Paralelización en CUDA: Transformada de Fourier Discreta y Continua

Jesús Losada Arauzo

22 de mayo de 2025

Resumen

En este informe se analiza la paralelización mediante CUDA de la Transformada de Fourier Discreta (DFT) y versiones continuas con integración numérica. El objetivo es medir el rendimiento frente a la versión secuencial utilizando diferentes técnicas de integración: suma de rectángulos, trapecios y método de Simpson. Se emplea memoria unificada y eventos CUDA para una gestión sencilla de datos y tiempos.

Paralelización en CUDA

El programa implementa varios *kernels* CUDA, uno para cada versión de la transformada:

- **DFT**: versión discreta clásica.
- **CFT**: versión continua básica, usando suma directa.
- **CFT_Trapecio** y **CFT_Simpson**: integración continua por métodos numéricos.

Se utiliza una distribución fija de hilos: `BLOCK_SIZE = 256` y `NUM_BLOCKS = 10`. Cada hilo se encarga de calcular una componente del vector de Fourier (una frecuencia). Internamente, el hilo hace un bucle que suma o integra la función con el método correspondiente.

Distribución del trabajo

Cada hilo CUDA calcula un índice global:

```
i = blockIdx.x * blockDim.x + threadIdx.x
```

Con ese índice se decide si se procesa la componente i del vector de salida. De este modo, la carga de trabajo se distribuye automáticamente entre los hilos, permitiendo ejecutar en paralelo todos los valores de la transformada.

Medición de tiempo

El tiempo se mide mediante `cudaEventRecord`, lo cual permite medir con precisión únicamente la ejecución del kernel (sin contar asignaciones ni E/S). El resultado se almacena en archivos para su posterior graficado.

Resultados obtenidos

Las siguientes gráficas muestran los tiempos medidos en milisegundos para distintos tamaños de muestra. Se comparan versiones secuenciales y paralelas.

- Para muestras pequeñas, CUDA tiene cierta penalización inicial.
- A partir de cierto tamaño, la versión paralela supera ampliamente a la secuencial.

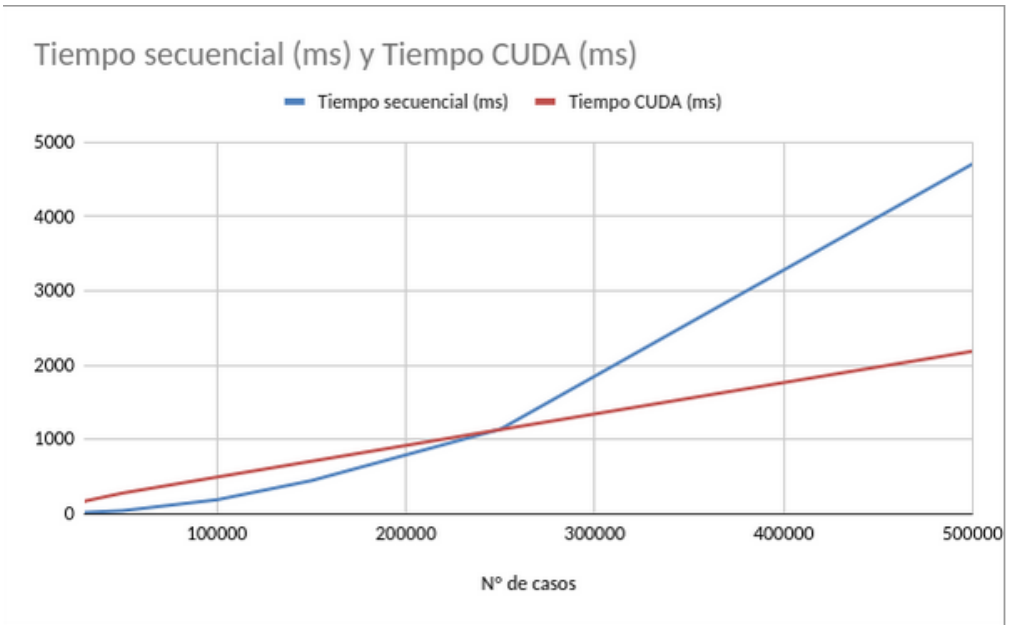

Figura 1: Tiempo de ejecución - DFT (Transformada Discreta)

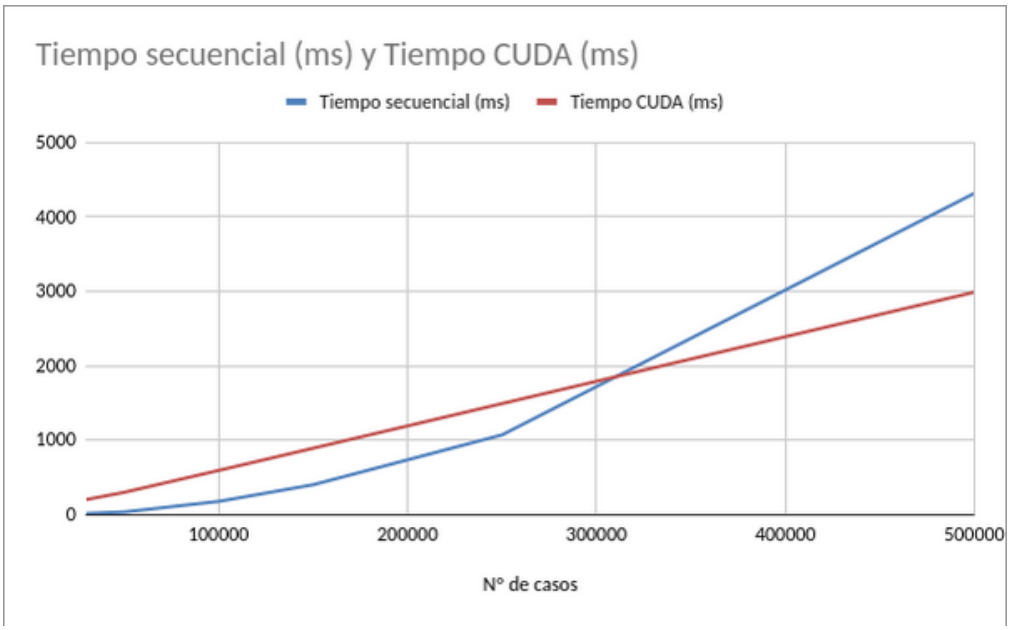

Figura 2: Tiempo de ejecución - Suma de Rectángulos

Código CUDA resumido

```
__global__ void DFT(cuDoubleComplex *Fourier, const double *muestras, int N){
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < N){
        cuDoubleComplex sum = make_cuDoubleComplex(0.0, 0.0);
        for (int j = 0; j < N; j++){
            double angle = -2.0 * PI * i * j / N;
            cuDoubleComplex term = make_cuDoubleComplex(muestras[j]*cos(angle),
                                                        muestras[j]*sin(angle))
            sum = cuCadd(sum, term);
        }
        Fourier[i] = sum;
    }
}
```

Conclusión

Gracias a la paralelización en CUDA, se consigue una mejora notable en el rendimiento del cálculo de transformadas de Fourier a gran escala. Aunque la inicialización de la GPU introduce cierta latencia, para volúmenes de datos grandes la ejecución en GPU resulta significativamente más rápida que la versión secuencial.

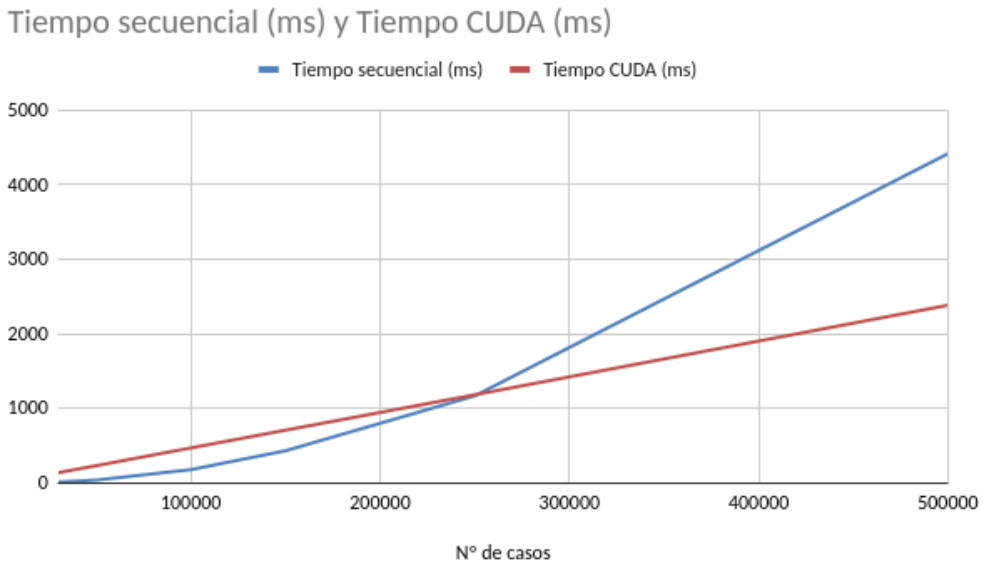

Figura 3: Tiempo de ejecución - Método del Trapecio

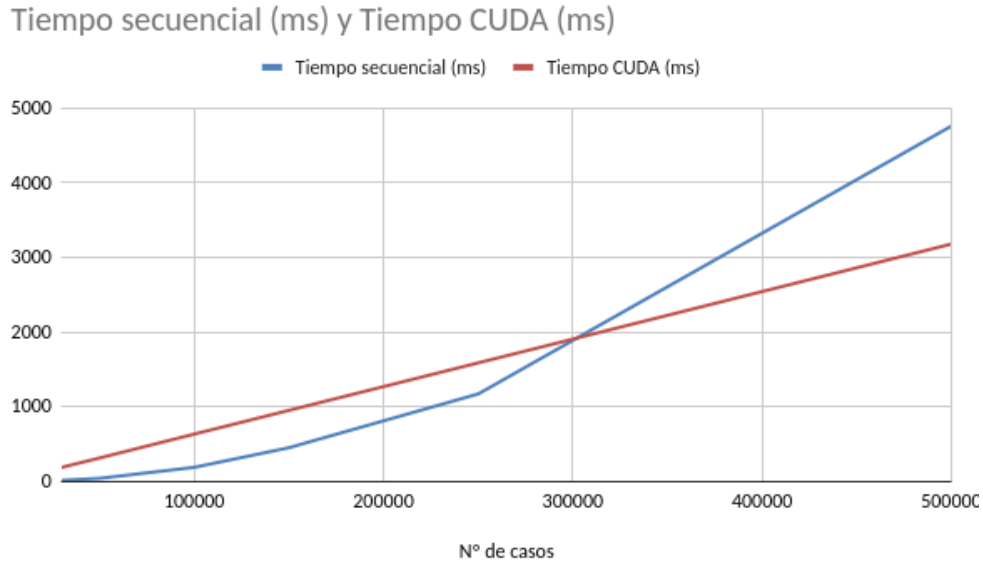

Figura 4: Tiempo de ejecución - Método de Simpson