

PROYECTO 3

Paralelismo en un ambiente controlado

DIEGO MÚNERA
ALEJANDRO TORRES

IMPORTANTE

Para cada archivo CSV, se implementó una lectura en fragmentos, lo cual permite reducir el uso de memoria y manejar archivos grandes sin problemas. Además, se incluyó un proceso de detección de codificación de los archivos, ya que los datos pueden provenir de distintas fuentes y tener diferentes formatos. Sin embargo, esta detección introduce un pequeño retraso adicional que afecta el tiempo total de procesamiento.



LECTURA SECUENCIAL

- ★ python dataload.py -f files
- ★ Hora de inicio del programa: 6am

Siguiente



load_sequential.py



Cada archivo se lee de manera individual, registrando el tiempo de inicio y fin de carga, así como el uso de memoria. Utiliza la librería rich para mostrar en tiempo real el uso de CPU, lo cual permite un monitoreo continuo y detallado del rendimiento del sistema durante la ejecución.

Núcleo	Uso (%)
Núcleo 0	52.10%
Núcleo 1	46.60%
Núcleo 2	37.50%
Núcleo 3	19.40%
Núcleo 4	20.50%
Núcleo 5	12.50%
Núcleo 6	23.00%
Núcleo 7	4.10%
Núcleo 8	1.40%
Núcleo 9	0.00%

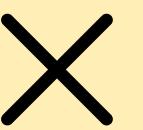
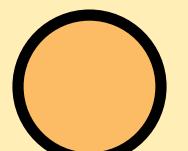
Macbook Pro M2 Pro 32GB RAM

🔍

X

Nombre	T. Inicial	T. Final	Duración (ms)	Memoria (MB)
files/MXvideos.csv	16:11:24.952	16:11:25.590	637.748	56.34
files/INvideos.csv	16:11:25.590	16:11:26.322	732.279	80.95
files/DEvideos.csv	16:11:26.322	16:11:27.063	740.882	104.61
files/JPvideos.csv	16:11:27.063	16:11:27.500	436.793	114.7
files/KRvideos.csv	16:11:27.500	16:11:28.007	507.06	130.66
files/CAvideos.csv	16:11:28.007	16:11:28.745	737.867	148.78
files/RUvideos.csv	16:11:28.745	16:11:29.256	510.86	170.52
files/FRvideos.csv	16:11:29.256	16:11:29.928	672.141	189
files/USvideos.csv	16:11:29.929	16:11:30.658	729.674	207.47
files/GBvideos.csv	16:11:30.658	16:11:31.295	636.731	225.38

Tiempo total: 6.35 s



LECTURA EN UN SOLO NÚCLEO

- ★ `python dataload.py -f files -s`
- ★ Se asigna arbitrariamente un solo núcleo

Siguiente



load_single_core.py



Cada archivo se asigna a un hilo individual dentro de un ThreadPoolExecutor, permitiendo múltiples cargas simultáneas en un único núcleo sin distribuir el trabajo entre varios. Durante el proceso, se registra el tiempo de carga y el uso de memoria para cada archivo, y se utiliza la librería rich para mostrar en tiempo real el uso de CPU, proporcionando un monitoreo continuo del rendimiento.

Núcleo	Uso (%)
Núcleo 0	0.00%
Núcleo 1	0.00%
Núcleo 2	0.00%
Núcleo 3	0.00%
Núcleo 4	0.00%
Núcleo 5	0.00%
Núcleo 6	50.00%
Núcleo 7	0.00%
Núcleo 8	0.00%
Núcleo 9	0.00%

Macbook Pro M2 Pro 32GB RAM

Search icon

X icon

Nombre	T. Inicial	T. Final	Duración (ms)	Memoria (MB)
files/GBvideos.csv	16:16:38.571	16:16:42.916	4344.88	175.48
files/CAvideos.csv	16:16:38.339	16:16:43.062	4722.6	181.5
files/FRvideos.csv	16:16:38.457	16:16:43.276	4819.25	188.98
files/INvideos.csv	16:16:38.328	16:16:43.689	5361.24	203.86
files/USvideos.csv	16:16:38.512	16:16:43.765	5253.03	205.58
files/DEvideos.csv	16:16:38.328	16:16:44.041	5713.6	216.7
files/MXvideos.csv	16:16:38.327	16:16:44.058	5730.84	217.28
files/KRvideos.csv	16:16:38.334	16:16:44.227	5893.66	225.42
files/JPvideos.csv	16:16:38.328	16:16:44.565	6237.32	239.59
files/RUvideos.csv	16:16:38.422	16:16:44.613	6191.09	241.53

Tiempo total: 6 s



LECTURA EN VARIOS NÚCLEOS

- ★ `python dataload.py -f files -m`
- ★ Se utilizan los núcleos disponibles.

Siguiente



load_multi_core.py



Cada archivo se asigna a un **proceso independiente** mediante `ProcessPoolExecutor`, distribuyendo la carga entre **varios núcleos** del procesador. Dentro de cada proceso, se utiliza `ThreadPoolExecutor` para dividir el archivo en **fragmentos** (chunks) y procesarlos en **paralelo**, mejorando la eficiencia. Durante la ejecución, se registra el tiempo de carga y el uso de memoria para cada archivo, y se utiliza `rich` para mostrar en tiempo real el uso de CPU, permitiendo un monitoreo detallado del rendimiento en un entorno de procesamiento intensivo.

Núcleo	Uso (%)
Núcleo 0	91.00%
Núcleo 1	89.20%
Núcleo 2	94.90%
Núcleo 3	94.90%
Núcleo 4	91.70%
Núcleo 5	90.40%
Núcleo 6	92.40%
Núcleo 7	91.80%
Núcleo 8	90.40%
Núcleo 9	89.20%

Macbook Pro M2 Pro 32GB RAM

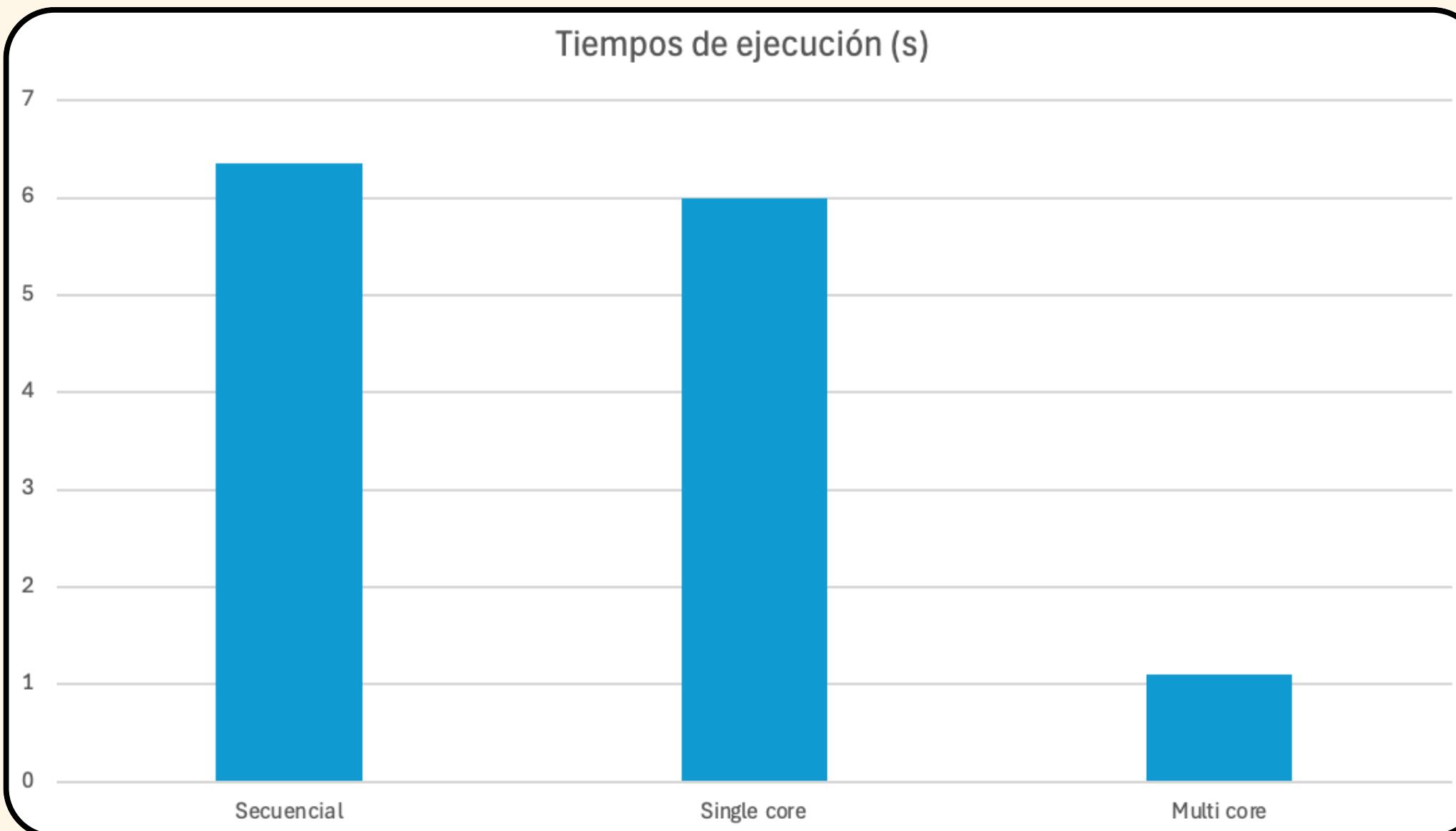
🔍 X

Nombre	T. Inicial	T. Final	Duración (ms)	Memoria (MB)
files/MXvideos.csv	16:18:05.782	16:18:06.814	1032.39	149.41
files/INvideos.csv	16:18:05.785	16:18:06.931	1145.66	162.27
files/DEvideos.csv	16:18:05.792	16:18:06.995	1202.93	180.25
files/JPvideos.csv	16:18:05.792	16:18:06.497	704.496	87.19
files/KRvideos.csv	16:18:05.794	16:18:06.695	900.879	125.45
files/CAvideos.csv	16:18:05.794	16:18:07.020	1226.13	177.47
files/RUvideos.csv	16:18:05.799	16:18:06.767	968.275	161.91
files/FRvideos.csv	16:18:05.806	16:18:06.934	1127.54	164.3
files/USvideos.csv	16:18:05.810	16:18:07.010	1199.59	173.95
files/GBvideos.csv	16:18:05.814	16:18:06.866	1051.21	153.27

Tiempo total: 1.1 s



COMPARATIVA



CONCLUSIONES Y APRENDIZAJES



- El uso de paralelismo y concurrencia optimiza el procesamiento de archivos grandes, reduciendo el tiempo total de ejecución.
- La ejecución en múltiples núcleos (multi-core) ofrece el mejor rendimiento comparado con el procesamiento secuencial y single-core.
- La lectura en bloques (chunks) permite manejar grandes volúmenes de datos sin agotar la memoria, ya que procesa partes del archivo por separado.
- La detección de codificación introduce un ligero retraso, pero es crucial para asegurar la correcta interpretación de los datos.
- Cada enfoque tiene ventajas específicas según los recursos y el contexto del procesamiento; la elección depende de los requisitos del sistema y la capacidad de hardware disponible.

GRACIAS