UNIVERSIDAD DE ANTIOQUIA

HILOS Y PROCESOS

PARA ACELERAR PROCESOS DE BÚSQUEDA

Daniel Lujan Agudelo
Luis Guillermo Sánchez Cubides
Jose David Gómez Muñeton
Juan Pablo Arango Gaviria

UdeA

HILOS

VS

PRO-CESOS

UdeA

Introducción

El manejo de grandes volúmenes de datos es crucial para sectores como ciencia de datos, comercio electrónico e inteligencia artificial. Las búsquedas secuenciales se vuelven ineficaces a medida que los datos crecen, pero el paralelismo, usando hilos y procesos, permite mejorar drásticamente el rendimiento y la escalabilidad, ofreciendo soluciones rápidas y eficientes.

UdeA

Hilos

- Espacio de memoria compartido entre hilos, lo que facilita la comunicación.
- El cambio de contexto es menos costoso.
- Menos costoso en terminos de memoria y CPU.
- I/O bound
- Es necesario manejar problemas de sincronización como condiciones de carrera.
- Los recursos compartidos pueden llevar a bloqueos entre hilos.

Procesos

- Espacio de memoria separado para cada proceso.
- Como tienen memoria independiente se eliminan los problemas de condiciones de carrera
- CPU bound
- Mayor uso de la CPU.
- Como cada proceso tiene su propio espacio de memoria incrementa el consumo de la misma.

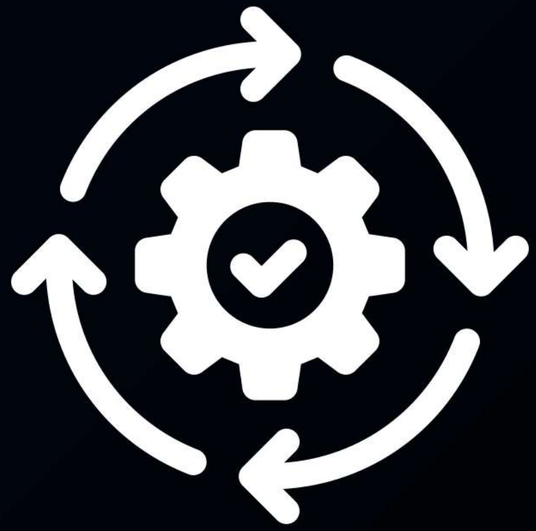
UdeA

Metodología

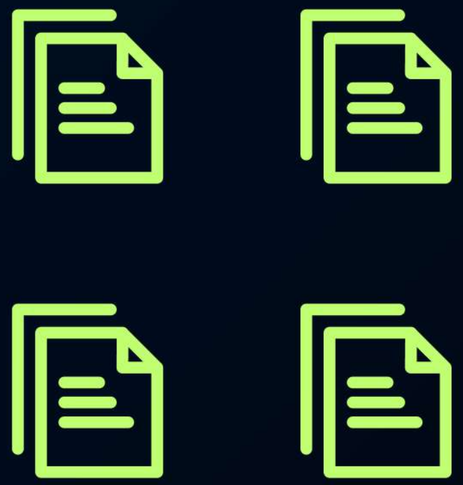
Implementación de las técnicas de búsqueda

Configuración del entorno experimental

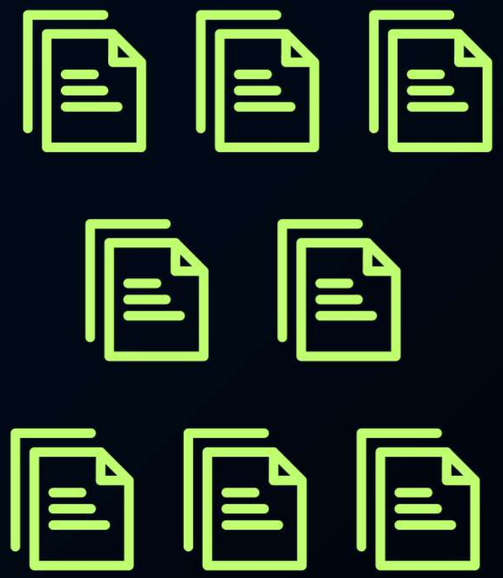
Ejecución de pruebas en datasets de diferentes tamaños



El Enfrentamiento...



4 Archivos
497,664 Registros C/U



8 Archivos
248,832 Registros C/U

12 Archivos
165,888 Registros C/U

UdeA

Multithreading

Se implementa **multithreading** para realizar la busqueda de un valor en múltiples archivos. Para cada archivo se crea un hilo independiente y este recorre linea por linea para comparar el contenido con el valor buscado.

Si un hilo encuentra el valor, todos los demas detienen su ejecución, evitando trabajo innecesario y mejorando el tiempo total de la busqueda.

UdeA

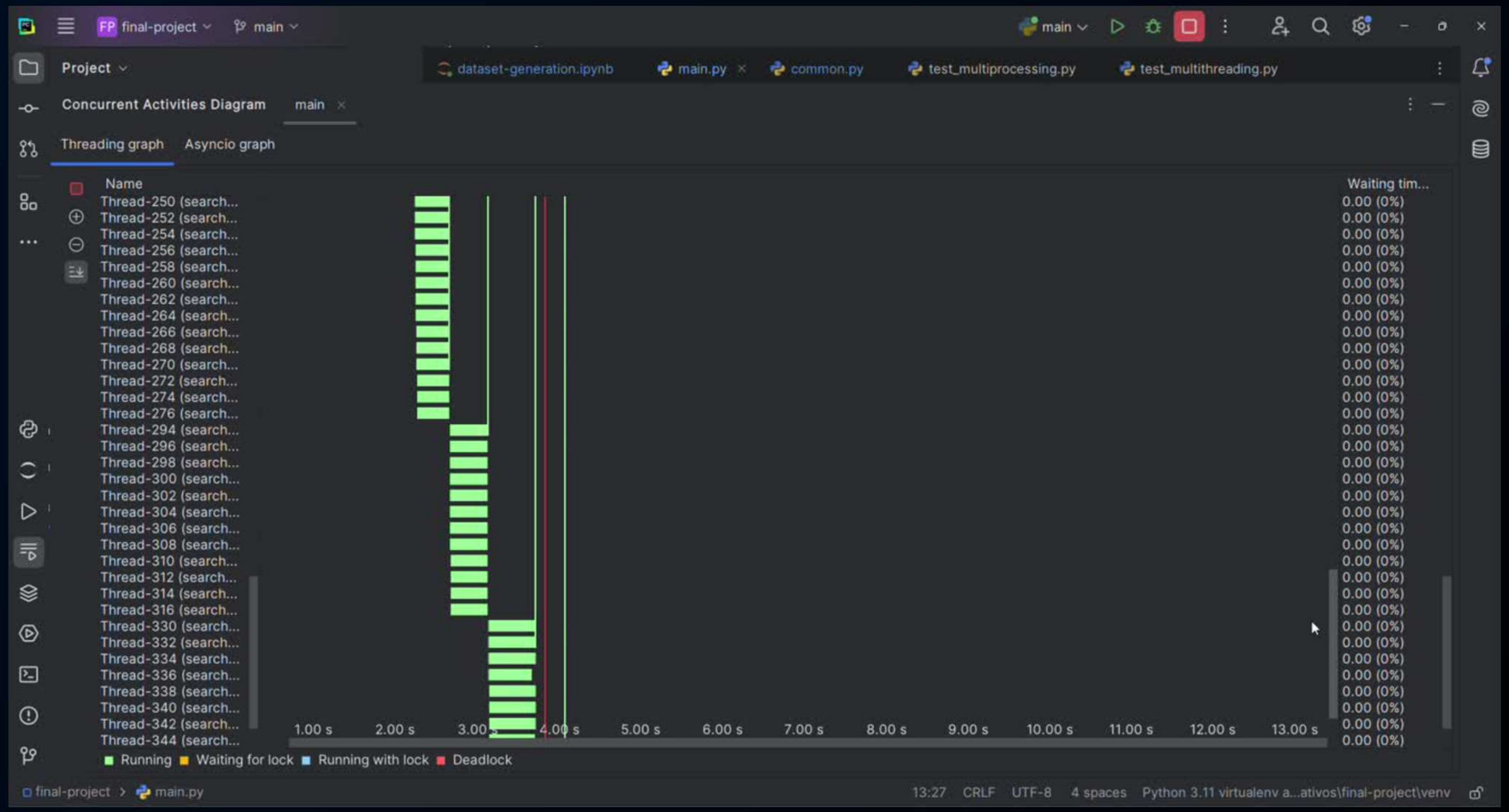
Multiprocessing

En este metodo en lugar de usar hilos, se crean proceos independientes, donde cada proceso es el encargado de buscar en un archivo en especifico. Cada proceso se ejecuta de forma aislada, trabajando en su propio espacio de memoria y evitando cualquier interferencia entre los mismos.

La implementación utiliza un pool de procesos y a través del método map, el pool asigna a cada proceso un archivo y un valor a buscar, ejecutando la búsqueda en paralelo y retornando una lista de resultados.

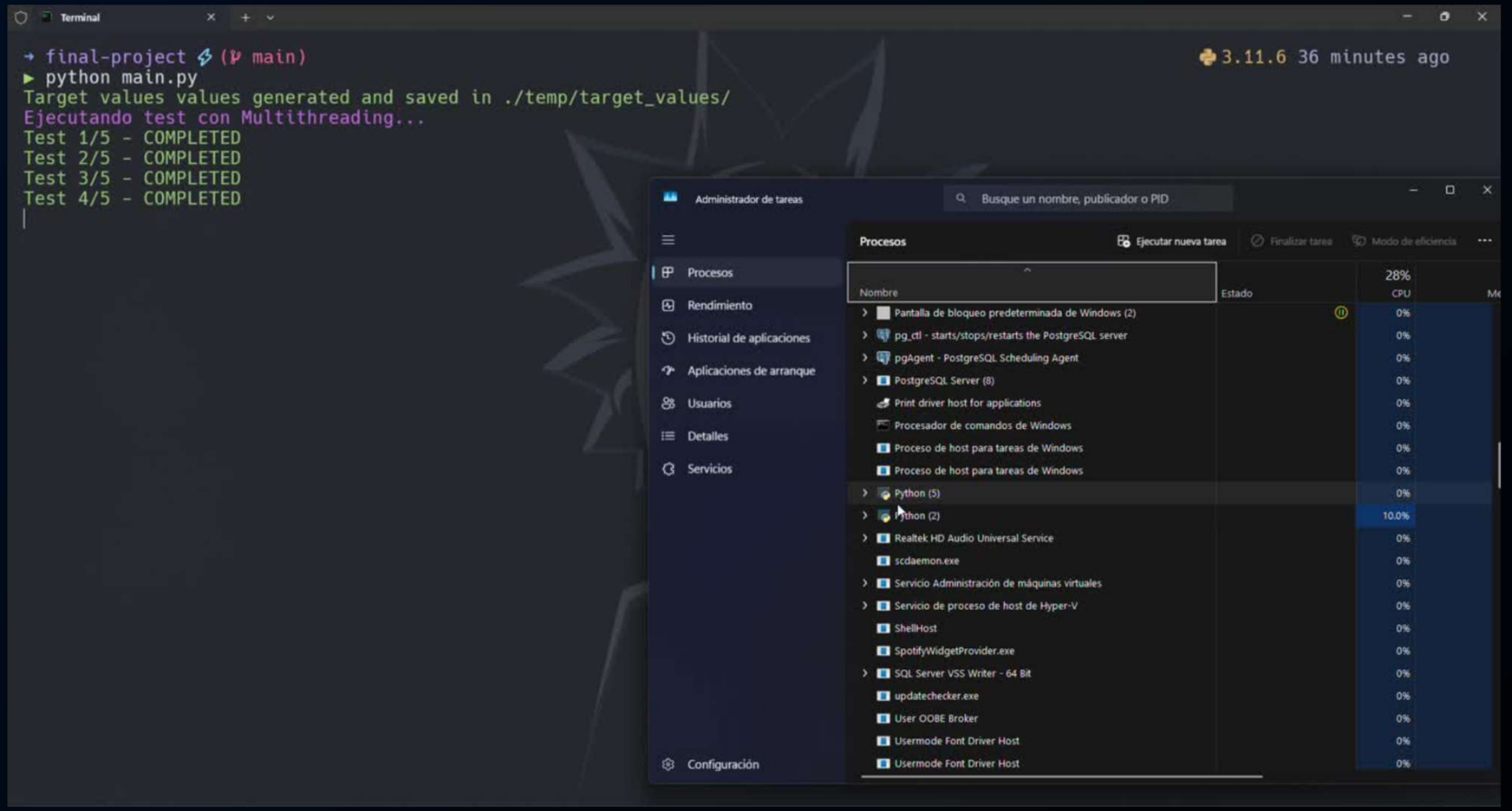
UdeA

Demo


final-project
main
Project
dataset-generation.ipynb
main.py
common.py
test_multiprocessing.py
test_multithreading.py
Concurrent Activities Diagram
main
Threading graph
Asyncio graph
Name
Waiting tim...
Thread-250 (search...
Thread-252 (search...
Thread-254 (search...
Thread-256 (search...
Thread-258 (search...
Thread-260 (search...
Thread-262 (search...
Thread-264 (search...
Thread-266 (search...
Thread-268 (search...
Thread-270 (search...
Thread-272 (search...
Thread-274 (search...
Thread-276 (search...
Thread-294 (search...
Thread-296 (search...
Thread-298 (search...
Thread-300 (search...
Thread-302 (search...
Thread-304 (search...
Thread-306 (search...
Thread-308 (search...
Thread-310 (search...
Thread-312 (search...
Thread-314 (search...
Thread-316 (search...
Thread-330 (search...
Thread-332 (search...
Thread-334 (search...
Thread-336 (search...
Thread-338 (search...
Thread-340 (search...
Thread-342 (search...
Thread-344 (search...
0.00 (0%)
1.00 s
2.00 s
3.00 s
4.00 s
5.00 s
6.00 s
7.00 s
8.00 s
9.00 s
10.00 s
11.00 s
12.00 s
13.00 s
Running
Waiting for lock
Running with lock
Deadlock
final-project
main.py
13:27
CRLF
UTF-8
4 spaces
Python 3.11 virtualenv a...ativos\final-project\venv

UdeA

Demo


Terminal
final-project (main)
3.11.6 36 minutes ago
python main.py
Target values values generated and saved in ./temp/target_values/
Ejecutando test con Multithreading...
Test 1/5 - COMPLETED
Test 2/5 - COMPLETED
Test 3/5 - COMPLETED
Test 4/5 - COMPLETED
Administrador de tareas
Busque un nombre, publicador o PID
Procesos
Ejecutar nueva tarea
Finalizar tarea
Modo de eficiencia
Procesos
Rendimiento
Historial de aplicaciones
Aplicaciones de arranque
Usuarios
Detalles
Servicios
Configuración
Nombre
Estado
28%
CPU
Pantalla de bloqueo predeterminada de Windows (2)
pg_ctl - starts/stops/restarts the PostgreSQL server
pgAgent - PostgreSQL Scheduling Agent
PostgreSQL Server (8)
Print driver host for applications
Procesador de comandos de Windows
Proceso de host para tareas de Windows
Proceso de host para tareas de Windows
Python (5)
Python (2)
10.0%
Realtek HD Audio Universal Service
scdaemon.exe
Servicio Administración de máquinas virtuales
Servicio de proceso de host de Hyper-V
ShellHost
SpotifyWidgetProvider.exe
SQL Server VSS Writer - 64 Bit
updatechecker.exe
User OOBE Broker
Usermode Font Driver Host
Usermode Font Driver Host

UdeA

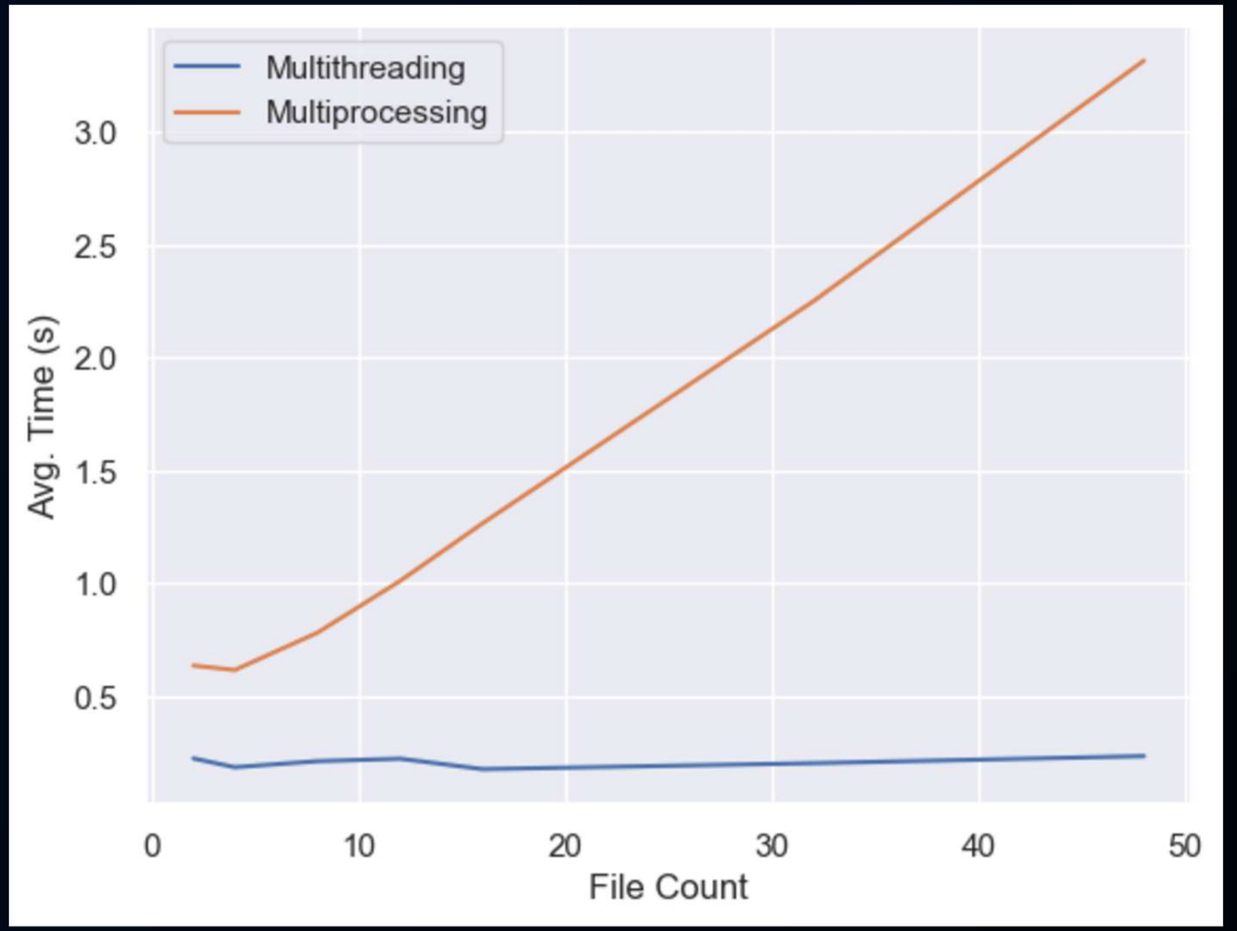
Uso de recursos

```
Starting Multithreading
Ejecutando test con Multithreading...
Test 1/3 - COMPLETED
Test 2/3 - COMPLETED
Test 3/3 - COMPLETED
Test finalizado. Resultados guardados en 'results-multithreading.csv'
Multithreading completed!
Peak CPU usage: 62.50%
Peak Memory usage: 66.10 MB
Elapsed time: 4.51 seconds
```

```
Starting Multiprocessing
Ejecutando test con Multiprocessing...
Test 1/3 - COMPLETED
Test 2/3 - COMPLETED
Test 3/3 - COMPLETED
Test finalizado. Resultados guardados en 'results-multiprocessing.csv'
Multiprocessing completed!
Peak CPU usage: 71.90%
Peak Memory usage: 65.00 MB
Elapsed time: 30.54 seconds
```

UdeA

Anális Estadístico



UdeA

Anális Estadístico

H0: No hay diferencias significativas en los tiempos de ejecución entre las técnicas de búsqueda.

H1: Existen diferencias significativas en los tiempos de ejecución entre las técnicas.

UdeA

Anális Estadístico

	SCE	DF	F	PR(>F)
Técnica de paralelismo	202.358518	1	447.732571	2.096616e-73
Residual	252.195307	558	----	----

Anális Estadístico

Se rechaza la hipótesis nula:

H1: Existen diferencias significativas en los tiempos de ejecución entre las técnicas.

... Y el ganador es 🥁

UdeA

Conclusiones

- El rendimiento de la técnica de procesamiento paralelo mediante multithreading se ve altamente afectado con el número de archivos, por la cantidad de cambios de contexto.

- Es mejor utilizar procesamiento paralelo mediante multiprocesing cuando las tareas que se requieren son de alto consumo de CPU.

- El multiprocessing es menos adecuado para este tipo de tareas debido a la sobrecarga asociada con la creación y manejo de procesos en comparación con los hilos.

UdeA

Referencias

1 **What is multiprocessing?**
https://www.techtarget.com/searchdatacenter/definition/multiprocessing

2 **Entendiendo los procesos, hilos y multihilos.**
https://medium.com/@diego.coder/entendiendo-los-procesos-hilos-y-multihilos-9423f6e40ca7

3 **Python time.time() vs time.perf_counter().**
https://superfastpython.com/time-time-vs-time-perf_counter/

Gracias

Repositorio
https://github.com/daniel-lujan/multiprocessing-multithreading

Notebook
https://github.com/daniel-lujan/multiprocessing-multithreading/blob/main/notebooks/EDA.ipynb