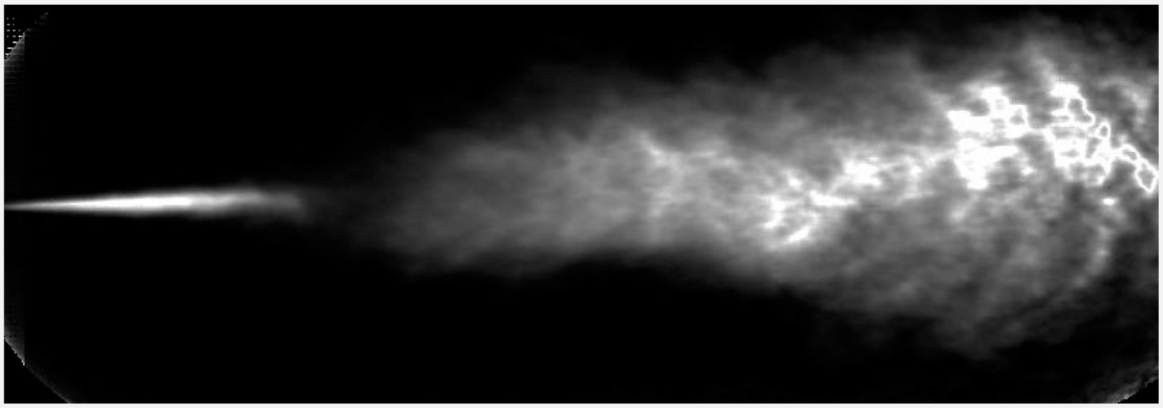
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

motores térmicos

Escuela Técnica Superior de Ingeniería del Diseño

USO DE HERRAMIENTAS DE PARALELIZADO PARA LA OPTIMIZACIÓN DE ALGORITMOS DE PROCESADO DE IMÁGENES EN CHORRO DIÉSEL

Septiembre 2017



Autor: Omar Diab Pascual

Tutor: José Vicente Pastor Soriano

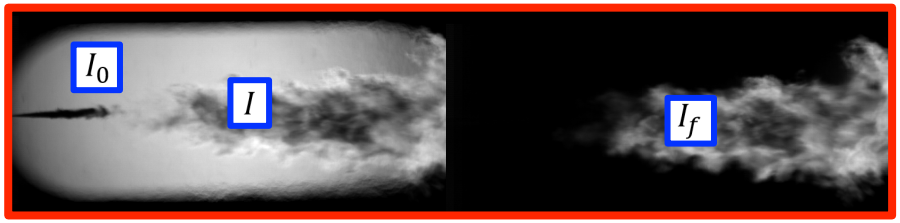
Objetivos

Evaluar el potencial de la paralelización en algoritmos para el procesado de imágenes procedentes de la combustión en chorros Diésel.

- Análisis de las mejoras: ***tiempo de cálculo*** y *coste computacional*
- Paralelización del código base en la CPU ➔ Nivel 1
- Paralelización del código base en la GPU ➔ Nivel 2
- Comparación código base vs código alfa. Ventajas e inconvenientes
- Elección del método de paralelizado óptimo

CONTEXTUALIZACIÓN

PROCESO FÍSICO



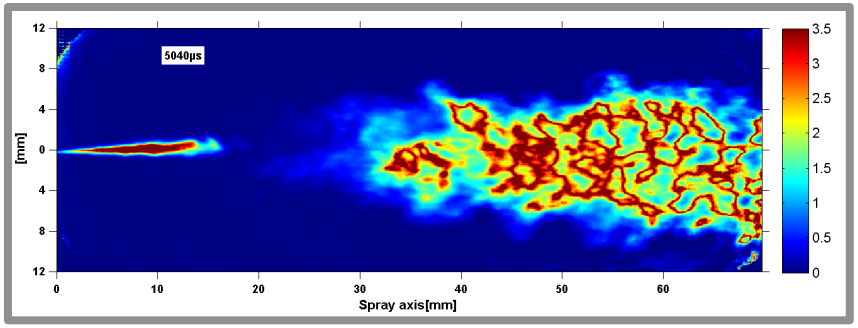
Extinción

$$\frac{I - I_f}{I_0} = \exp(-KL)$$

"Optical thickness"

PROCESADO

1) Promediado imágenes fondo
2) Restado fondo a luminosas
3) Obtención mapas de concentración y distribución
4) Promediado de los mapas

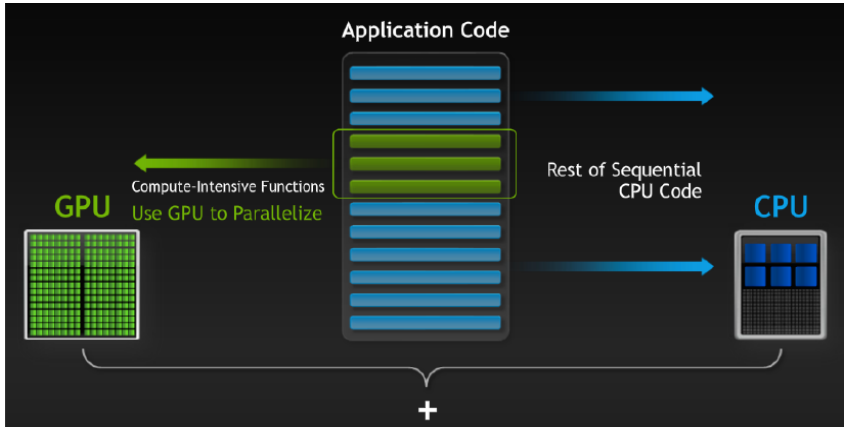


MATERIALES Y MÉTODOS

¿Por qué paralelizar un algoritmo? ¿Qué herramientas se necesitan?

- Los algoritmos que procesan imágenes son complejos (⇧nº líneas de código)
- Ensayo típico: 250 imágenes/rep. 30 rep./caso ➔ *7500 imágs/caso (≈5 GB)*
- Realizan tareas adicionales (segmentaciones, líneas de control, medidas)

Repercusión sobre el tiempo de cálculo por caso de ensayo

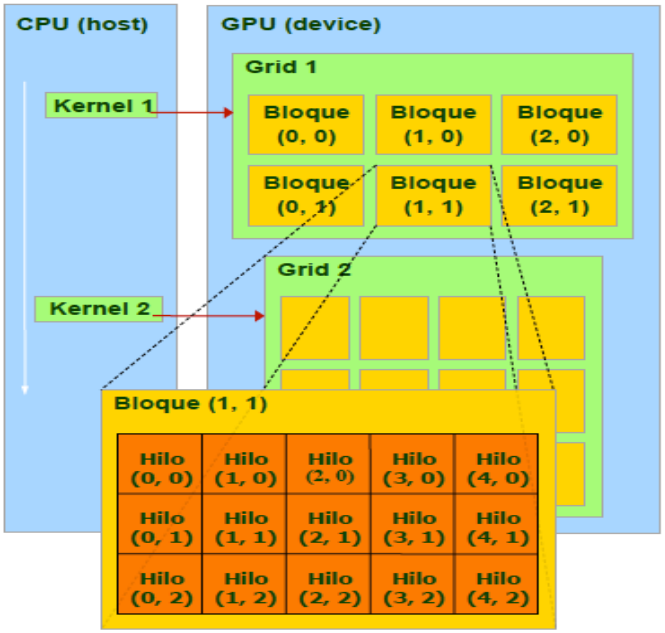


PC convencional (2, 4, 6 u 8 núcleos), los cálculos se pueden repartir...

¡¡Una NVidia GeForce GTX 960 contiene 1024 capaces de trabajar de manera simultánea!!

Arquitectura CUDA (*Compute Unified Device Architecture*)

La fracción de código a ejecutar en la GPU se denomina **Kernel** y se invoca desde la CPU creando una serie de hilos que lo ejecutan de individual e independiente.



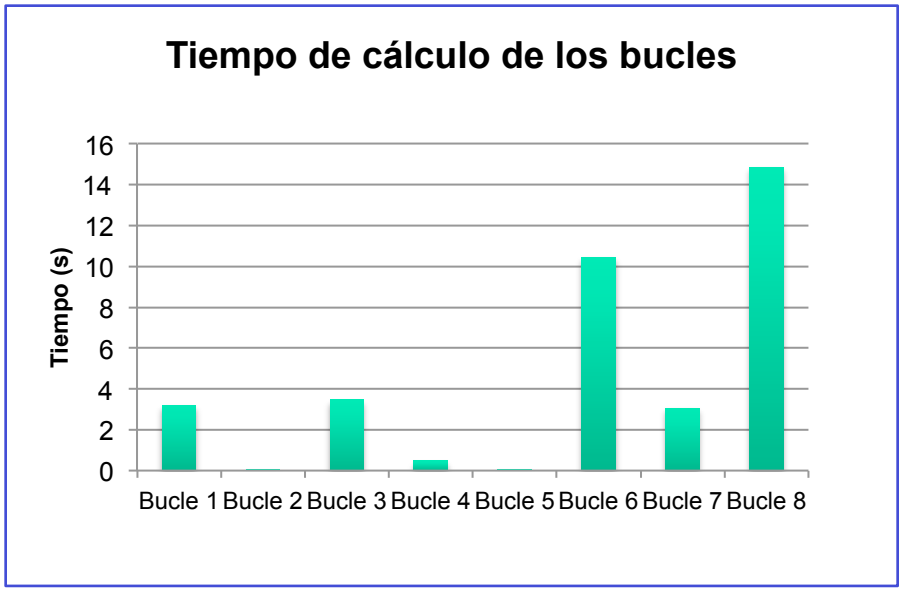
- Todos los hilos tienen una memoria compartida

MatLab gestiona todo esto internamente con su *Parallel Computing Toolbox*

METODOLOGÍA EXPERIMENTAL

1) Estudio previo para identificar los cuellos de botella
2) Paralelizado CPU/GPU
3) Realización análisis y estudios parámetricos

(El trabajo se desarrolló haciendo uso del código base con *5 repeticiones* por caso)



Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
Codigo_base_puro	1	37.235 s	15.821 s	
imshow	246	12.900 s	4.377 s	
GetMovie	10	6.175 s	0.597 s	
imreadmraw	10	5.454 s	5.454 s	
initSize	246	3.096 s	0.163 s	
movegui	246	2.479 s	2.442 s	
newplot	492	2.427 s	0.463 s	

- Atacar a los 8 bucles que consumen el 97,5 %
- Buscar la mejora de las tres funciones “top”

2) Paralelizado CPU/GPU

Paralelización CPU

- Estudio previo de los recursos
- Sustitución *for* ➔ *parfor*
- Gestión autónoma (*Parallel Pool*)
- Problemas de implementación
- OK para Bucles 1,2,6 y 7 (cinco *parfor*)

Paralelización GPU

- Estudio previo de los recursos
- *Modificación de muchas líneas*
- Ayuda de *Parallel Computing Toolbox*
- Problemas sólo con la memoria video
- OK para Bucles 1, 3, 6, 7 y 8

```
parfor i=1:size(Iback,2)*size(Iback,1)
                if ~isempty(Iback{i})
                    Ibacksum=Ibacksum+double(Iback{i}(:,:));
                    Iback_count=Iback_count+1;
                end
            end
```

```
for r=1:length(Rep2Process)
                n=Rep2Process(r);
                data =
GetMovie_paralelizada(0,n,'Photron','mraw',NumPerRep,Framerate);
                data=gather(data);
```

METODOLOGÍA EXPERIMENTAL

3) Realización análisis y estudios parámetricos

1) Sensibilidad al número de repeticiones

- Tiempos totales y parciales variando de 5 a 30 nº repeticiones
- 40 tomas de muestra (asegurar significatividad estadística)
- Elaboración de 6 tablas de datos

2) Sinergias en el paralelizado de bucles internos

- Estudiar si su ejecución es más rápida aislada o en continuo
- 40 tomas de muestra (asegurar significatividad estadística)
- Elaboración de 2 tablas de datos (continuo y aislado)

3) Sensibilidad al tipo de paralelizado

- Dilucidar cuál de las dos vías CPU/GPU es la más exitosa
- Comparación en términos de tiempo de cálculo y de coste

Paralelización CPU

	SP 5 Repeticiones	P 5 Repeticiones	SP 10 Repeticiones	SP 10 Repeticiones	SP 15 Repeticiones	SP 15 Repeticiones
Tiempo (s)	36,453	81,313	54,449	275,221	72,459	663,34
Desv. St.	0,318	2,54	1,217	34,818	1,049	42,25
Mejora(%)	-123,065		-405,281		-815,469	

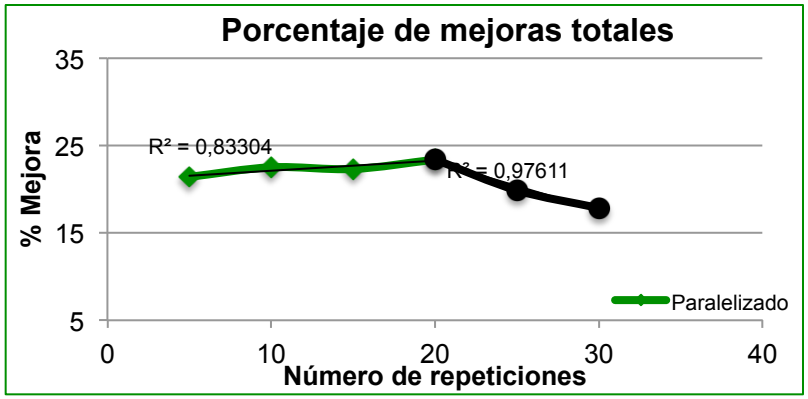
- No se pudo concluir la tanda de ensayos (error por falta de memoria)
- Posible explicación: - El procesador no gestiona bien el reparto de tareas (Bucle 7)

Consecuencia ➔ Se abandona el estudio de sensibilidad al tipo de paralelizado

Conclusión: - **Paralelizado inviable, tanto en términos lógicos como prácticos**.

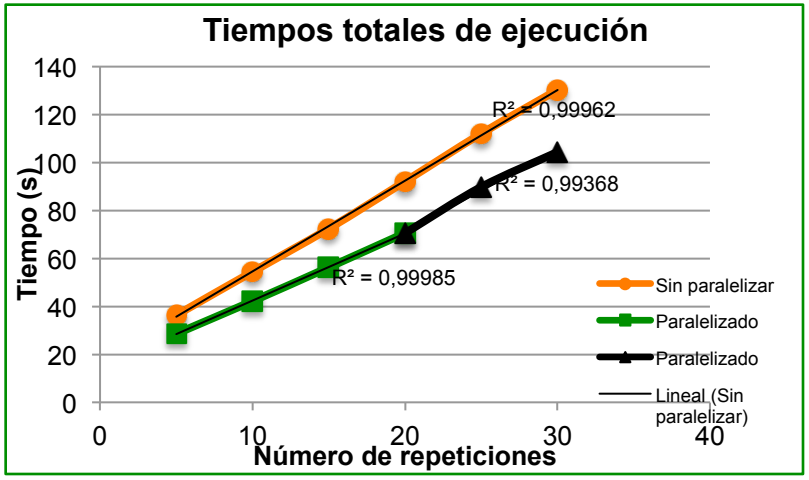
SENSIBILIDAD AL NÚMERO DE REPETICIONES

Paralelización GPU



- Mejoras entre un 18 y un 23 %
- Hasta las 20 repeticiones: mejoras > 21%
- Para 25 y 30 repeticiones: mejoras < 20 %
- Bucle 8 es el responsable del deslucimiento

Conclusión: - **El paralelizado mejora significativamente el tiempo de cálculo** (Bucle 6)

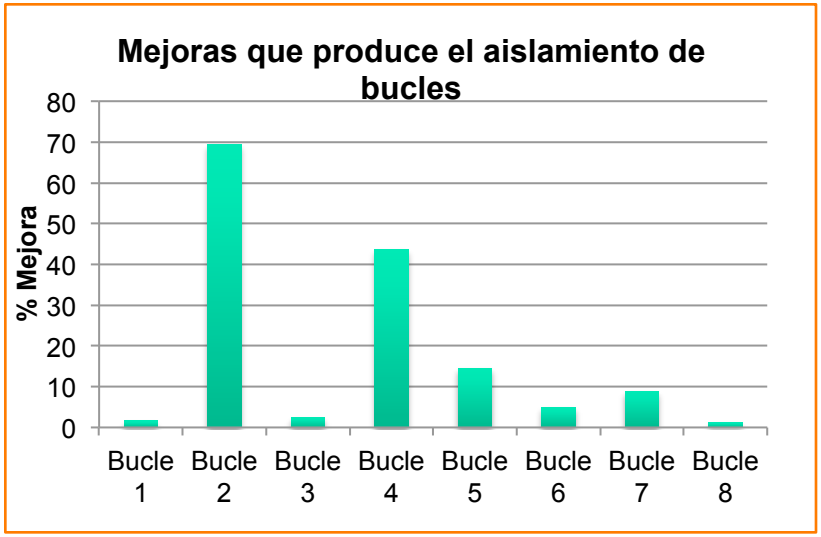


- Dos tramos (por la decaída de la mejora)
- Fuerte correlación lineal (no curvatura)
- La divergencia: efecto aditivo del % mejora

Conclusión: - **No afecta significativamente al paralelizado (ni mejora ni empeora)**

SINERGIAS EN EL PARALELIZADO DE BUCLES INTERNOS

Paralelizado					
	Tiempo en continuo (s)	Desv. St.	Tiempo en aislado (s)	Desv. St.	Porcentaje variación
Bucle 1	2,879	0,006	2,832	0,026	1,617
Bucle 2	0,009	0,000	0,003	0,0002	69,543
Bucle 3	3,31	0,036	3,228	0,02	2,468
Bucle 4	0,461	0,004	0,259	0,04	43,782
Bucle 5	0,016	0,001	0,014	0,006	14,539
Bucle 6	6,149	0,068	5,848	0,18	4,905
Bucle 7	2,135	0,010	1,945	0,16	8,912
Bucle 8	12,98	0,011	12,813	0,269	1,293
Total	**27,94**	**0,135**	**26,942**	**0,701**	**3,575**



- La no migración de líneas podría explicar el orden
- Bucles paralelizados: Poca afección
- Posible creación de elementos virtuales efímeros

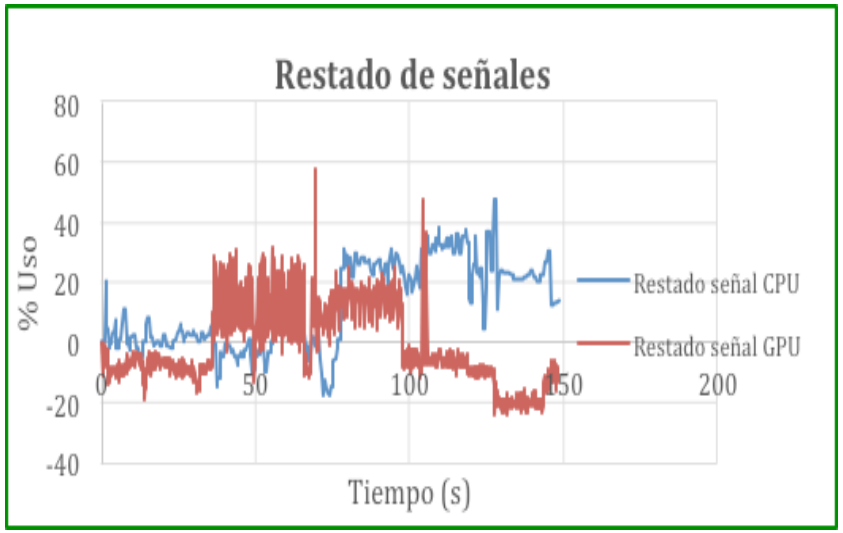
Conclusiones:

1) Hipotéticamente se obtendría menos de 3 % mejora

2) Los bucles paralelizados no empeoran significativamente su ejecución en continuo (búffers)

RESULTADOS. COSTE COMPUTACIONAL

Se procedió al restado de señales para estudiar variaciones entre los código base y paralelizado (código alfa)



Señal código base – Señal código alfa (CPU)
Señal código alfa – Señal código base (GPU)

- La señal en CPU encierra mayor área positiva ➔ El código base lo emplea más (especialmente en el último tercio, que se asocia a la ejecución de los Bucles 6, 7 y 8)
- La señal en GPU encierra mayor área positiva ➔ El código alfa la emplea más

Conclusión: - **Los recursos que consume el ordenador con el paralelizado son menores**

6. CONCLUSIONES

- Las herramientas de paralelización **han conseguido optimizar el algoritmo.**

  - Las mejoras reportadas por **vía GPU supera el 20 % en términos de reducción de tiempo de cálculo** (horas e incluso días si se procesan muchos casos).

  - La paralelización **vía CPU no es aplicable** para este algoritmo.

  - No hay significatividad en el incremento de la mejora con el aumento hasta las 20 repeticiones, observándose un empeoramiento al seguir aumentando por el colapso de buffer que se produce.

  - Los bucles paralelizados han demostrado un alto nivel de aprovechamiento de los recursos al no mostrar empeoramiento significativo entre su ejecución en continuo o aislado.

  - En coste computacional, se ahorra en RAM, memoria física y CPU con el paralelizado, lo que puede permitir la **ejecución de dos aplicaciones simultáneamente.**

Presupuesto total del trabajo

PRESUPUESTO TOTAL	
Concepto	**Total [€]**
Mano de obra	11531
Materiales	363,91
Costes indirectos (10% sobre c.directos)	1153,1
Costes de beneficio industrial (20% sobre c.directos)	576,55
IVA (21%)	2784,74
Coste total	**16409,3**

MUCHAS GRACIAS POR SU ATENCIÓN

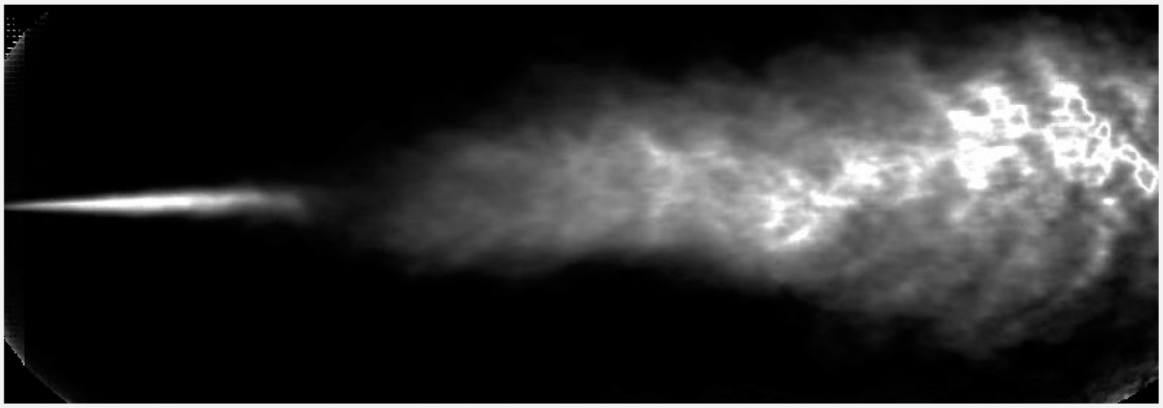
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

motores térmicos

Escuela Técnica Superior de Ingeniería del Diseño

USO DE HERRAMIENTAS DE PARALELIZADO PARA LA OPTIMIZACIÓN DE ALGORITMOS DE PROCESADO DE IMÁGENES EN CHORRO DIÉSEL

Septiembre 2017



Autor: Omar Diab Pascual

Tutor: José Vicente Pastor Soriano