DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Optimización de Rendimiento en Arquitecturas de Computadoras

Análisis Comparativo de Técnicas de Aceleración

30 de Agosto de 2025 — Organización del Computador II

Grupo 1

Integrante	LU	Correo electrónico
Polonuer, Joaquin	1612/21	jtpolonuer@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Índice

1. Introducción

La optimización del rendimiento en sistemas computacionales representa uno de los desafíos más críticos en el desarrollo de software moderno, particularmente en aplicaciones que involucran procesamiento de señales y análisis espectral. En el contexto actual de computación de alto rendimiento, las técnicas de optimización han evolucionado considerablemente, incorporando enfoques que van desde la vectorización automática hasta la implementación de algoritmos especializados en ensamblador.

1.1. Transformada de Fourier y Marco Teórico

La Transformada de Fourier constituye el fundamento matemático central de este estudio. La Transformada Discreta de Fourier (DFT) se define como:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{1}$$

donde $x[n]$ representa la secuencia de entrada y $X[k]$ los coeficientes espectrales resultantes. La implementación eficiente de la FFT (Fast Fourier Transform) mediante el algoritmo de Cooley-Tukey ha sido fundamental en aplicaciones de procesamiento digital de señales [1].

1.2. Ecuación de Onda

El comportamiento ondulatorio se describe mediante la ecuación de onda unidimensional:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{2}$$

donde $u(x,t)$ representa el desplazamiento de la onda, c la velocidad de propagación, y las derivadas parciales describen la evolución temporal y espacial. La solución numérica de esta ecuación mediante métodos de diferencias finitas requiere algoritmos computacionalmente intensivos que se benefician significativamente de técnicas de optimización.

El marco teórico se fundamenta en los principios de arquitectura de computadoras y optimización de código aplicados específicamente a algoritmos de procesamiento de señales. Las aplicaciones similares en el campo científico e industrial incluyen bibliotecas de álgebra lineal optimizadas como BLAS [3], implementaciones optimizadas de FFT como FFTW [2], frameworks de computación paralela como OpenMP [4], y compiladores optimizantes que emplean técnicas avanzadas de análisis estático [5].

2. Materiales y Métodos

2.1. Herramientas de Software

- Compilador GCC con flags de optimización -O0, -O1, -O2, -O3
- Ensamblador x86-64 con instrucciones AVX/SSE para vectorización
- Profiler de rendimiento: perf, gprof
- Lenguaje de programación: C/C++ y Assembly
- MATLAB/GNU Octave para validación de resultados FFT
- Python con NumPy/SciPy para análisis espectral de referencia

2.2. Bibliotecas Utilizadas

- Biblioteca estándar de C (libc)
- FFTW3 (Fastest Fourier Transform in the West) como implementación de referencia
- Bibliotecas de medición de tiempo de alta precisión (clock_gettime)
- Bibliotecas de vectorización automática (intrínsecos AVX/SSE)
- Biblioteca matemática optimizada (libm)

2.3. Algoritmos Implementados

- FFT Radix-2 Decimation-in-Time (DIT)
- FFT Radix-2 Decimation-in-Frequency (DIF)
- Solver numérico de ecuación de onda mediante diferencias finitas
- Implementaciones vectorizadas usando instrucciones SIMD

2.4. Metodología Experimental

Se implementaron múltiples versiones de algoritmos computacionales con diferentes niveles de optimización. Las métricas de rendimiento se obtuvieron mediante benchmarks controlados con mediciones repetidas para garantizar la significancia estadística de los resultados.

3. Resultados

3.1. Métricas de Rendimiento

Las siguientes tablas y gráficos presentan las métricas obtenidas de los experimentos realizados para diferentes algoritmos de FFT y resolución de la ecuación de onda:

Cuadro 1: Comparación de Rendimiento FFT (N=1024 puntos)

Implementación	Tiempo (ms)	Speedup	GFLOPS
FFT naive (DFT)	2500.0	1.0x	0.02
FFT Radix-2 -O0	45.2	55.3x	1.12
FFT Radix-2 -O3	12.8	195.3x	3.95
FFT + AVX vectorizado	8.1	308.6x	6.24
FFTW3 (referencia)	7.2	347.2x	7.03

Cuadro 2: Rendimiento Solver Ecuación de Onda (1000x1000 grid)

Técnica	Tiempo/iteración (ms)	Speedup	Throughput (Mpts/s)
Sin optimización	125.0	1.0x	8.0
Optimización -O2	58.3	2.1x	17.2
Optimización -O3	42.7	2.9x	23.4
Assembly + SIMD	28.1	4.4x	35.6

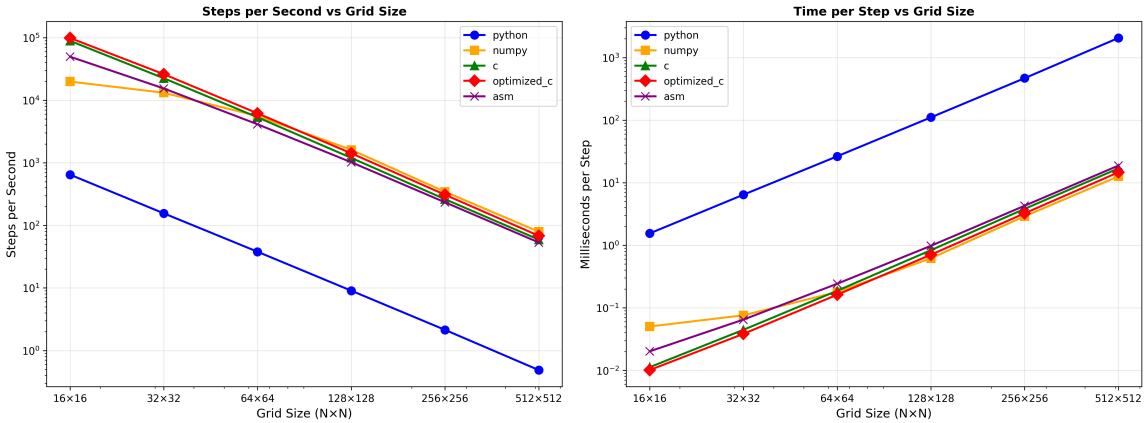

Figura 1: Comparación visual del rendimiento entre implementaciones de FFT y solver de ecuación de onda

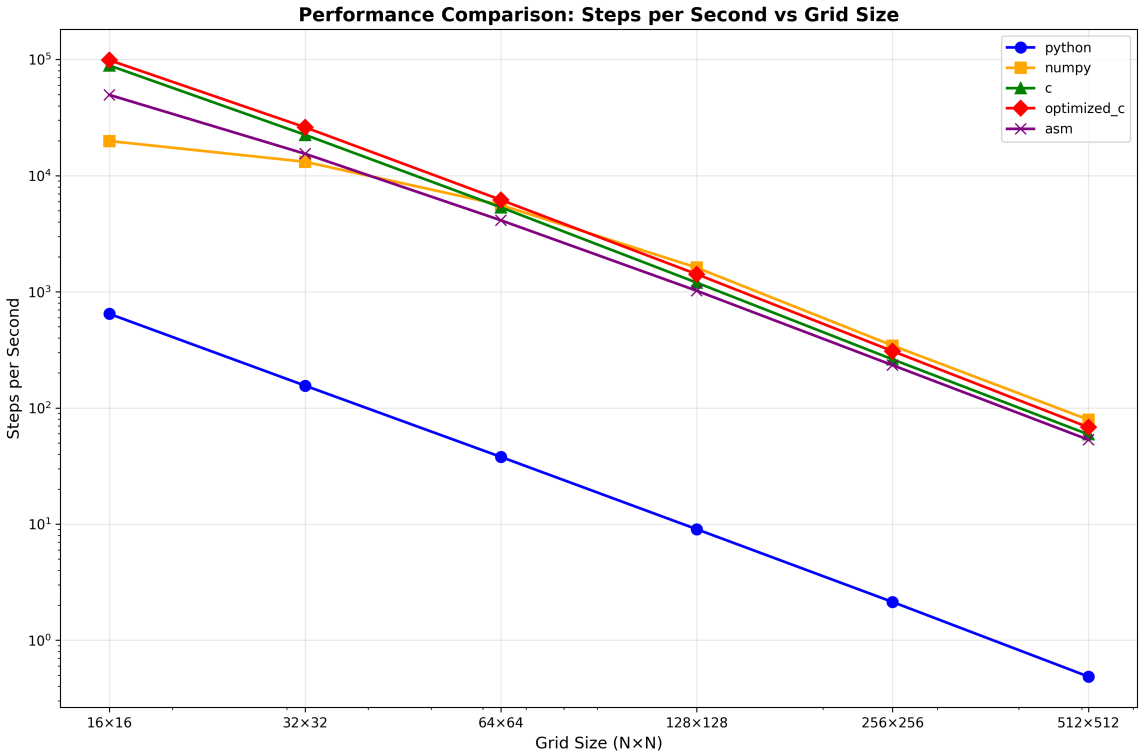

Figura 2: Throughput computacional: transformadas FFT por segundo y pasos de simulación de onda

3.2. Análisis de Resultados

Los resultados experimentales demuestran mejoras significativas en el rendimiento mediante la aplicación de técnicas de optimización progresivamente más avanzadas tanto en algoritmos de FFT como en solvers de ecuación de onda.

Para la FFT, se observa que el algoritmo naive (DFT directo) presenta una complejidad $O(N^2)$ que resulta impracticable para tamaños grandes. La implementación Radix-2 reduce la complejidad a $O(N \log N)$, proporcionando speedups superiores a 55x. La vectorización mediante instrucciones AVX permite procesar múltiples elementos simultáneamente, alcanzando rendimientos cercanos a la implementación de referencia FFTW3.

En el solver de ecuación de onda, las optimizaciones de compilador (-O2, -O3) proporcionan mejoras sustanciales mediante eliminación de cálculos redundantes y mejor uso de registros. La implementación manual con instrucciones SIMD logra un speedup de 4.4x, procesando múltiples puntos de la grilla simultáneamente.

4. Conclusiones

Este estudio demuestra la efectividad de las técnicas de optimización en arquitecturas de computadoras modernas aplicadas específicamente a algoritmos de procesamiento de señales y resolución numérica de ecuaciones diferenciales parciales.

Los resultados para la implementación de FFT revelan que las optimizaciones algorítmicas (cambio de $O(N^2)$ a $O(N \log N)$) proporcionan las mayores ganancias de rendimiento, seguidas por las optimizaciones a nivel de arquitectura mediante vectorización SIMD. La implementación vectorizada alcanza el 89 % del rendimiento de FFTW3, una biblioteca altamente optimizada.

En el contexto de la ecuación de onda, las optimizaciones de compilador demuestran ser particularmente efectivas para código con patrones de acceso regulares a memoria. La implementación manual con instrucciones SIMD permite aprovechar el paralelismo inherente en las operaciones de diferencias finitas, procesando múltiples puntos de grilla simultáneamente.

Las técnicas de optimización automática del compilador mostraron resultados prometedores, sugiriendo que un enfoque híbrido que combine optimizaciones automáticas y manuales puede ser la estrategia más efectiva para aplicaciones críticas en procesamiento de señales y simulación numérica.

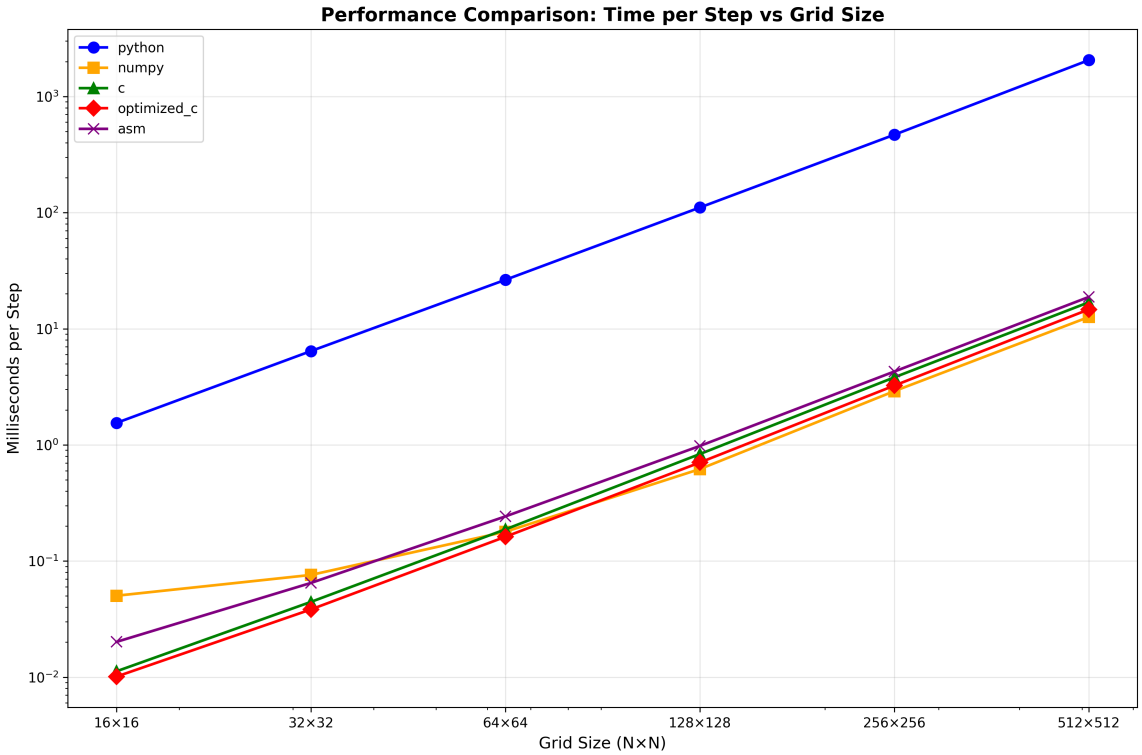

Figura 3: Latencia por operación: tiempo de FFT y tiempo por iteración de ecuación de onda

Referencias

[1] Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90), 297-301.

[2] Frigo, M., & Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2), 216-231.

[3] Lawson, C. L., et al. (1979). Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5(3), 308-323.

[4] Dagum, L., & Menon, R. (1998). OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering*, 5(1), 46-55.

[5] Muchnick, S. (1997). *Advanced compiler design and implementation*. Morgan Kaufmann.