeno un genere.")
    sys.exit(1)

output_path = "output_top/Top_lastUpdated.csv"
generate_top_lastUpdated_csv(listaGeneros, output_path, number_choice)
```

El menú de usuario que decidimos implementar se basa en una interfaz simple e intuitiva, ligera y esencial, como se muestra en la figura, donde es posible especificar la/las categoría/s de interés y el número de líneas de salida deseadas (max 1000).



Los resultados obtenidos, más allá de los tiempos de ejecución que pueden variar según la configuración de hardware, muestran un resultado acorde con nuestras expectativas iniciales de proyecto: las salidas procesadas, ya sea en formato .csv o directamente en monitor, constituyen una base sobre la cual será posible implementar, en un segundo momento, métodos adicionales que se vuelvan necesarios para optimizar un sistema de consulta de este tipo, actualizando, por supuesto, el DataSet utilizado.