Análisis de Datos en un Entorno de Big Data: Linux (Awk, Sort, Grep) frente a Python en Google Colab

Juan Canteros Murcia, Sergio Lopez, Brisa Rios, Gabriel Román, Leopoldo Rios[1]

[1] Dpto. Informática, Fac. de Ciencias Exactas Naturales y Agrimensura, Universidad Nacional del Nordeste, Corrientes.

canterosmurciajuan@gmail.com, se.lopez87@gmail.com, brisaabigail16@gmail.com, gbrielroman98@gmail.com, ljr@comunidad.unne.edu.ar

Abstract: En un entorno tecnológico que exige eficiencia en la gestión y análisis de datos, esta investigación compara herramientas clave en sistemas Linux, como awk, sort y sed, con el uso de Python en Google Colab. Se centra en la optimización del procesamiento de datos en ambos contextos. El estudio abarca la creación y configuración detallada de una máquina virtual Linux, destacando aplicaciones y herramientas para el análisis de datos. Además, explora la captura de datos desde Wireshark, presentando un escenario real de análisis de datos. Se analizan herramientas de línea de comandos Linux y Python en Google Colab para procesar y analizar archivos CSV, evaluando factores críticos como el tiempo de ejecución y la eficiencia. Este estudio de caso proporciona una visión completa de las diferencias y similitudes entre las herramientas de procesamiento de datos en Linux y Python, ofreciendo perspectivas valiosas para futuros proyectos de análisis de datos.

Keywords: Análisis de datos, herramientas de línea de comandos, escalbilidad, python.

1. Introducción

En el dinámico mundo tecnológico actual, el análisis de datos se ha convertido en un motor impulsor detrás de las decisiones empresariales, la innovación y la eficiencia operativa. Este fenómeno ha dado lugar al surgimiento del término Big Data, que engloba conjuntos de datos de gran volumen y complejidad que requieren enfoques y herramientas informáticas especializadas para su manejo efectivo. Estos conjuntos no solo son extensos, sino también diversos, incluyendo datos no estructurados, como los generados por las redes sociales, y datos semiestructurados.

Analizar grandes volúmenes de datos en tiempo real exige herramientas y técnicas que puedan ofrecer resultados rápidos y precisos. Esta necesidad de eficiencia nos lleva a una variedad de tecnologías, desde las clásicas herramientas de línea de comandos de Linux, como awk, grep, sed y sort, hasta el potencial de Python en Google Colab.

Este estudio, realizado en el marco de la asignatura Redes de Datos de la carrera Licenciatura en Sistemas – FaCENA UNNE, tiene como objetivo abordar estas cuestiones críticas. Nos proponemos comparar y optimizar las herramientas de línea de

comandos de Linux y Python en Google Colab para el procesamiento eficiente de datos, específicamente archivos CSV capturados con Wireshark. Evaluaremos no solo la velocidad y la eficiencia, sino también la adaptabilidad y escalabilidad de estas herramientas en el contexto del análisis de datos contemporáneo.

En los próximos capítulos, vamos a describir las soluciones propuestas y las metodologías empleadas para proporcionar una visión integral sobre cómo estas tecnologías pueden ser aprovechadas para abordar el desafío del Big Data, enfocándonos en la eficiencia como un componente fundamental para el éxito en el análisis de datos en el mundo actual. [1] [2] [3] [4]

2. Instrumentación

Para la implementación de este estudio se emplearon la siguiente serie de herramientas y entornos específicos:

- Sistema Operativo Linux: Se utilizó la distribución Ubuntu 22.04.2, ejecutada en una máquina virtual Oracle versión 6.1.22. La configuración de la máquina virtual incluyó asignación de recursos, otorgando 8 GB de RAM y un disco dinámico de 40 GB configurado como SSD.
- Wireshark 4.0.4: Esta versión específica de Wireshark se utilizó para la captura de datos. La recopilación de información se realizó en una red Wi-Fi definida, generando el conjunto de datos crucial para la investigación.Wireshark en su versión 4.0.4
- Google Colab: Se empleó esta plataforma de Google Research para realizar análisis y procesamiento de datos en Python.Colab, un producto de Google Research
- PuTTY 0.78: Esta herramienta se utilizó para el acceso remoto desde el sistema operativo Windows 10 a la máquina virtual Ubuntu.

Conjunto de Datos (CSV):

- Nombre y Características: El conjunto de datos clave en esta investigación es "registros_ws_500k." Este archivo CSV contiene 500,000 registros capturados por la herramienta Wireshark. Está compuesto por las siguientes columnas: 'No.', 'Time', 'Source', 'Destination', 'Protocol', 'Length', e 'Info'. La estructura y el tamaño preciso de este conjunto de datos son esenciales para realizar análisis detallados de la eficiencia y el rendimiento de las herramientas de procesamiento de datos tanto en el entorno Linux como en Python a través de Google Colab. [5]

Configuración de Hardware: La máquina que alojó la máquina virtual Ubuntu y ejecutó Google Colab cuenta con un sistema operativo Windows 10 y 8 GB de RAM.

3. Desarrollo

En la preparación del conjunto de datos en el entorno de Linux, se llevaron a cabo los siguientes procedimientos:

- Paso 1: Acceso al sistema de Windows desde Linux

Para facilitar la preparación del conjunto de datos en Linux, se accedió al sistema de Windows desde la máquina virtual de Linux. Este acceso se logró mediante el uso del símbolo del sistema de Windows, lo que permitió navegar y recuperar el archivo de datos necesario.

En el entorno de Google Colab, la preparación del conjunto de datos se llevó a cabo siguiendo estos pasos:

- Montaje de Google Drive
- Para acceder a los archivos almacenados en Google Drive desde Google Colab, se procedió a montar Google Drive en el entorno de trabajo.

Código:

```
from google.colab import drive
drive.mount('/content/drive')
```

● Paso 2: Carga del Conjunto de Datos

Una vez que Google Drive estaba montado, se procedió a cargar el conjunto de datos en Google Colab. El archivo de datos "registros_ws_500k.csv" se localizó en una ruta específica en Google Drive, se utilizó la biblioteca Pandas en Python, que es una herramienta ampliamente utilizada en el análisis de datos.

Código:

```
import pandas as pd
# Define la ruta a la carpeta donde se encuentran los archivos CSV
carpeta_csv = '/content/drive/MyDrive/Trabajo/csv/'
# Lee el archivo CSV
registros_ws_500k = pd.read_csv(carpeta_csv + 'registros_ws_500k.csv')
```

Estos procedimientos de preparación del conjunto de datos desempeñan un papel fundamental en esta investigación, ya que establecen la disponibilidad y accesibilidad de los datos en ambos entornos (Linux y Google Colab). [6] [7]

Consultas y Códigos de Consulta:

● Consulta N°1: protocolos utilizados y frecuencia.

Código en Linux (awk, sort, uniq):

```
time awk -F ',' '{print $5}' registros_ws_500k.csv | sort | uniq -c | sort -nr
```

Código en Python en Google Colab (usando Pandas):

```
import time
start_time = time.time()
contador_protocolos = registros_ws_500k['Protocol'].value_counts()
end_time = time.time()
tiempo_ejecución = end_time - start_time
print(f"Tiempo de ejecución: {tiempo_ejecución} segundos")
print(contador_protocolos)
```

● Consulta N°2: Obtener las direcciones ip de destino que fueron utilizados y frecuencia. Medir el tiempo de respuesta de la consulta.

Código en Linux (awk, sort, uniq):

```
time awk -F ',' '{print $4}' registros_ws_500k.csv | sort | uniq -c | sort -nr
```

Código en Python en Google Colab (usando Pandas):

```
import time
start_time = time.time()
top_ips_destino = registros_ws_500k['Destination'].value_counts().head(10)
```

```
end_time = time.time()
tiempo_ejecucion = end_time - start_time
print(f"Tiempo de ejecución: {tiempo_ejecucion} segundos")
print(top_ips_destino)
```

● Consulta N°3: Obtener las direcciones ip de origen que fueron utilizados y frecuencia. Medir el tiempo de respuesta de la consulta.
Código en Linux (awk, sort, uniq):

```
time awk -F ',' '{print $3}' registros_ws_500k.csv | sort | uniq -c | sort -nr
```

Código en Python en Google Colab (usando Pandas):

```
import time
start_time = time.time()
top_ips_destino = registros_ws_500k['Source'].value_counts().head(10)
end_time = time.time()
tiempo_ejecucion = end_time - start_time
print(f"Tiempo de ejecución: {tiempo_ejecucion} segundos")
print(top_ips_destino)
```

● Consulta N°4: Cálculo de tamaño promedio de paquetes en bytes. Medir el tiempo de respuesta de la consulta.
Código en Linux (awk, sort, uniq):

```
time cat registros_ws_500k.csv | awk -F "," '{sum+=$6} END {print sum/NR}'
```

Código en Python en Google Colab (usando Pandas):

```
import time
start_time = time.time()
tamaño_promedio = registros_ws_500k['Length'].mean()
end_time = time.time()
tiempo_ejecucion = end_time - start_time
print(f"Tiempo de ejecución: {tiempo_ejecucion} segundos")
print("Tamaño de paquete promedio:", tamaño_promedio, "bytes")
```

Además, para ampliar este análisis a un espectro más amplio, se realizará el mismo procedimiento con dos archivos adicionales: "registros_ws_250k.csv" y "registros_ws_100k.csv". Estos conjuntos de datos contienen muestras de tamaño reducido que nos permitirá contrastar y comparar los resultados obtenidos anteriormente.
Con el objetivo de ampliar el espectro de nuestro análisis, se replicará el procedimiento utilizando dos archivos adicionales: "registros_ws_250k.csv" y "registros_ws_100k.csv" como se mencionó previamente.

4. Resultados y Comparativa

Para llevar a cabo un análisis comparativo detallado entre las herramientas de línea de comandos en Linux y Python en Google Colab en el contexto del manejo de Big Data, se tomaron medidas específicas para evaluar la eficiencia, escalabilidad y adaptabilidad de ambas plataformas. Para medir la escalabilidad y evaluar ciertos criterios, se optó por dividir el conjunto de datos original en tres archivos de tamaños variados: 500,000

registros, 250,000 registros y 100,000 registros. Estos conjuntos de datos proporcionaron un escenario diversificado para evaluar el rendimiento de las herramientas en diferentes volúmenes de datos.

Eficiencia en el Análisis de Datos.
En el análisis realizado, los resultados destacan consistentes el rendimiento superior de Python en comparación con la herramienta de línea de comandos en Linux en todas las consultas específicas examinadas. Estos hallazgos resaltan la eficacia innegable de Python para realizar operaciones específicas de análisis de datos en comparación con el entorno Linux utilizado, que hace uso de herramientas de línea de comandos.
Python, como lenguaje de programación de propósito general, se destaca por su versatilidad y eficiencia, respaldado por una amplia gama de bibliotecas y funciones incorporadas especialmente optimizadas para el análisis de datos. En contraste, la herramienta de línea de comandos en Linux son poderosas, pero están diseñadas principalmente para operaciones específicas del sistema operativo y carecen de la optimización necesaria para el análisis de datos avanzado.

La naturaleza de las operaciones específicas de análisis de datos juega un papel crucial en esta disparidad de rendimiento. Python, con su capacidad para realizar operaciones para realizar operaciones en memoria, se destaca, mientras que algunas operaciones específicas pueden estar mejor optimizadas en Python gracias a las bibliotecas especializadas que ofrece. Este contraste subraya la importancia de seleccionar la herramienta adecuada según las necesidades y la naturaleza de las operaciones en el contexto del análisis de datos.

Escalabilidad y Adaptabilidad:
El análisis deja en evidencia la relación directa entre el tamaño del conjunto de datos y el rendimiento de las herramientas empleadas es claramente observable que, a medida que aumenta el volumen del conjunto de datos, se evidencia un incremento significativo en el tiempo de ejecución de las consultas tanto en el entorno de Linux como en Python en Google Colab. Este hallazgo remarca la importancia de la eficiencia en el análisis de datos.

Otro aspecto importante a considerar es la escalabilidad y adaptabilidad de las herramientas utilizadas en el análisis. Python destaca en estos aspectos, debido a que ofrece una amplia variedad de bibliotecas especializadas lo que permite realizar prácticamente cualquier análisis de datos. En contraste, la herramienta de línea de comandos en Linux carece de estas mismas características, pese a que cuenta con una poderosa capacidad para ciertas operaciones específicas. Su limitación para manejar la diversidad de datos y satisfacer la variedad necesidades de análisis queda en evidencia al compararla con Python y sus bibliotecas especializadas.

Consultas	**Tiempo Ejecución**					
	Linux (5)			**Python (5)**		
	100 mil	250 mil	500 mil	100 mil	250 mil	500 mil

(1)	0,176s	0,397s	0,763s	0.007s	0.016s	0.035s
(2)	0,171s	0,438s	0,838s	0,007s	0,017s	0,037s
(3)	0,190s	0,471s	0,917s	0,006s	0,025s	0,035s
(4)	0,143s	0,315s	0,608s	0,0005s	0,0012s	0,0024s

(1) Protocolos que experimentan repetición significativa
(2) IP destino más frecuentes
(3) IP origen más frecuentes
(4) Cálculo de tamaño promedio de paquetes en bytes
(5) Cantidad de registros

5. Conclusiones

En este análisis comparativo entre herramientas de línea de comandos en Linux y Python para el manejo de Big Data, se ha revelado una disparidad significativa en términos de eficiencia y adaptabilidad. Las herramientas de línea de comandos en Linux, como awk, sort, grep y sed, aunque son útiles para tareas específicas y operaciones simples, muestran limitaciones evidentes cuando se enfrentan a operaciones específicas de análisis de datos a gran escala, especialmente a medida que aumenta el volumen y la complejidad de los datos.

Los resultados obtenidos subrayan la importancia de elegir la herramienta adecuada en un entorno tecnológico que exige eficiencia en la gestión y análisis de datos. La eficiencia de las herramientas de análisis de datos disminuye a medida que aumenta el volumen de datos, lo que destaca la relevancia de comprender estas herramientas para tareas de análisis de datos.

6. Referencias

[1] Stratebi, Libro Verde del Big Data, Madrid, España, 2014.
[2] M. Pérez Marqués, Big Data. Técnicas, herramientas y aplicaciones., RC Libros., 2015.
[3] V. &. C. K. Mayer-Schönberger, Big data. La revolución de los datos masivos, Turner, 2013.
[4] María Dolores Pérez, Linux avanzado, Editorial ICB, ISBN 9788492889464.2015
[5] M. Probert, «grep, awk and sed – three VERY useful command-line utilities,» Uni of York, York, Reino Unido, [En línea] https://www-users.york.ac.uk/~mijp1/teaching/2nd_year_Comp_Lab/guides/grep_awk_sed.pdf, 2016.
[6] O. Campesato, Working with grep, sed, and awk. Pocket Primer, Dulles, VA 20166: Mercury Learning and Information, 2023.
[7] P. w. d. G. Colab. [En línea]. https://research.google.com/colaboratory/intl/es/faq.html