PROYECTO FIN DE CARRERA

ANÁLISIS, DESARROLLO Y OPTIMIZACIÓN DE FUNCIONES DE GREEN

Carlos Pérez Alcaraz
Universidad de Murcia

Índice.

- Introducción.
  - Funciones de Green.
  - Objetivos.
  - Metodología.
- Funciones de Green unidimensionales.
  - Análisis del algoritmo secuencial.
  - Estudio del paralelismo.
- Funciones de Green bidimensionales.
  - Análisis del algoritmo secuencial.
  - Estudio del paralelismo.
- Trabajos futuros.

Introducción. Funciones de Green.

- Las ecuaciones diferenciales constituyen una herramienta indispensable para el modelado de sistemas.
- Diferentes métodos de resolución:
  - Separación de variables.
  - Series de Fourier.
  - Funciones de Green.

Introducción. Funciones de Green.

- Resolución de ecuaciones diferenciales no homogéneas bajo ciertas condiciones de contorno.
- Diversas aplicaciones en ingeniería.
- En particular, nos interesan las funciones de Green de las guías de ondas.
  - Sencillez analítica.
  - Alto coste computacional.

Introducción. Objetivos.

- Aceleración de una aplicación científica basada en un problema real: funciones de Green de las guías de ondas.
  - Método de cálculo de Ewald.
  - Análisis en función del número de dimensiones.
- Estudio de diferentes técnicas y herramientas para su paralelización en diversos sistemas.

Introducción. Metodología.

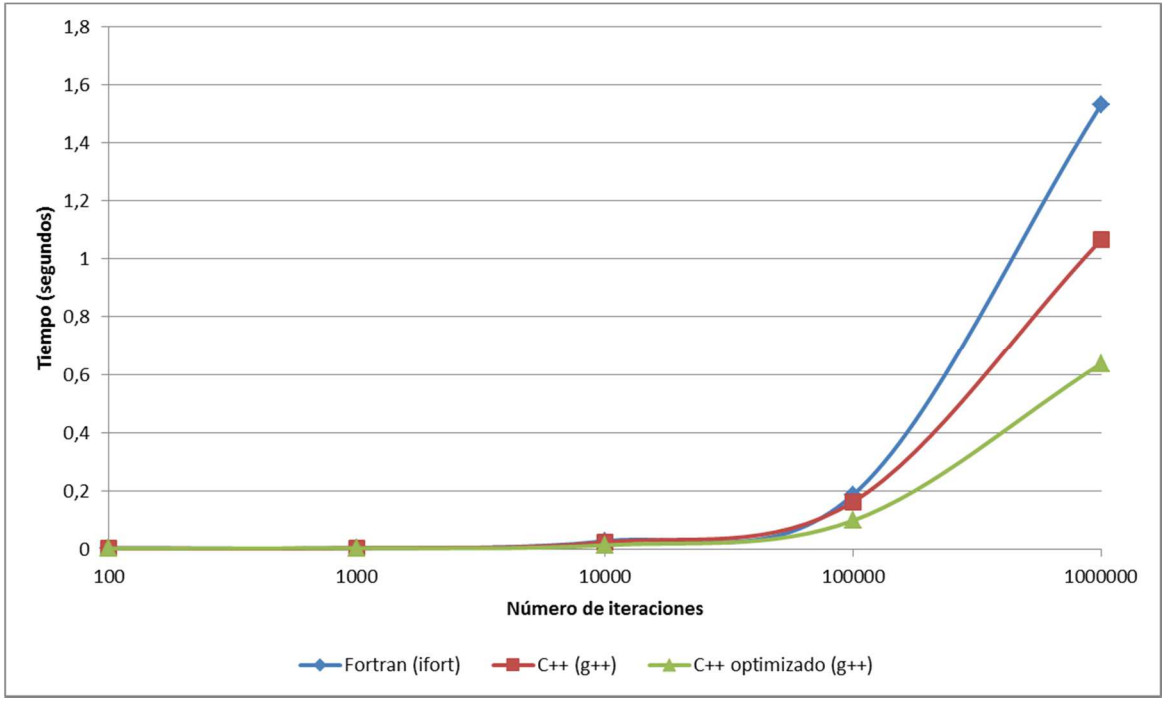
- Desarrollar y optimizar una versión en lenguaje C++.
- Identificar zonas susceptibles de ser paralelizadas.
- Analizar diferentes posibilidades de implementación.
- Evaluar y comparar rutinas desarrolladas.

Funciones de Green unidimensionales.

- Guía de placas paralelas donde situamos n puntos fuente y m puntos de observación.
- Para cada pareja de puntos evaluar nmod términos de la serie dada por el método de Ewald.
- Orden de complejidad O(n * m * nmod).

Análisis del algoritmo secuencial.

- Utilizamos callgrind para obtener un perfil de ejecución, a fin de identificar las partes del código más costosas y tratar de optimizarlas.
- Mejora de hasta 3 veces el tiempo de ejecución:

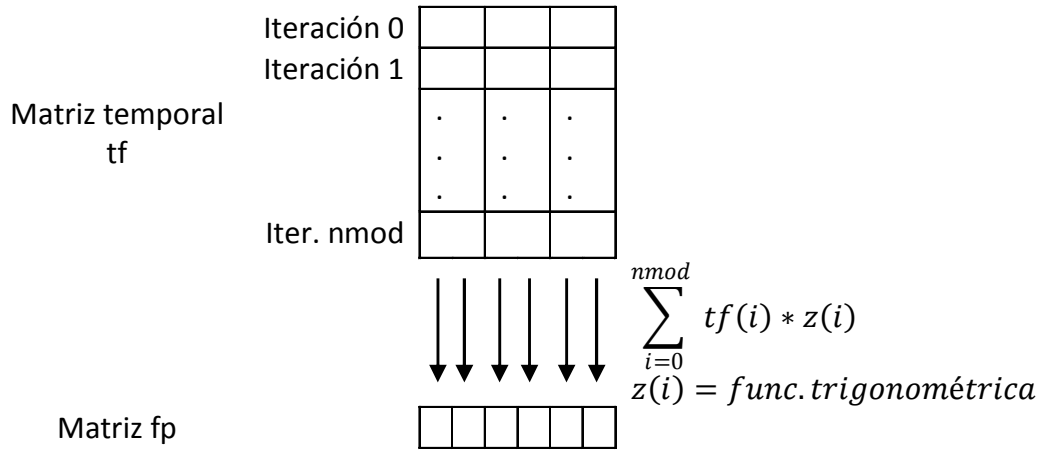


Estudio del paralelismo.

- Dos niveles de paralelismo:
  - Grano fino: cálculo de las iteraciones de una única función de Green.
  - Grano grueso: cálculo de varias funciones de Green al mismo tiempo.
- Dos posibles funciones a paralelizar.
  - spectgf: 76% del tiempo total. Sin dependencias de datos entre iteraciones.
  - dsum: 21% del tiempo total. Con dependencias de datos entre iteraciones.

Estudio del paralelismo.

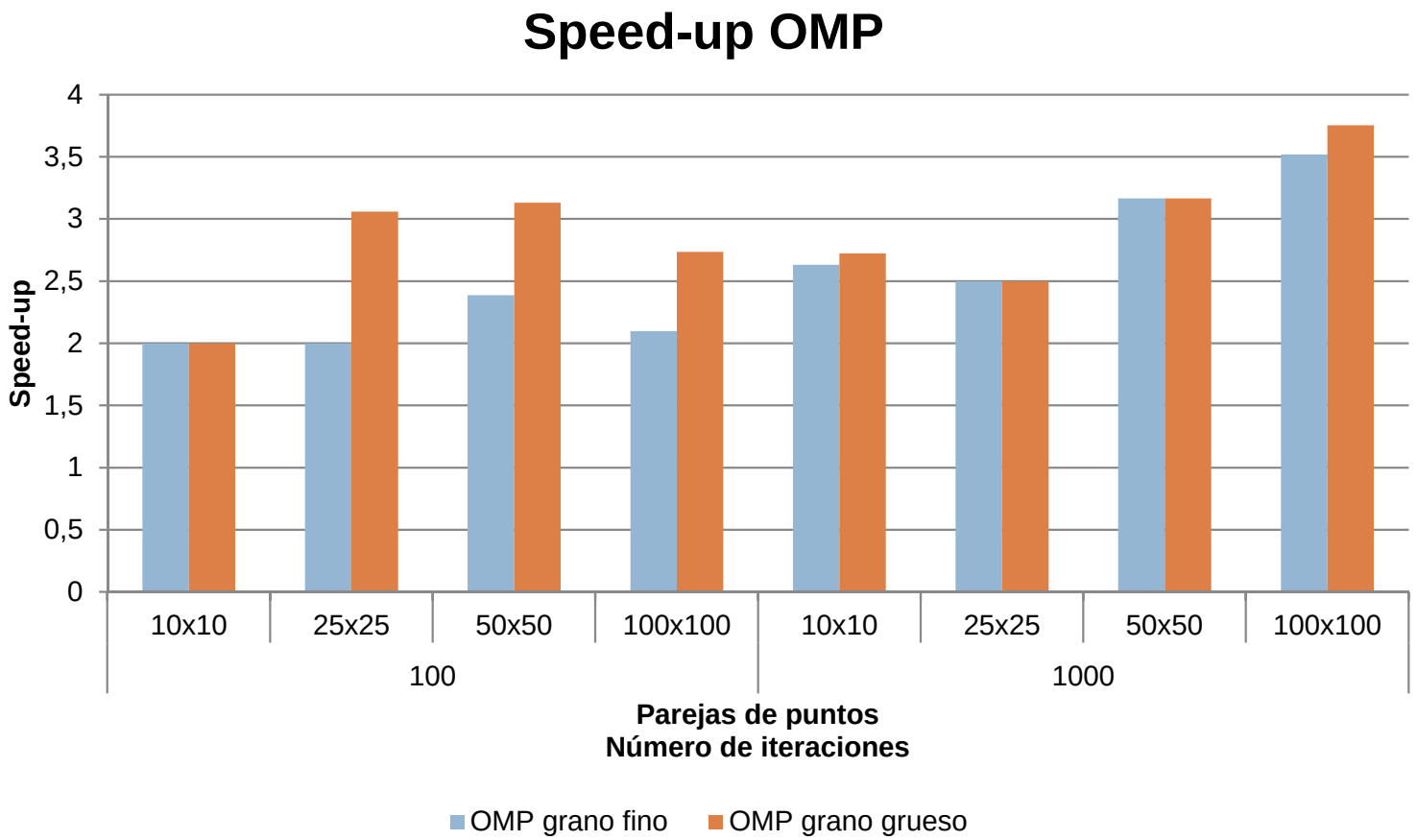
- Si nos fijamos en la relación entre ambas funciones:



- Podemos juntar ambas y eliminar la matriz temporal, así como la necesidad de crear dos regiones paralelas.

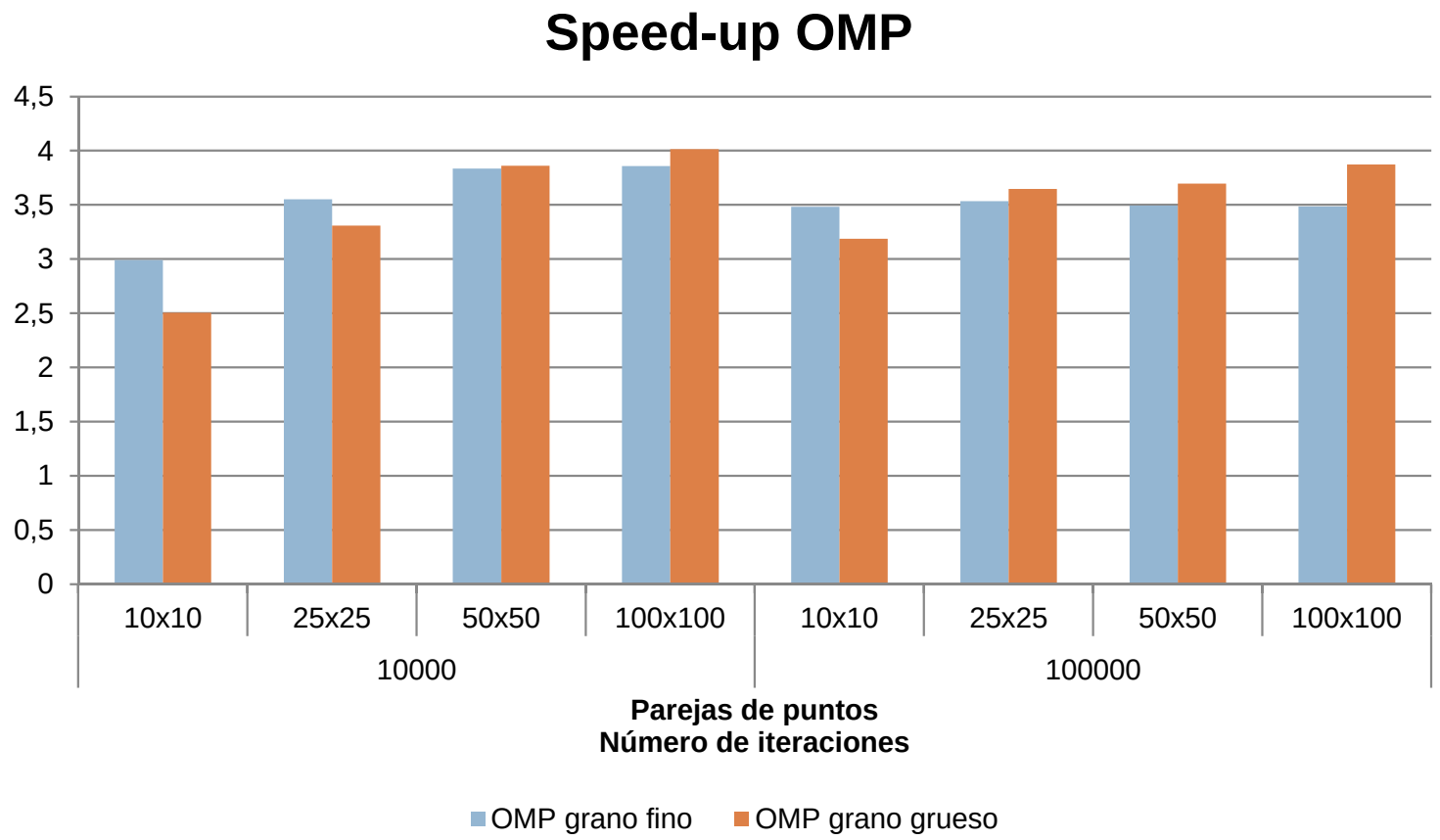
Estudio del paralelismo.

- Para una ejecución con 4 threads:



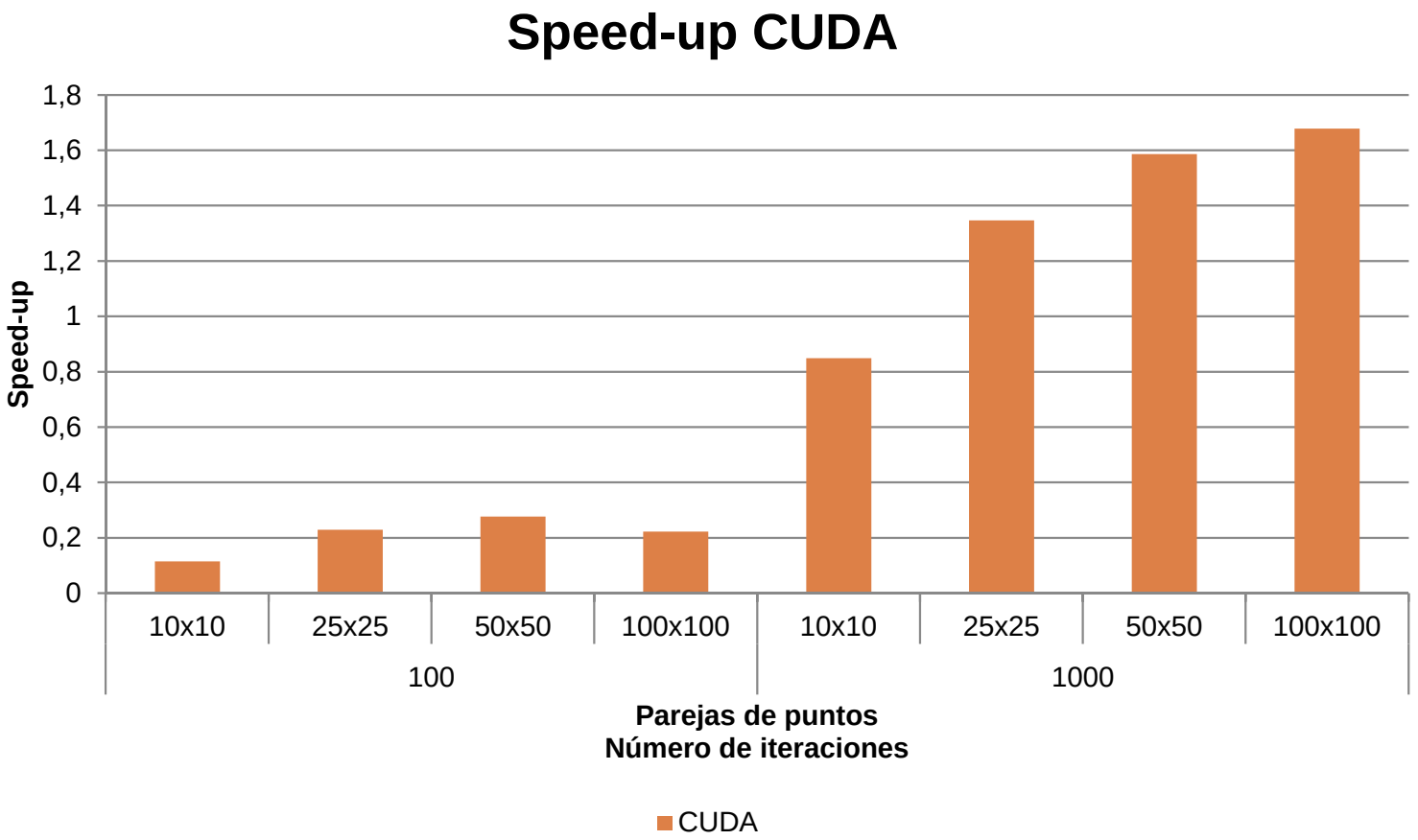
Estudio del paralelismo.

- Para una ejecución con 4 threads:



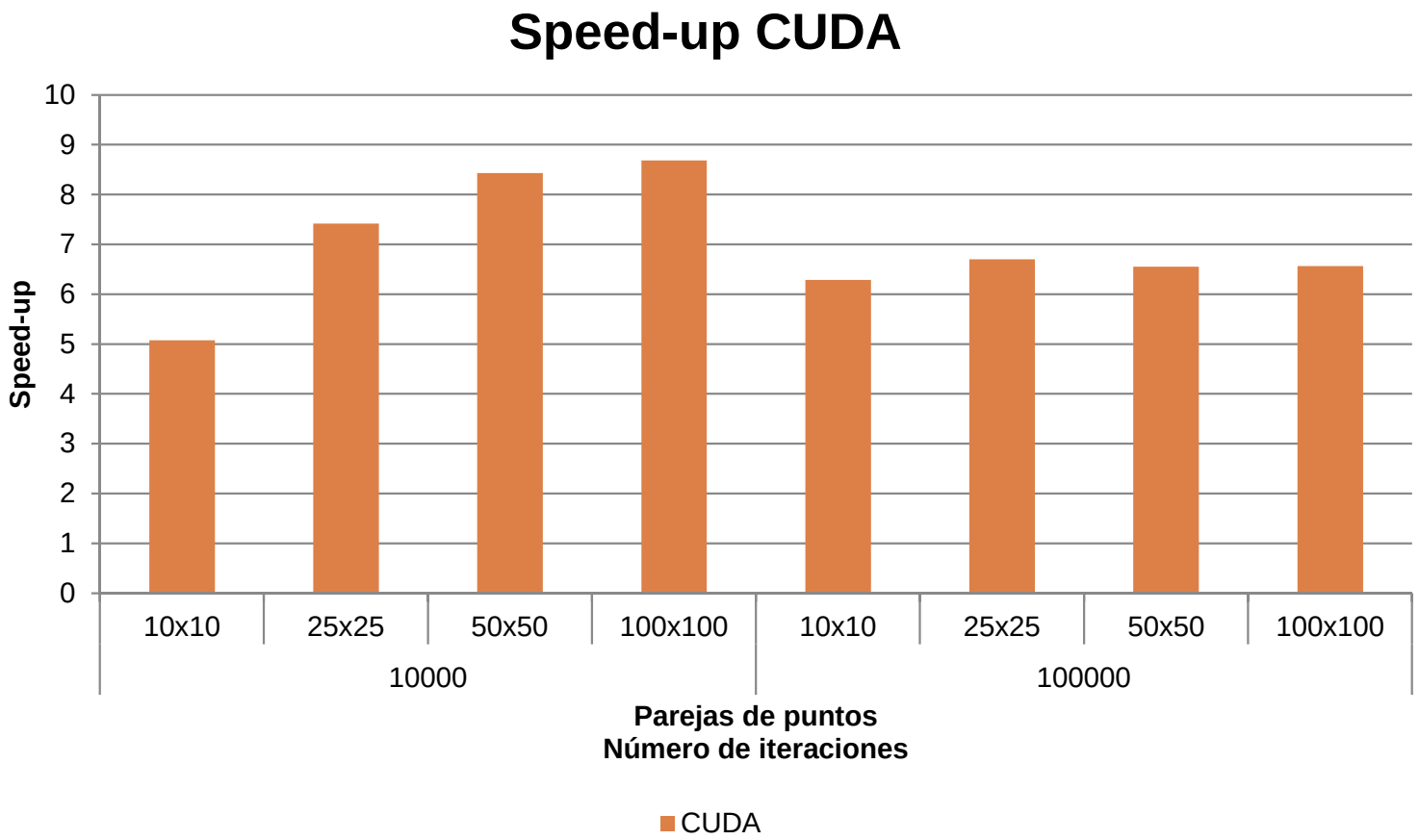
Estudio del paralelismo.

- En una GPU con 112 cores:



Estudio del paralelismo.

- En una GPU con 112 cores:

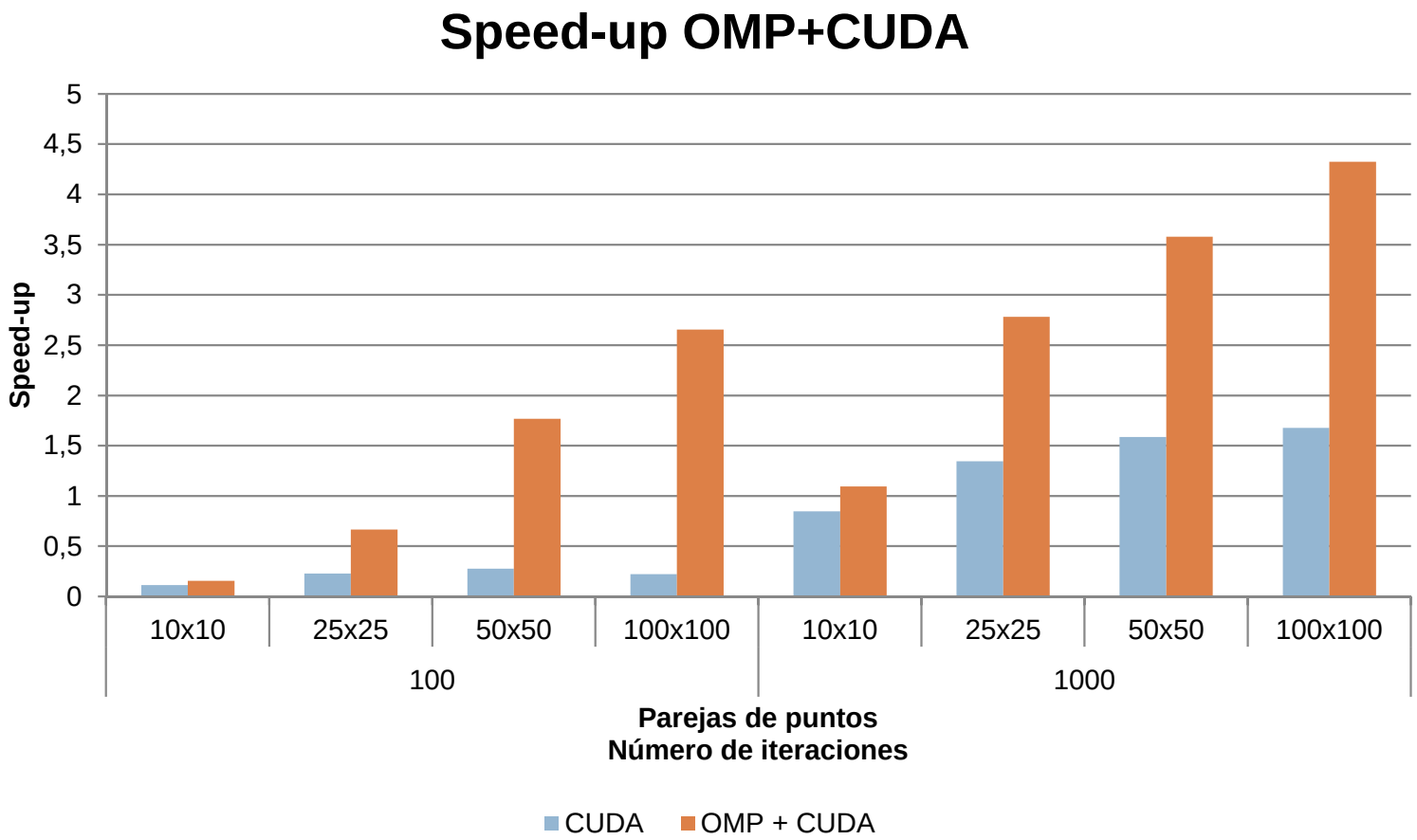


Estudio del paralelismo.

- También se ha desarrollado una versión híbrida para CPU + GPU.
- Partimos de la versión de OMP de grano grueso, y utilizamos un thread adicional para que realice las invocaciones al kernel CUDA.
- En sistemas multi GPU podríamos utilizar otro thread adicional por cada GPU a utilizar.

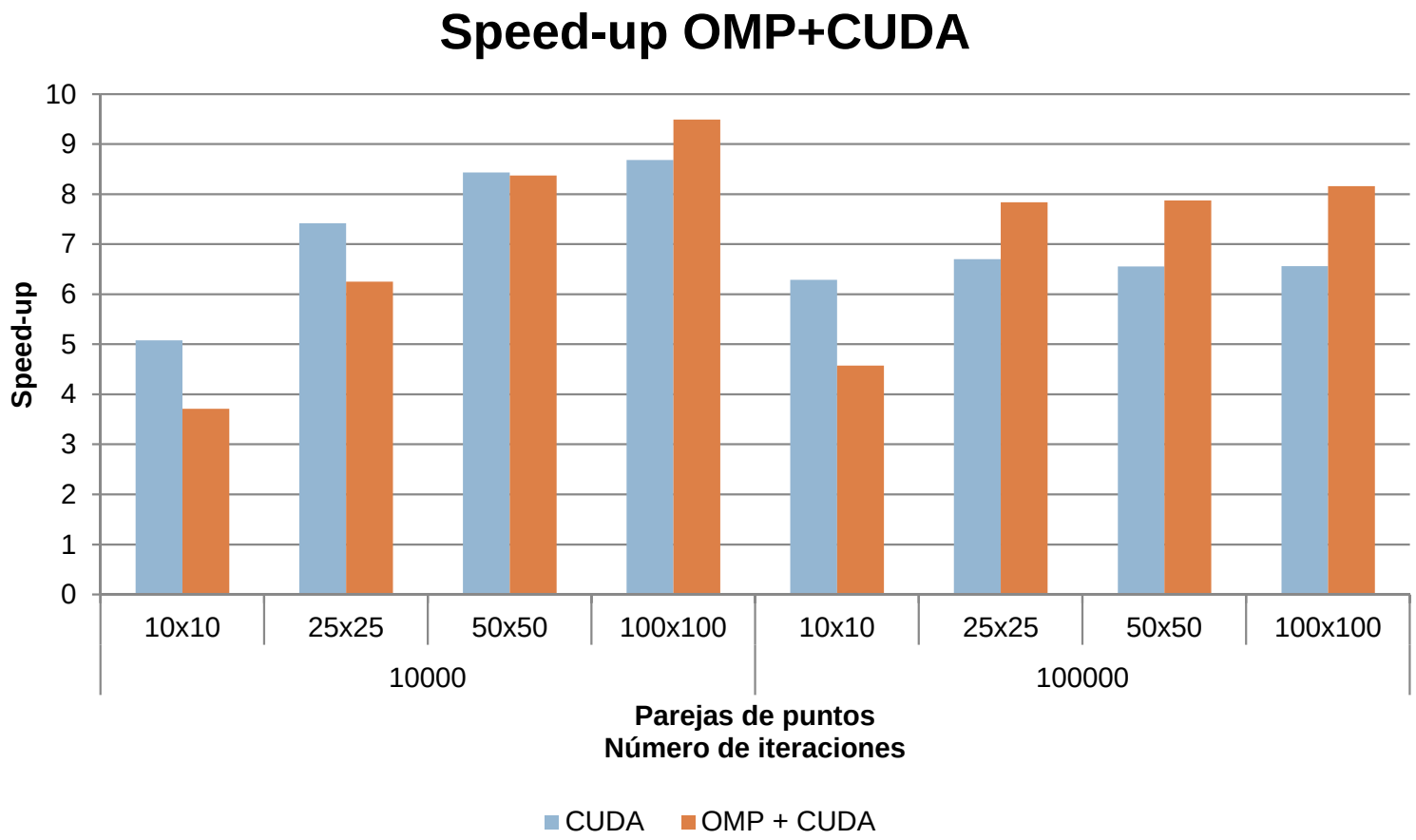
Estudio del paralelismo.

- Los resultados con 4 threads + 1 GPU:



Estudio del paralelismo.

- Los resultados con 4 threads + 1 GPU:



Estudio del paralelismo.

- En resumen:

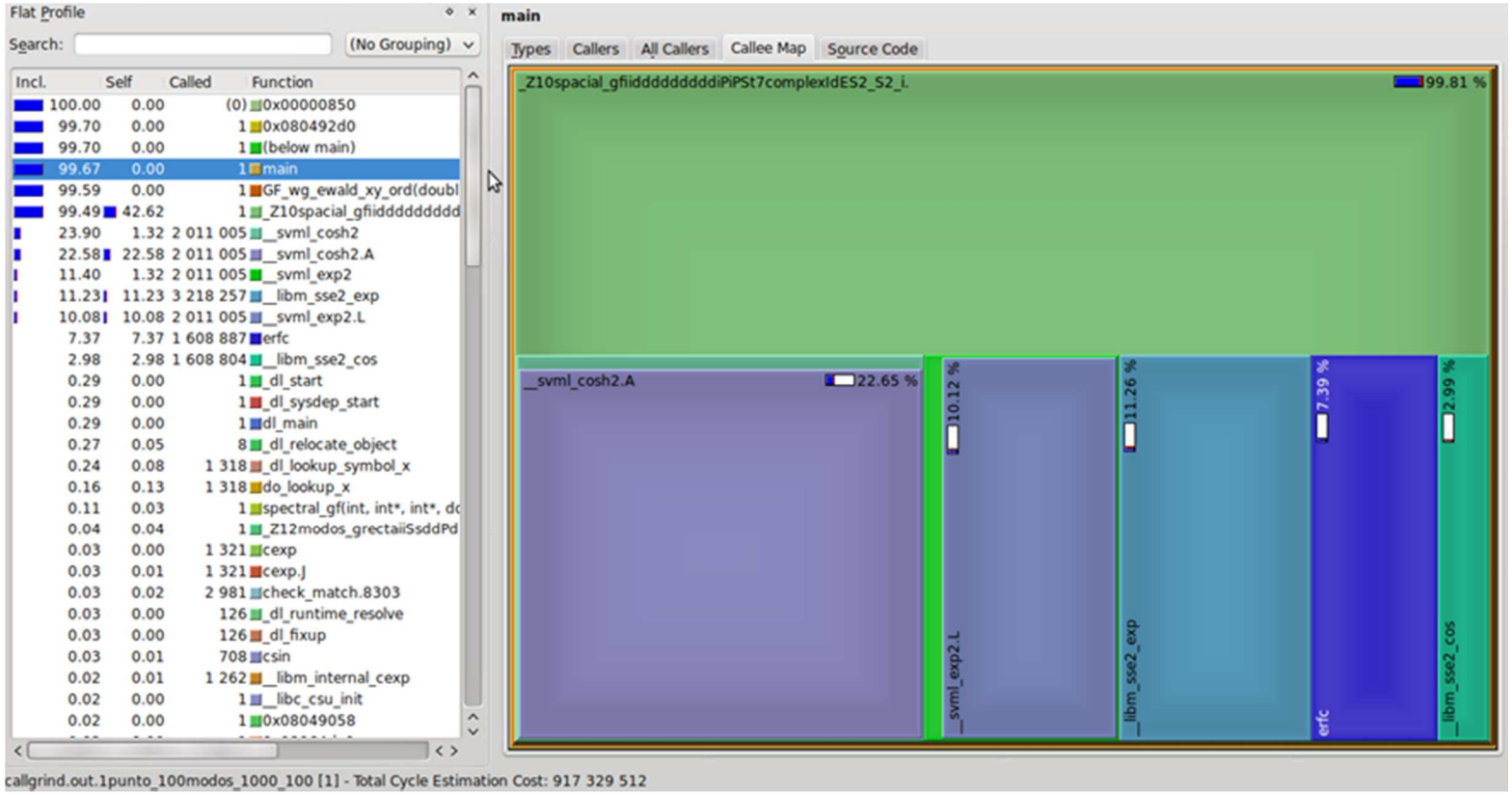
modos	puntos	Fortran	C++ optimizado	OMP grano fino	OMP grano grueso	CUDA	OMP + CUDA
100	10x10	0,015	0,010	0,005	0,005	0,087	0,064
	25x25	0,074	0,052	0,026	0,017	0,227	0,078
	50x50	0,261	0,191	0,080	0,061	0,689	0,108
	100x100	0,833	0,558	0,266	0,204	2,502	0,210
1000	10x10	0,139	0,079	0,030	0,029	0,093	0,072
	25x25	0,635	0,345	0,138	0,138	0,256	0,124
	50x50	2,323	1,314	0,415	0,415	0,828	0,367
	100x100	9,091	5,023	1,427	1,338	2,992	1,161
10000	10x10	1,615	0,772	0,258	0,308	0,152	0,208
	25x25	9,867	4,340	1,221	1,311	0,585	0,694
	50x50	39,207	17,725	4,619	4,589	2,101	2,117
	100x100	156,87	70,798	18,349	17,627	8,150	7,458
100000	10x10	14,445	6,260	1,797	1,963	0,995	1,368
	25x25	89,786	38,688	10,944	10,604	5,771	4,935
	50x50	360,826	149,817	42,868	40,528	22,853	19,023
	100x100	1435,542	598,326	171,615	154,463	91,140	73,298

Funciones de Green bidimensionales.

- Se utiliza un único punto fuente y un conjunto de puntos situados en un plano de observación.
- Para el cálculo mediante el método de Ewald se descompone la serie en dos términos:
  - Parte calculada en el dominio espectral, donde se evalúan nmod modos.
  - Parte calculada en el dominio espacial, mediante el método de imágenes, evaluándose en nimag x mimag imágenes en los ejes x e y.

Funciones de Green bidimensionales.

- Prácticamente todo el tiempo de ejecución se emplea en el cálculo de la parte espacial.



Funciones de Green bidimensionales.

- Consideraremos el número de modos de la parte espectral fijo.
- El orden de ejecución, por tanto, nos queda:

$$O(nmodos, nimag, mimag)$$
$$= O_{spectral}(nmodos) + O_{spatial}(nimag * mimag)$$
$$= O(nimag * mimag)$$

- También tenemos otros parámetros externos, pero los consideraremos fijos en este estudio.

Análisis del algoritmo secuencial.

- En este caso el compilador tiene un gran efecto sobre el código generado:

imágenes	puntos	Fortran (ifort)	C++ (g++)	speed down	C++ (icpc)	Speed up
100	10x10	0,148	0,487	3,290	0,162	0,913
	25x25	0,689	2,680	3,889	0,729	0,945
	50x50	2,584	10,616	4,108	2,767	0,933
	100x100	9,986	42,372	4,243	10,945	0,912
1000	10x10	0,859	3,603	4,194	0,674	1,274
	25x25	5,018	22,272	4,438	4,072	1,232
	50x50	19,659	88,853	4,519	16,209	1,212
	100x100	78,710	355,210	4,512	65,489	1,201
10000	10x10	7,432	33,351	4,487	5,794	1,282
	25x25	46,318	208,091	4,492	36,568	1,266
	50x50	185,205	832,248	4,493	144,560	1,281
	100x100	760,954	3420,284	4,494	576,456	1,320
100000	10x10	74,676	330,927	4,431	56,974	1,310
	25x25	459,856	2067,221	4,495	354,847	1,295
	50x50	1822,058	8195,091	4,497	1491,650	1,221

Estudio del paralelismo.

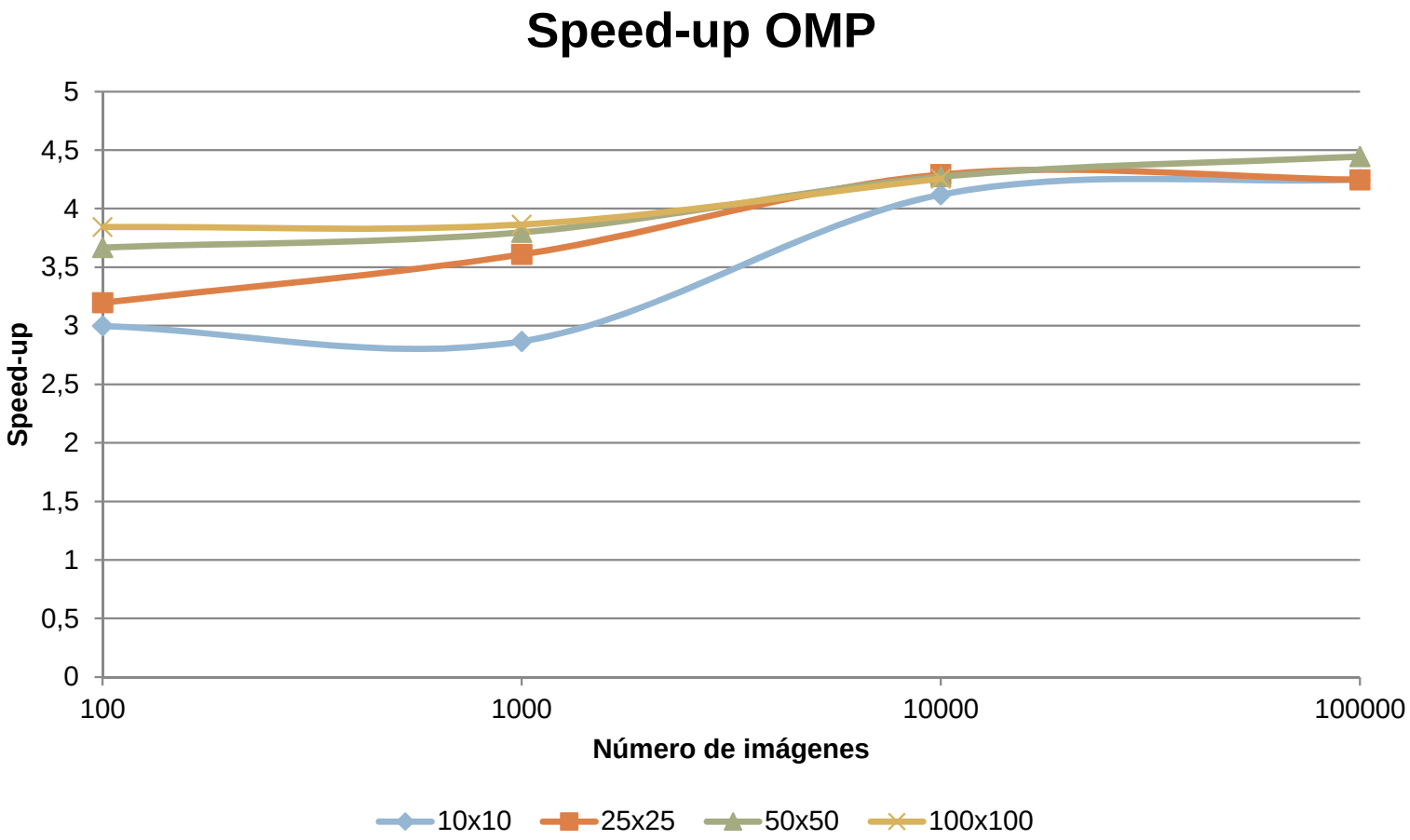
- Al igual que en el caso unidimensional, tenemos dos posibles niveles de paralelismo:
  - Grano fino: cálculo de las iteraciones de la serie dada por el método de Ewald.
  - Grano grueso: cálculo de varias funciones de Green al mismo tiempo.
- Nos centraremos en el paralelismo de grano fino.

Estudio del paralelismo.

- Repartir el cálculo de la serie en un único punto entre el número de threads.
- Cada imagen evaluada en la parte espacial nos da una suma parcial. El resultado es la suma de todos los términos evaluados.
- Utilizar el mecanismo de reducción proporcionado por OMP.

Estudio del paralelismo.

- Ejecución con 4 threads:

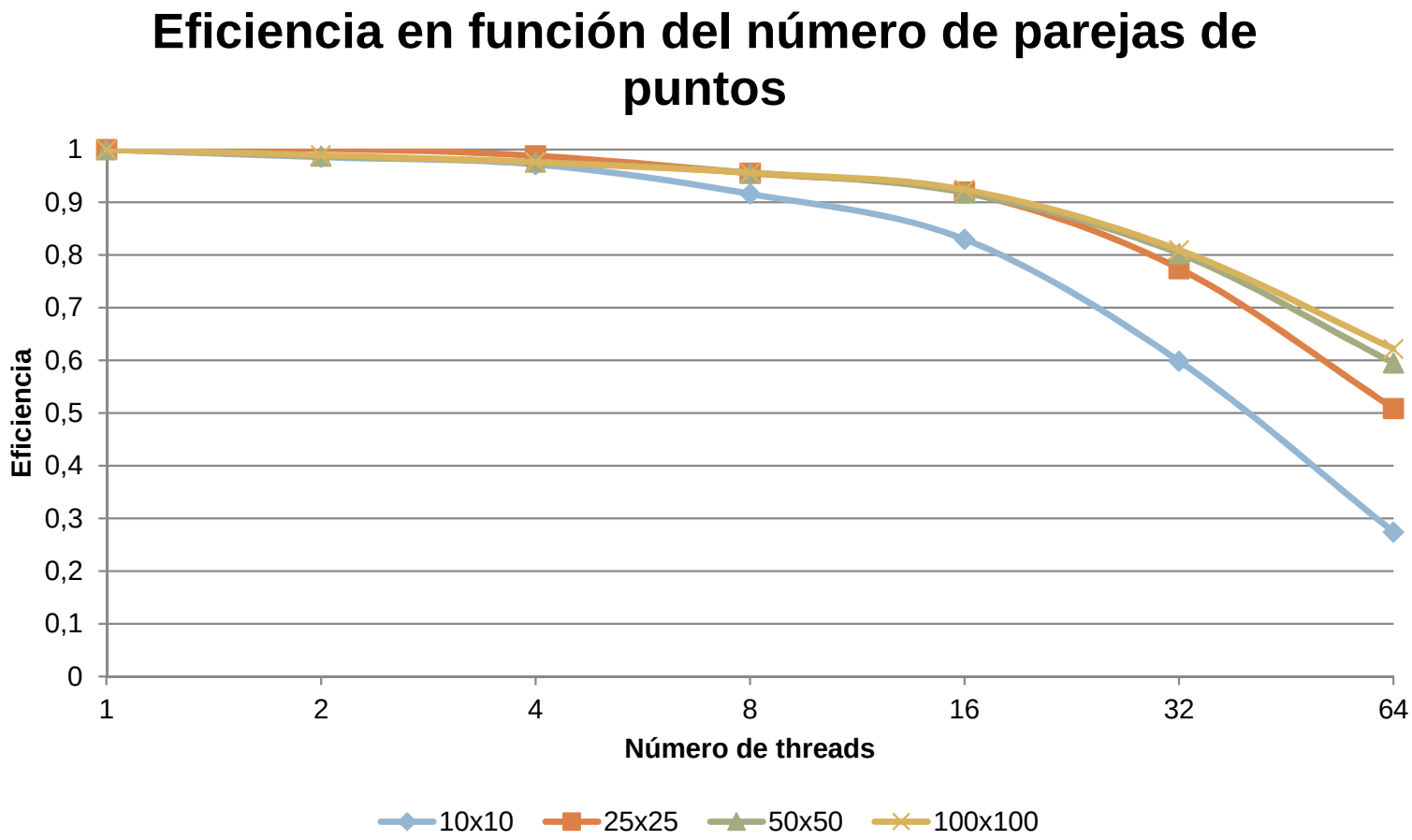


Estudio del paralelismo.

- Es interesante comprobar la escalabilidad de la rutina desarrollada:
  - En función del número de parejas de puntos, para un número de imágenes fijo (10000).
  - En función del número de imágenes a evaluar, fijando el número de puntos (50x50).
- Evaluaremos los resultados en Ben.

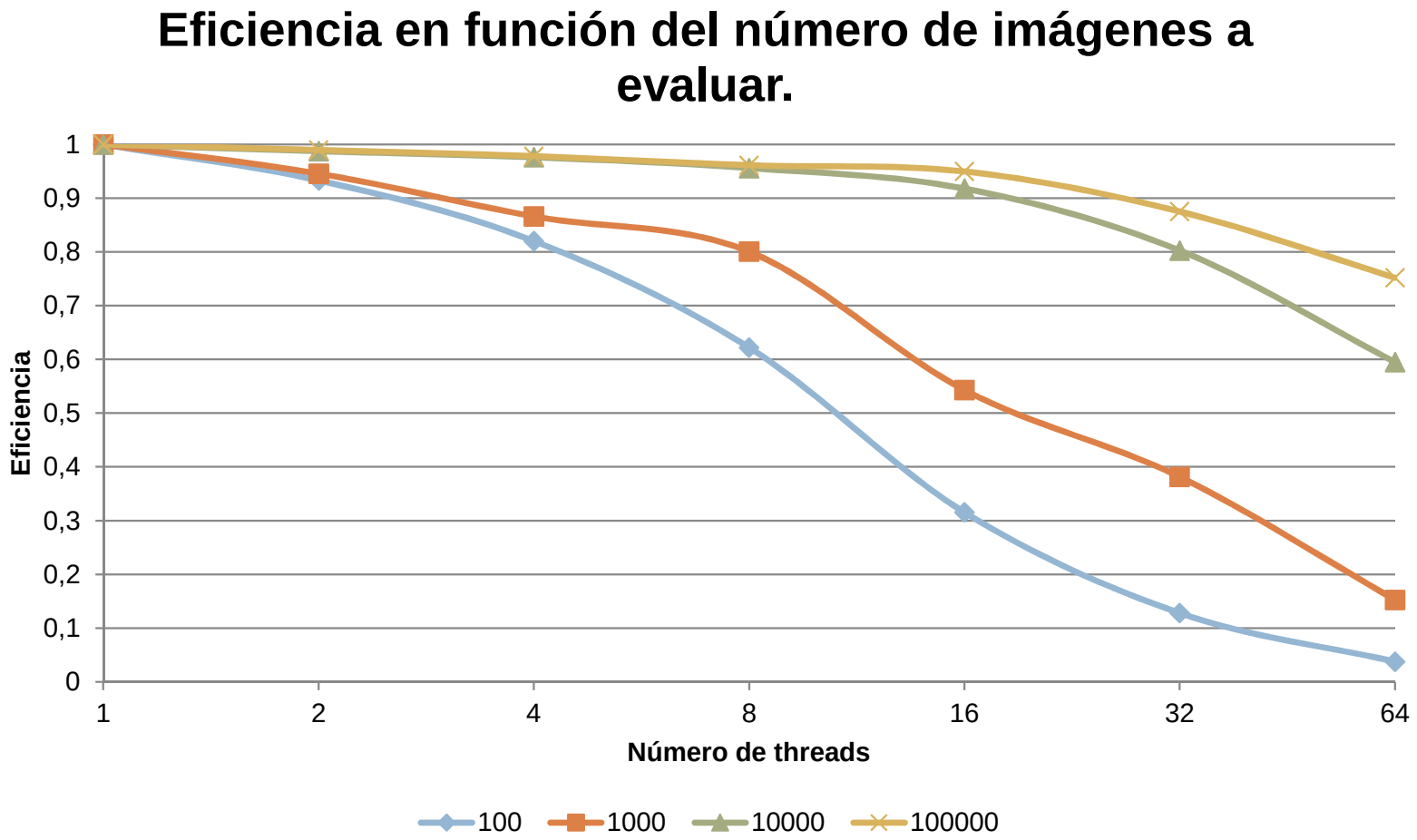
Estudio del paralelismo.

- Resultados en Ben:



Estudio del paralelismo.

- Resultados en Ben:

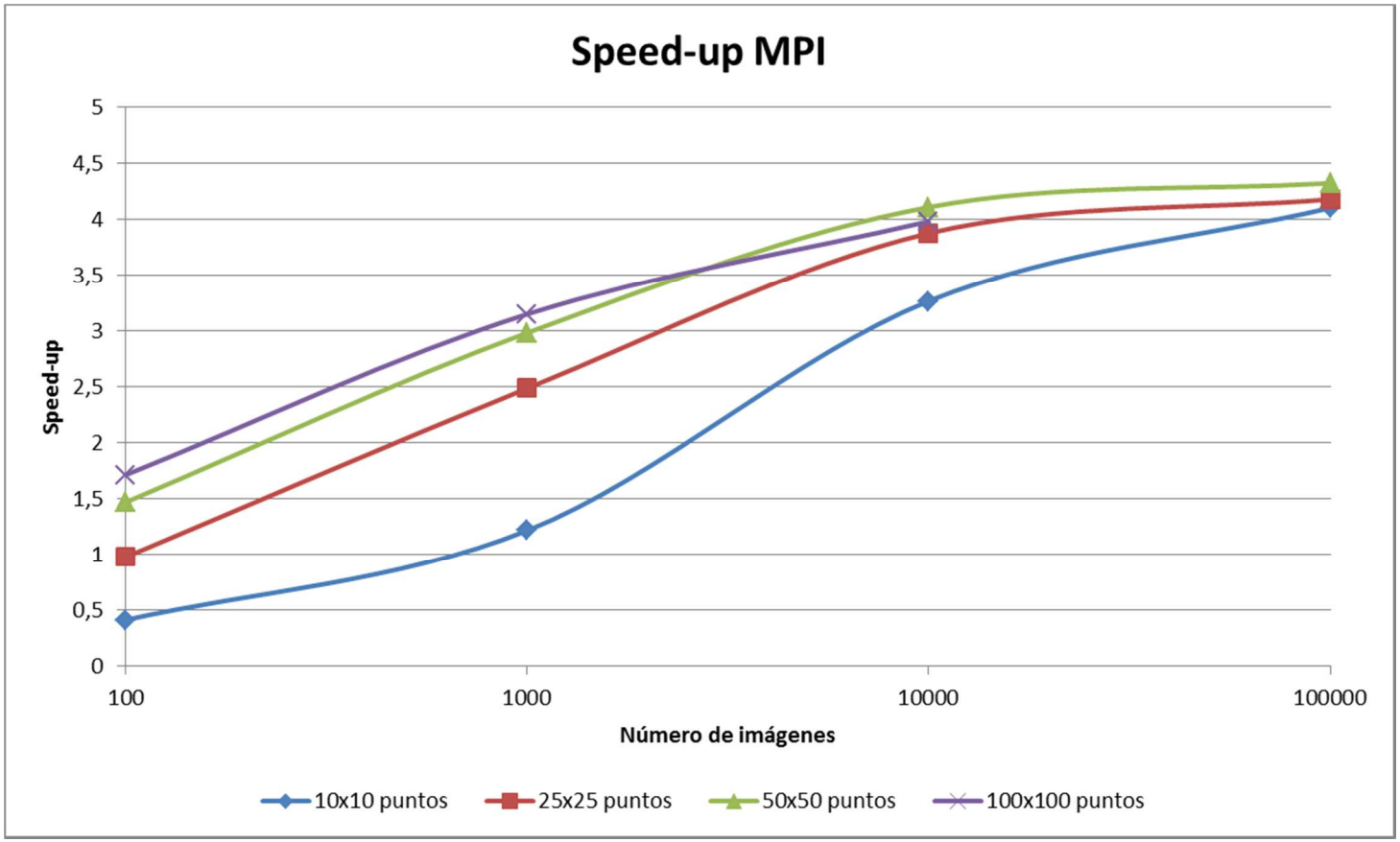


Estudio del paralelismo.

- Misma idea que en OpenMP, pero realizando una asignación estática del trabajo realizado.
- Utilizamos la distribución MPICH2.
- Importante especificar el compilador de C++ a utilizar (-CXX=icpc).
- Los resultados obtenidos son similares, con speed-up ligeramente inferiores a los obtenidos en OpenMP.

Estudio del paralelismo.

- Ejecución con 4 cores:

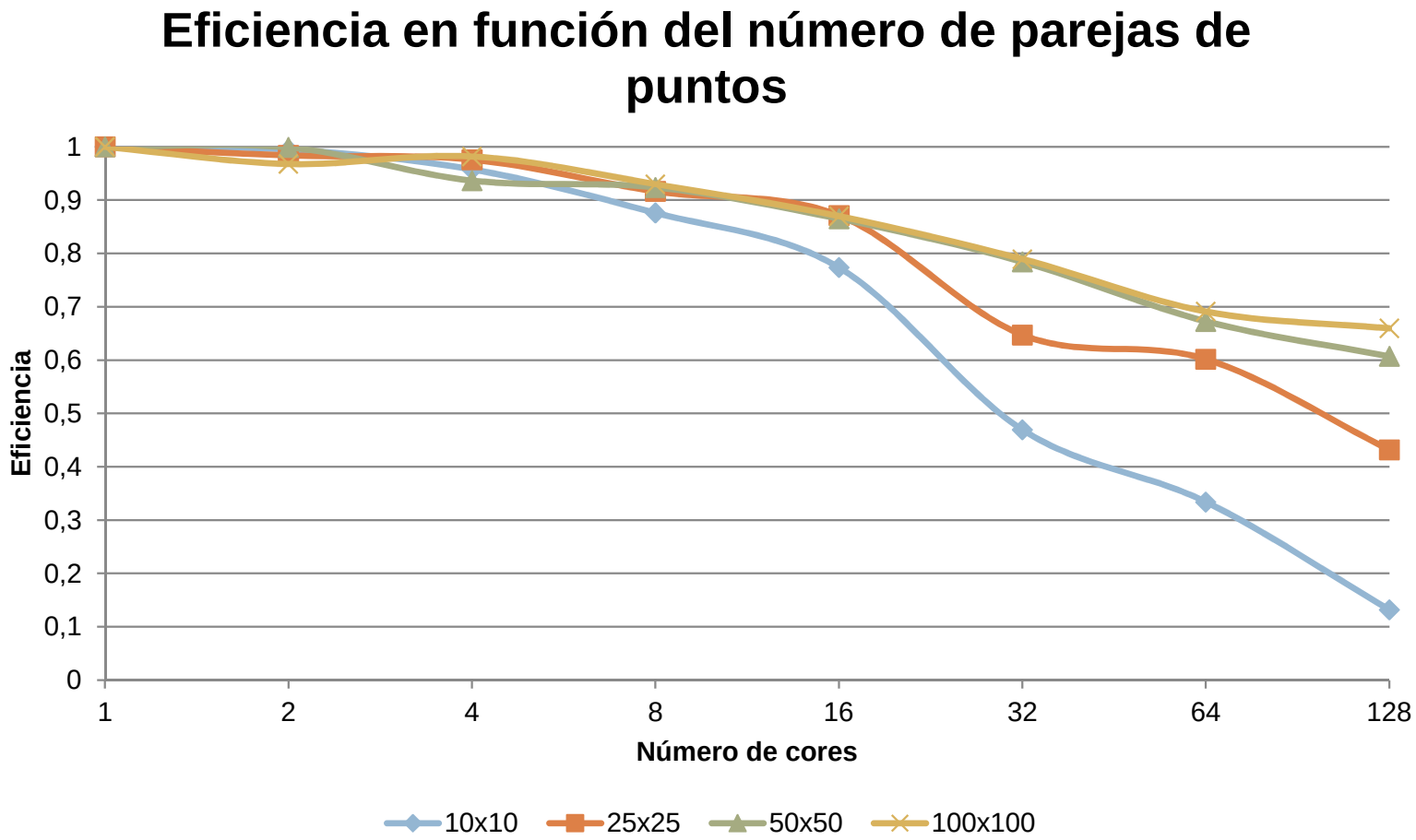


Estudio del paralelismo.

- Al igual que para la rutina en OMP, comprobaremos la escalabilidad de esta rutina haciendo uso de Arabí.
- En este caso:
  - En función del número de parejas de puntos, fijamos el número de imágenes a 100000.
  - En función del número de imágenes a evaluar, fijamos el número de puntos a 100x100.

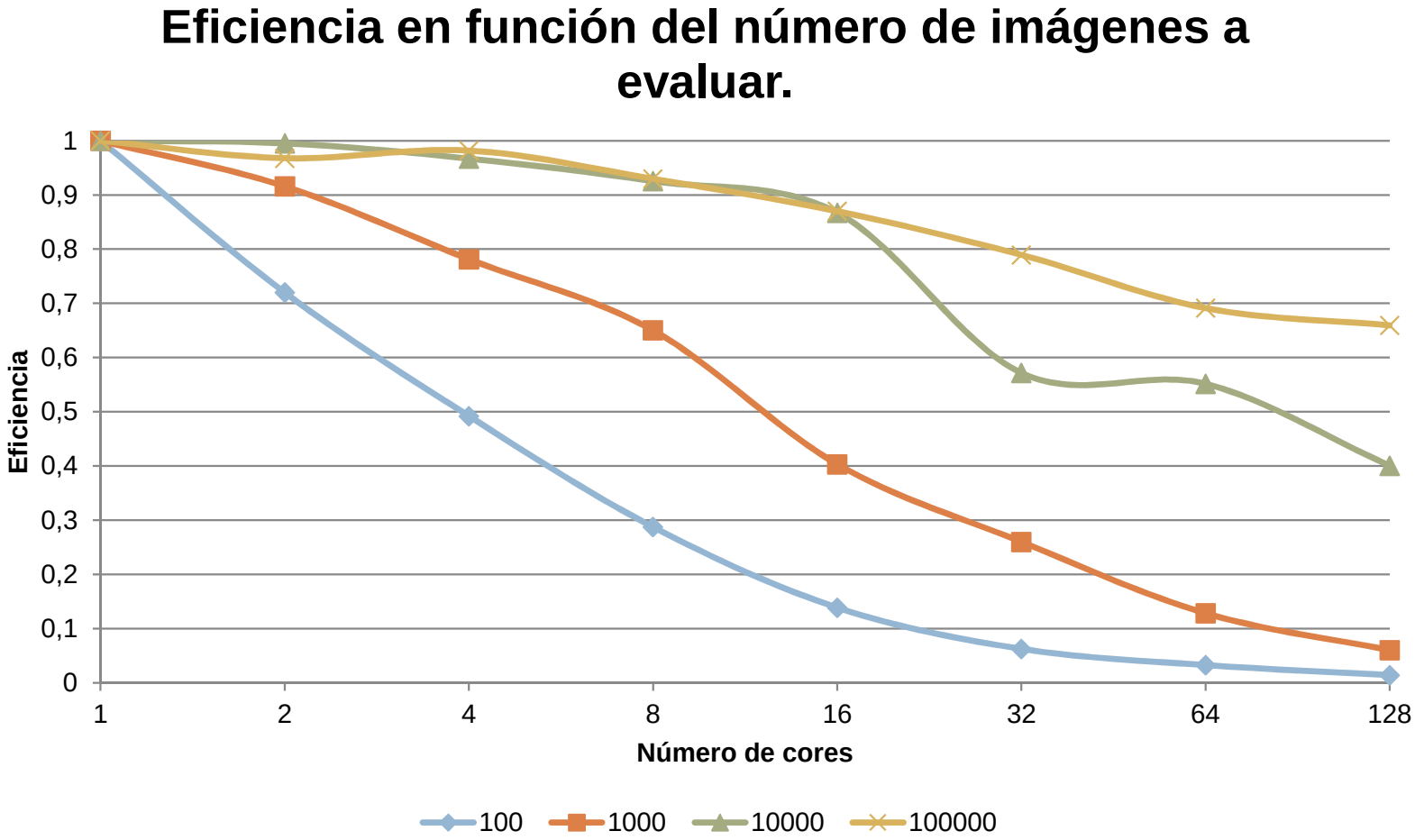
Estudio del paralelismo.

- Resultados en Arabí:



Estudio del paralelismo.

- Resultados en Arabí:

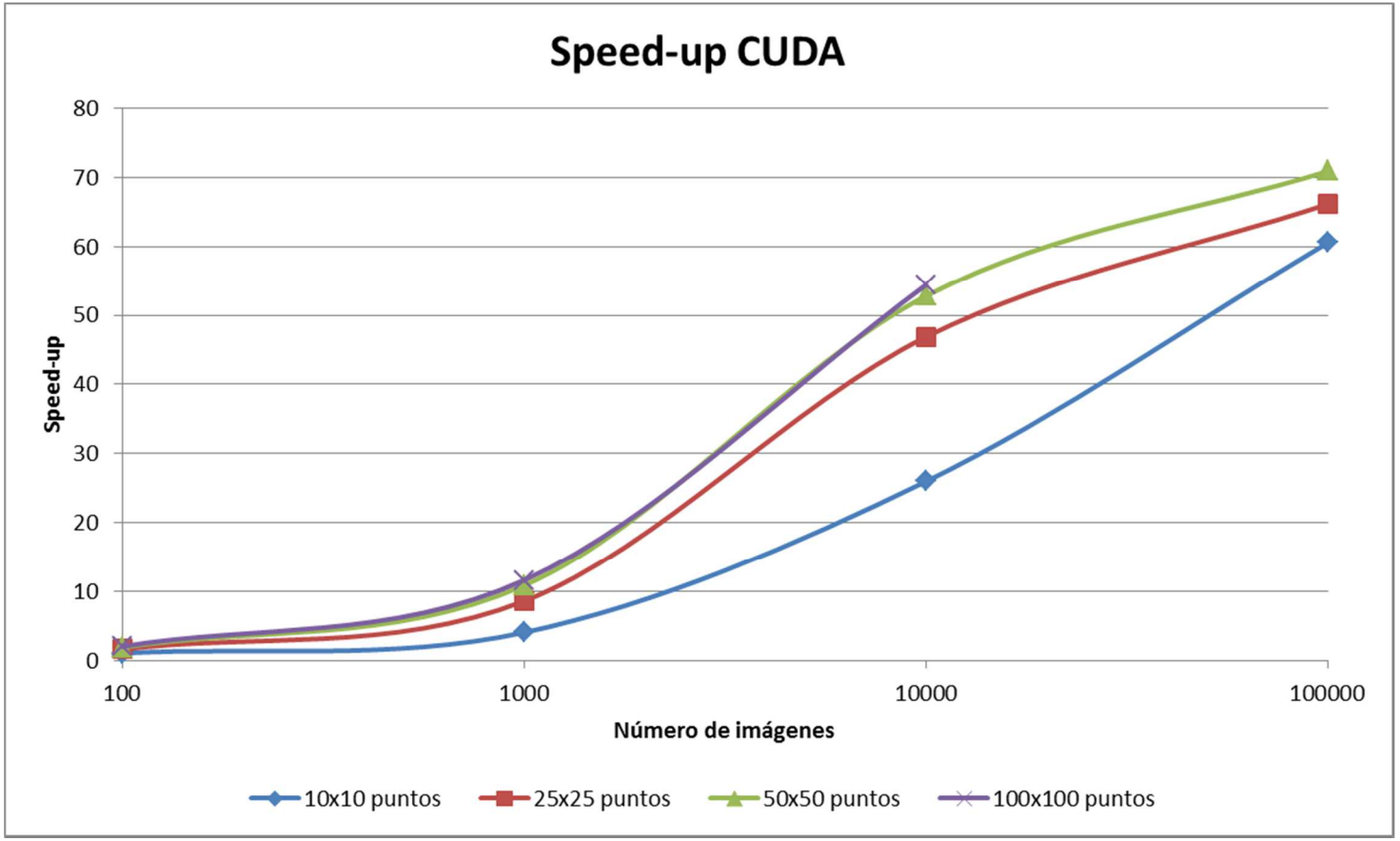


Estudio del paralelismo.

- Misma idea para CUDA, poniendo un thread por cada imagen a evaluar.
- Utilizar un grid bidimensional.
- Los resultados parciales los almacenamos en una matriz temporal. Para la reducción de los datos podríamos utilizar un kernel de reducción en árbol.

Estudio del paralelismo.

- En una GPU con 112 cores:



Estudio del paralelismo.

- En resumen:

imágenes	puntos	Fortran (ifort)	C++ (icpc)	OMP	CUDA	MPI (mpich2)
100	10x10	0,148	0,156	0,054	0,155	0,392
	25x25	0,689	0,727	0,228	0,453	0,745
	50x50	2,584	2,823	0,754	1,476	1,882
	100x100	9,986	10,972	2,847	5,463	6,386
1000	10x10	0,859	0,726	0,235	0,167	0,554
	25x25	5,018	4,202	1,128	0,474	1,637
	50x50	19,659	16,688	4,267	1,492	5,434
	100x100	78,71	67,411	16,944	5,636	20,789
10000	10x10	7,432	6,081	1,406	0,223	1,773
	25x25	46,318	37,605	8,517	0,78	9,431
	50x50	185,205	151,638	33,816	2,737	35,167
	100x100	760,954	607,812	135,392	10,59	144,722
100000	10x10	74,676	59,391	13,406	0,94	13,882
	25x25	459,856	370,951	83,526	5,361	84,873
	50x50	1822,058	1495,109	335,502	21,022	344,977

Trabajos futuros.

- Finalizar el estudio de las funciones de Green bidimensionales y tridimensionales.
- Evaluar las rutinas desarrolladas en otro tipo de sistemas.
- Diseñar y evaluar rutinas híbridas que aprovechen las capacidades de distintos sistemas.

Trabajos futuros.

- Continuar explorando las posibilidades del uso de dispositivos gráficos para este tipo de problemas.
  - CUDA Compute Capability 2.0.
  - Entornos multi GPU.
- Desarrollar una librería de rutinas de cálculo de funciones de Green.
  - Diversos tipos de funciones.
  - Mecanismos de autooptimización transparentes al usuario para diferentes arquitecturas y sistemas.

¿Preguntas?