Programación Paralela con GNU Parallel: Informe de Resultados

Práctica: CSV y números aleatorios

3 de octubre de 2025

1. Objetivo

Comparar la ejecución en serie frente a la ejecución en paralelo (con GNU Parallel) en dos actividades:

- Procesamiento de archivos CSV (cálculo de media y mediana por columna).
- Generación de archivos con números aleatorios y cómputo de suma por archivo.

2. Entorno y reproducibilidad

Se utilizó un contenedor Docker que extiende la imagen `alhumaidyaroob/gnu-parallel` para incluir Python 3.[1]

Construcción de la imagen

```
docker build -t local/gnu-parallel-python:latest .
```

Ejecución dentro del contenedor

Todos los comandos se ejecutaron montando el directorio de trabajo en `/work`:

```
docker run --rm -it -v ${PWD}:/work -w /work \
  local/gnu-parallel-python:latest bash -lc "<comandos>"
```

3. Actividad 1: CSV en serie vs. paralelo

3.1. Generación de datos

Se generaron tres archivos con 200 000 filas cada uno:

```
python .\data_gen.py --rows 200000
# output: data1.csv, data2.csv, data3.csv (columnas: value1,value2,value3)
```

3.2. Script de procesamiento

El script lee un CSV y entrega: archivo, columna, conteo, media, mediana y tiempo (s).

```
python .\script.py <archivo> --column value2
```

[1] https://hub.docker.com/r/alhumaidyaroob/gnu-parallel

3.3. Ejecución en serie

Se ejecutó uno por vez dentro del contenedor:

```
python3 script.py data1.csv --column value2
python3 script.py data2.csv --column value2
python3 script.py data3.csv --column value2
```

Resultados (tiempo individual en segundos):

- `data1.csv`: 0,624 s
- `data2.csv`: 0,623 s
- `data3.csv`: 0,628 s

Total en serie: $0{,}624 + 0{,}623 + 0{,}628 \approx 1{,}875$ s.

3.4. Ejecución en paralelo con GNU Parallel

```
time parallel python3 script.py --column value2 ::: data1.csv data2.csv data3.csv
```

Tiempos por archivo reportados por el script: 0,659 s, 0,660 s, 0,668 s.
Tiempo total (*real*): 1,021 s.

3.5. Comparación y speed-up

Se define la aceleración como:

$$S = \frac{T_{\text{serie}}}{T_{\text{paralelo}}} \tag{1}$$

Con $T_{\text{serie}} \approx 1{,}875$ s y $T_{\text{paralelo}} \approx 1{,}021$ s, se obtiene

$$S \approx \frac{1{,}875}{1{,}021} \approx 1{,}84 \times . \tag{2}$$

Observaciones: el tiempo total paralelo se acerca al de la tarea más lenta más el *overhead*; al aumentar el trabajo (más filas), la ganancia suele mejorar.

Cuadro 1: Resumen de tiempos (Actividad CSV)

Modo	Métrica	Tiempo (s)
Serie	Suma de 3 tareas	1.875
Paralelo	*real* (wall clock)	1.021

4. Actividad 2: Archivos de números aleatorios

4.1. Caso A: 5 archivos, 100 números

Generación y suma por archivo (paralelo):

```
seq 5 | parallel "shuf -i 1-1000 -n 100 > data{}.txt"
parallel "awk '{s+=\$1} END {print \"{}: \" s}' {}" ::: data*.txt | sort -V | tee sums
    .txt
```

Ejemplo de salida:

```
data1.txt: 51043
data2.txt: 47010
data3.txt: 50168
...
```

4.2. Caso B: 20 archivos, 200 números

Se automatizó en `numbers_parallel_20.sh`:

1. Generar 20 archivos: `seq 20 | parallel "shuf -i 1-1000 -n 200 >data{}.txt"`
2. Sumas en serie con `time` y en paralelo con `time parallel`.

Tiempos registrados:

- Serie (*real*): 0,081 s
- Paralelo (*real*): 0,253 s

Conclusión específica: con poco trabajo por archivo (200 números), el *overhead* de lanzar tareas paralelas puede superar la ganancia, y la versión paralela resulta más lenta. Aumentar el tamaño (p.ej., `-n 10000`) suele revertir esto.

5. Conclusiones

- El paralelismo reduce el tiempo de pared cuando cada tarea tiene suficiente cómputo y es independiente.
- El *overhead* (procesos, orquestación, E/S, contenedores) puede anular la ventaja si la tarea es pequeña.
- En la actividad CSV, se obtuvo un speed-up aproximado de $\sim 1{,}84\times$. Para cargas mayores (más filas), la aceleración debería aumentar.
- Ajustar `-jobs` en GNU Parallel al número de núcleos/recursos disponibles ayuda a aprovechar mejor el hardware.
- Scripts y contenedor reproducibles facilitan repetir y comparar los experimentos.

Anexos

Silenciar aviso de citación de GNU Parallel

```
docker run --rm -it local/gnu-parallel-python:latest bash -lc "parallel --citation"
```