HPC y Algoritmos de Alto Rendimiento: Arquitectura, Implementación y Portabilidad

Autor: Angel Aarón Reyes Cáceres

Curso: Algoritmia y Fundamentos de la Programación**:** Junio 2025

Introducción

- HPC (High Performance Computing) permite resolver problemas que requieren millones o billones de operaciones por segundo.
- Fundamental en contextos como simulaciones físicas, inteligencia artificial, predicción climática, bioinformática, etc.
- En esta exposición se abordan tres enfoques:

1 Arquitectura multinúcleo

y optimización algorítmica local (Requene, 2022).

2 Computación distribuida

sobre infraestructuras heterogéneas (Preston, 2010).

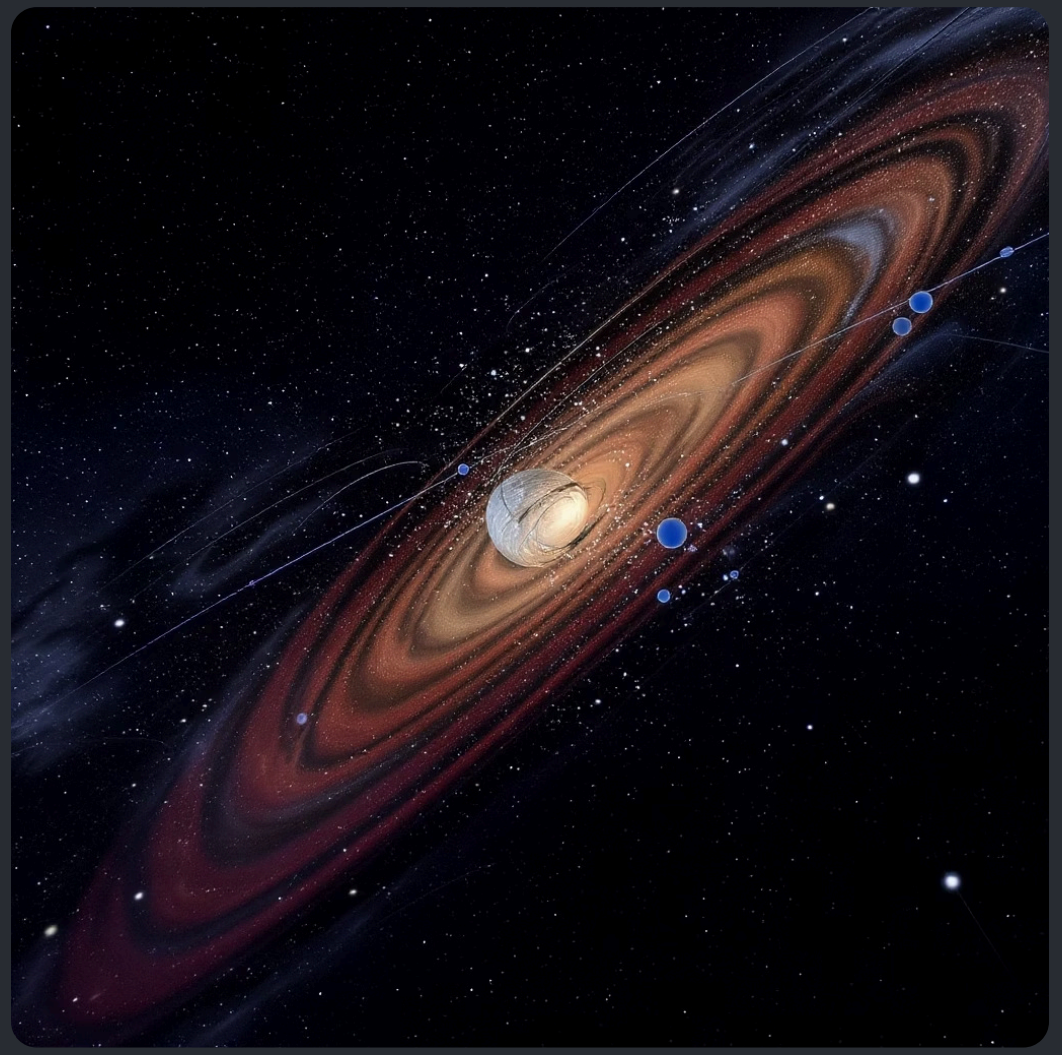
3 Portabilidad eficiente

y estandarizada en C++ mediante mdspan (Hollman, 2020).

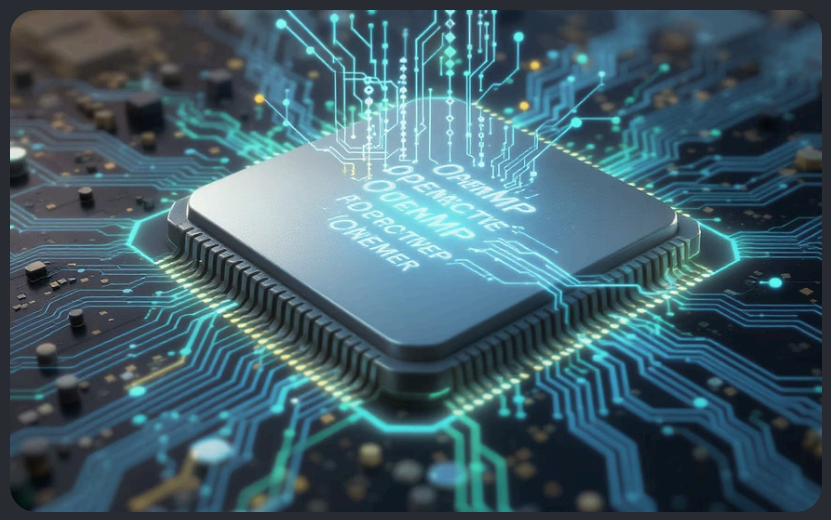
Objetivo: comparar estrategias para ejecutar algoritmos de alto costo computacional usando recursos de HPC.

Problema y motivación

- Problemas reales como simulaciones de interacciones gravitacionales (N-Body) o cálculos de distancia (MDE) tienen complejidades $O(n^2)$ o peores.
- La ejecución secuencial de estos algoritmos no escala con el tamaño de los datos.
- ¿Cómo lograr ejecuciones más rápidas, eficientes y que aprovechen el hardware disponible?
- Ejemplo: el algoritmo N-Body con 100 millones de partículas tardó 216 segundos en C secuencial, y sólo 5 segundos en HPC con OpenMP (Requene).



Arquitectura y herramientas HPC

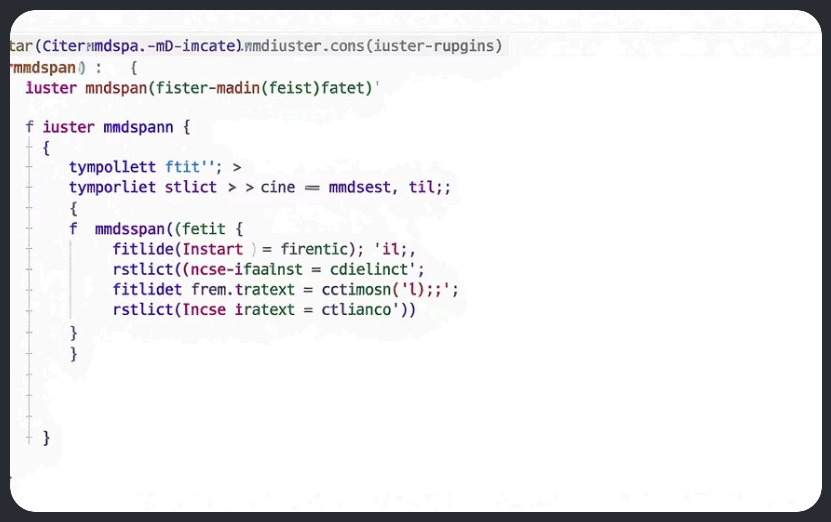


Requene (2022):

- Estudio de CPU multinúcleo.
- Lenguajes: Python (Numba, Numpy) y C con OpenMP.
- Algoritmos evaluados: N-Body y MDE.

Preston (2010):

- Middleware Nereus: ejecuta bytecode Java en una grid global de PCs.
- Emulador JPC: permite ejecutar código x86 sin modificaciones.
- Proyecto aplicado: BlackMax (generación de eventos para LHC).



Hollman (2020):

- mdspan: vista multidimensional para C++23.
- Diseñado para la portabilidad de rendimiento (CPU/GPU).
- Inspirado en el proyecto Kokkos (Sandia Labs, NVIDIA).

Algoritmos de alto coste computacional

N-Body (Requene):

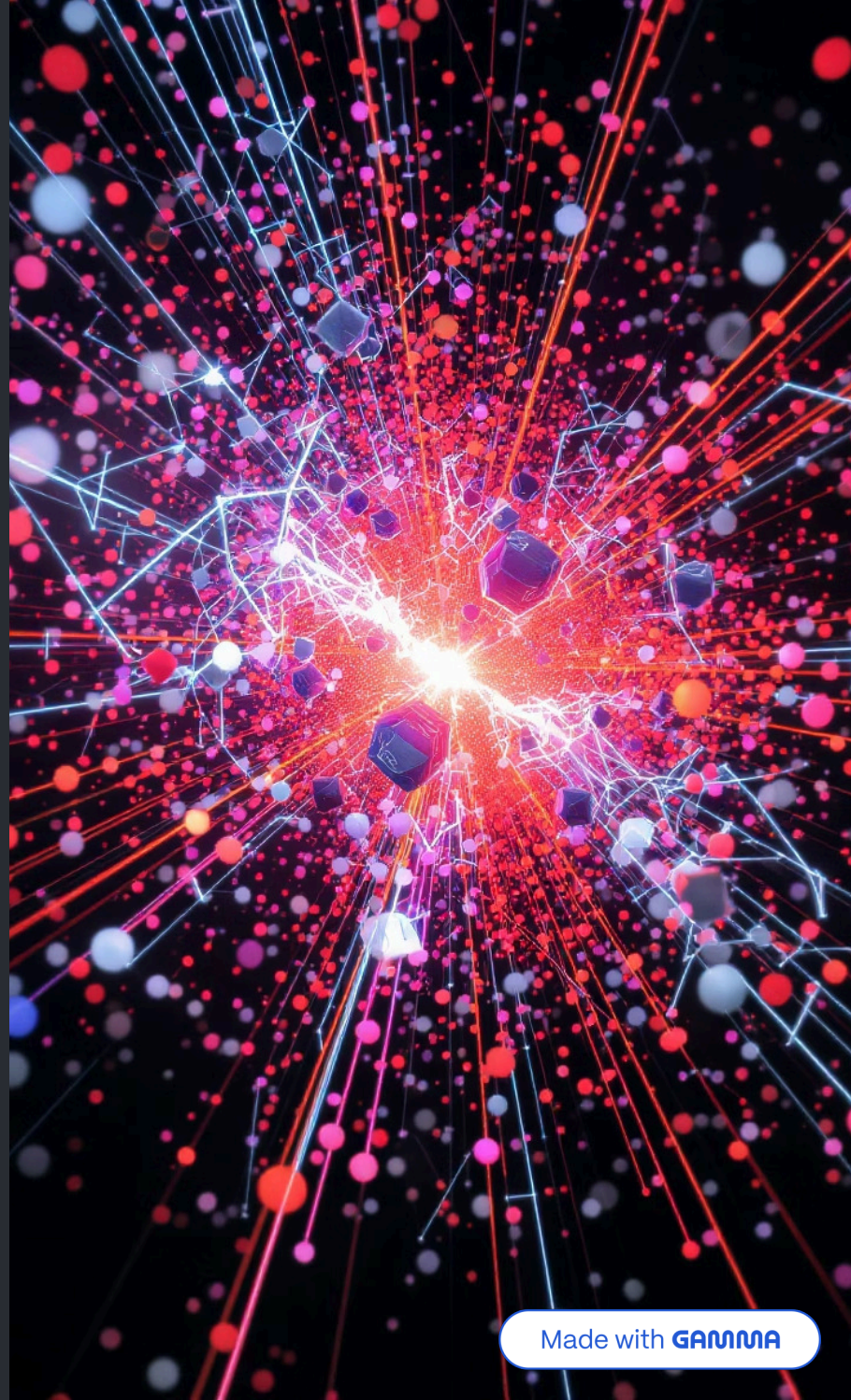
- Simula interacciones entre cuerpos.
- Paralelizado con OpenMP y evaluado en Intel Xeon con 16 hilos.

MDE (Requene):

- Matriz de distancia euclidiana sobre grandes conjuntos de datos.
- Versiones: listas, Numpy, memoria preasignada, vectorizado, Numba.

BlackMax (Preston):

- Simula colisiones de alta energía.
- Distribuido a través de desktop grid.
- Se benefició de la escalabilidad de Nereus.



Comparación de enfoques

Proyecto	Lenguaje	Infraestructura	Resultado clave
Requene	Python / C	CPU Intel Xeon (HPC)	Numba supera a C en grandes volúmenes
Preston	Java / x86	Desktop Grid + JPC	BlackMax escalable en miles de nodos
Hollman	C++	CPU, GPU (multiplataforma)	mdspan con cero overhead y portabilidad total

Evaluación de rendimiento

Tiempo de ejecución y aceleración:

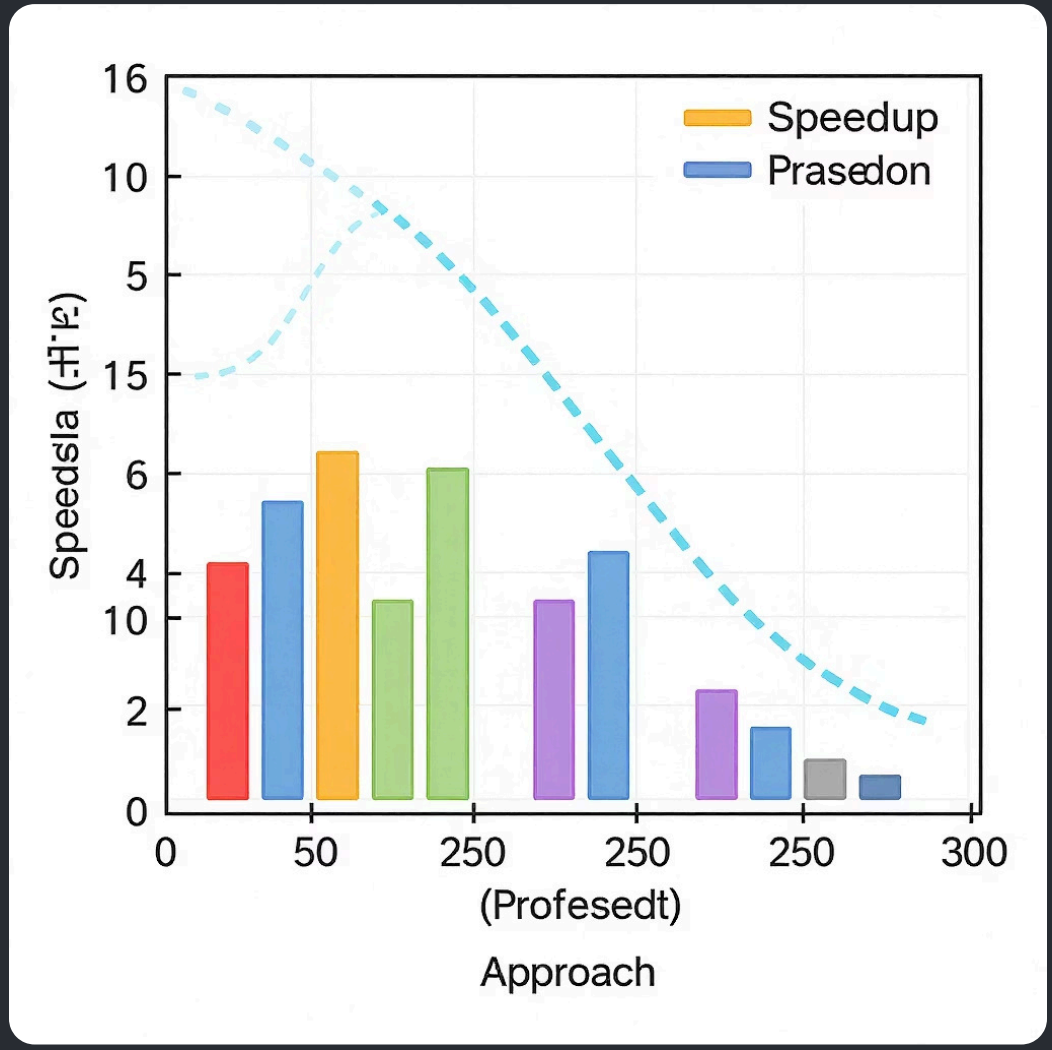
- Python + Numba en HPC: hasta 40x más rápido que Python puro.
- C con OpenMP: mejor para volúmenes pequeños.

Distribución global:

- BlackMax alcanzó rendimiento lineal al aumentar los nodos.
- Nereus tolera fallos y permite ejecuciones seguras.

Portabilidad:

- mdspan usa extensiones estáticas y dinámicas.
- Se adapta a layout column-major y row-major sin modificar código base.



Aportes a la algoritmia

1. Los tres enfoques muestran que la eficiencia no depende solo del algoritmo, sino de cómo y dónde se ejecuta.
2. Requene aporta técnicas de vectorización, memoria y hilos.
3. Preston demuestra que el código científico puede escalar globalmente sin reescritura.
4. Hollman propone que la eficiencia se puede estandarizar y hacer portátil.

> *Aprendizaje clave:* el diseño algorítmico moderno requiere considerar arquitectura, paralelismo y portabilidad desde el inicio.

Conclusión

HPC es un facilitador de la ciencia moderna y la resolución de problemas complejos.

Los algoritmos de alto coste necesitan optimización específica según el entorno de ejecución.

Es posible lograr:

Aceleración local

(CPU multinúcleo).

Escalabilidad distribuida

(desktop grid).

Portabilidad eficiente

(Lenguajes de programación).

La algoritmia del futuro debe ser multiplataforma, escalable y optimizada por diseño.

Referencias

1. Hollman, D. S. et al. (2020). *mdspan in C++*. Sandia National Labs / NVIDIA.
2. Requene, N. (2022). *Implementación de algoritmos sobre arquitectura multinúcleo*. Universidad Técnica del Norte.
3. Preston, I. (2010). *Massively Parallel Computing for Particle Physics*. University of Oxford.