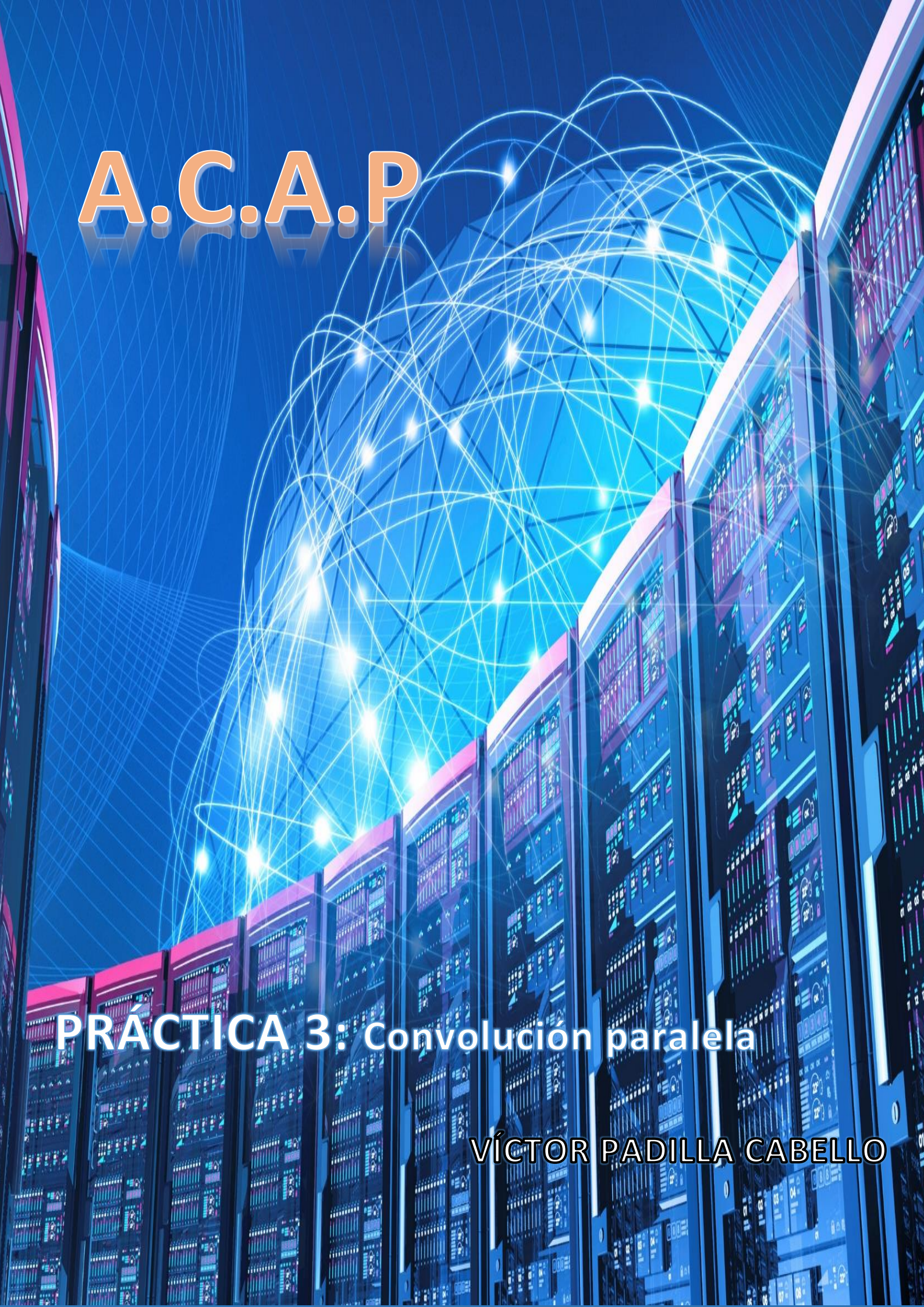


A.C.A.P

PRÁCTICA 3: Convolución paralela

VÍCTOR PADILLA CABELLO

Índice

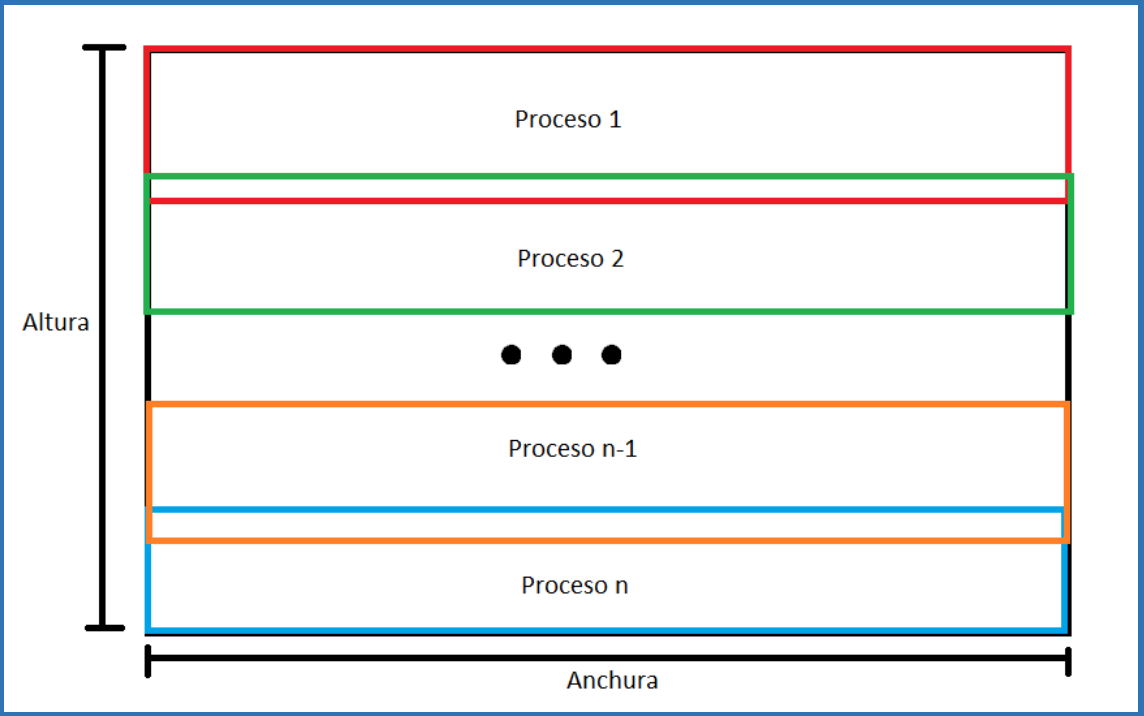
• Cuestiones:

1. Hoja de Cálculo con el Tamaño de las Imágenes:

1_in.png (bosque)	1155072 pixels (1504 x 768)
2_in.png (montañas)	8294400 pixels (3840 x 2160)
3_in.png (mapamundi)	14580000 pixels (5400 x 2700)

2. Reparto de la Carga de Trabajo

Para el reparto de la carga de trabajo, me he limitado en dividir la imagen en trozos iguales para cada proceso. La única complejidad reside en que cada proceso necesita su fila de inicio − 1 y su fila de fin + 1 para realizar correctamente la convolución. A continuación, un diagrama que refleje lo explicado:



3. Tiempos de ejecución obtenidos:

Para mejorar los tiempos en la segunda versión del programa simplemente he realizado un desenrollado de bucle en *"for"* más interno de la parte de computación.

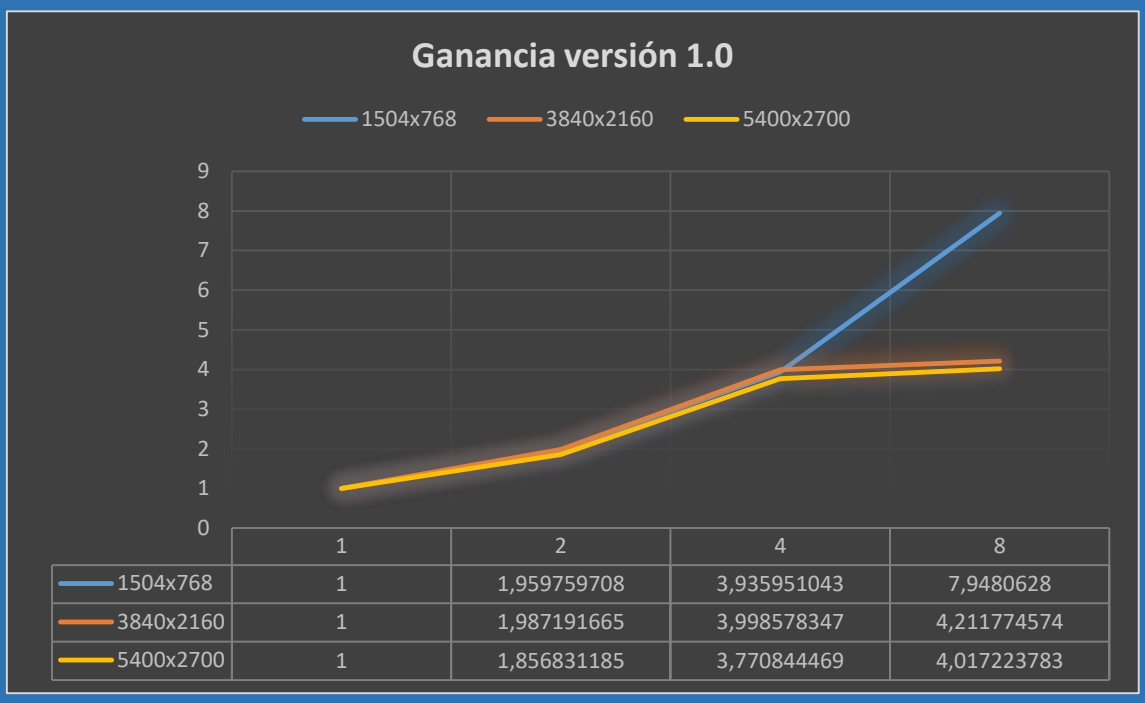
Los tiempos obtenidos se pueden encontrar en los archivos convolucion_mpi.o* y convolucion_2.0.o*.

Convolucion 1.0			
Tiempos de ejecución obtenidos (en Seg)			
Nº de procesos	1504x768	3840x2160	5400x2700
1	0,180731	1,355686	2,172836
2	0,092221	0,682212	1,170185
4	0,045918	0,339042	0,57622
8	0,020713	0,32188	0,54088

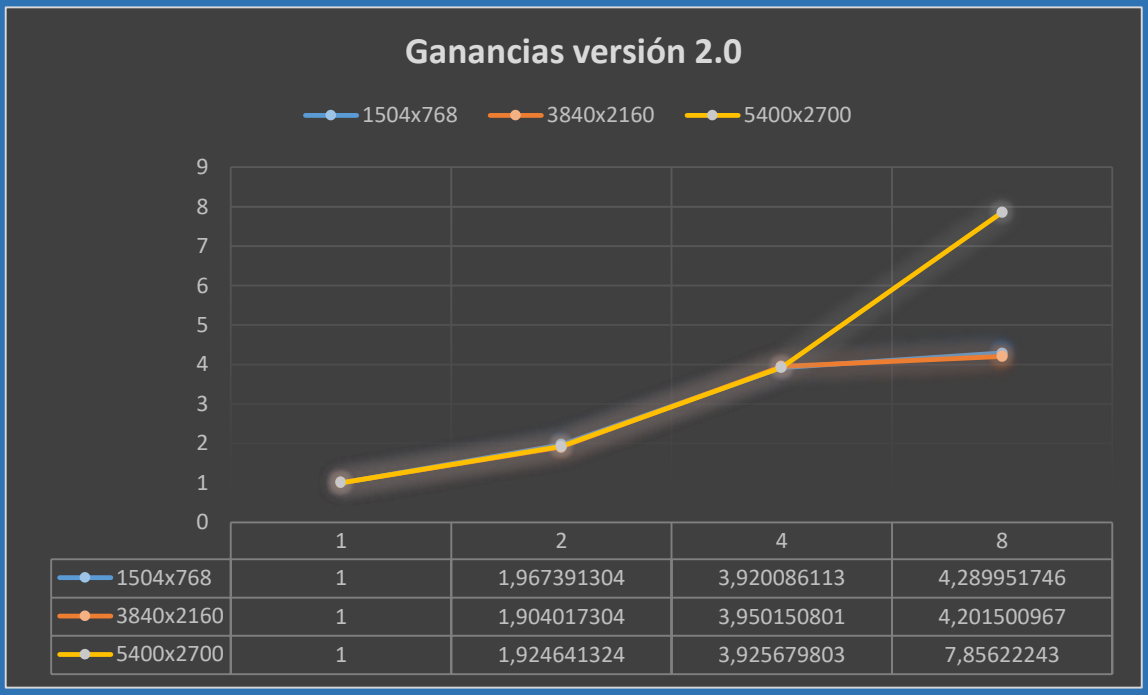
Convolucion 2.0			
Tiempos de ejecución obtenidos (en Seg)			
Nº de procesos	1504x768	3840x2160	5400x2700
1	0,151135	1,138153	1,921687
2	0,07682	0,597764	0,998465
4	0,038554	0,288129	0,489517
8	0,03523	0,270892	0,244607

4. Análisis de la Ganancia:

Se muestra unas gráficas donde se aprecia la ganancia con incrementales procesos:



	1	2	4	8
1504x768	1	1,959759708	3,935951043	7,9480628
3840x2160	1	1,987191665	3,998578347	4,211774574
5400x2700	1	1,856831185	3,770844469	4,017223783



	1	2	4	8
1504x768	1	1,967391304	3,920086113	4,289951746
3840x2160	1	1,904017304	3,950150801	4,201500967
5400x2700	1	1,924641324	3,925679803	7,85622243

5. Conclusiones:

Las conclusiones que podemos adherir a esta experimentación es la misma que cualquiera de las que incluya programación en paralelo.

La programación paralela puede acelerar nuestro programa de una forma considerable si este lo permite en su lógica. Pero siempre debemos de tener en cuenta que es tipo de programación añade tiempos de sobrecarga que, si el tamaño del problema no es lo suficientemente grande, puede ser contraproducente. Así pues, hemos observado que los tiempos de ejecución se reducen en una cantidad lo suficientemente grande para considerar usar programación paralela, pero a veces la ganancia puede hacer cosas extrañas y no respetar lo que uno espera que resultaría.