Análisis de la Paralelización de un Contador de Objetos en Imágenes Binarias

Pablo Novarini[1], Luciano Iglesias[2], Andrés Barbieri[3], Juan Cortabitarte[4]

Laboratorio de Investigación y Desarrollo en Informática[5]

Facultad de Informática

Universidad Nacional de La Plata

Resumen

Se presenta el análisis e implementación de un algoritmo de conteo de objetos en imágenes binarias, realizado sobre una arquitectura multiprocesador con 32 nodos, trabajo realizado dentro de la asignatura Procesamiento Paralelo de la Licenciatura en Informática de la UNLP.

Se estudia el speed-up alcanzable en función de diversas características de las imágenes de entrada.

Se analiza el ajuste del algoritmo a la configuración de la arquitectura elegida (en el cubo multiprocesador utilizado los links físicos son reprogramables) y se discute la portabilidad de la aplicación a otras arquitecturas multiprocesador.

Por último se presentan los resultados obtenidos según determinadas características de la imagen de entrada (por ejemplo histograma), tratando de establecer la dependencia del tiempo de ejecución de cada uno de estos parámetros.

Palabras claves: Procesamiento Paralelo – Procesamiento de Imágenes – Arquitecturas Paralelas – Labelling -

[1] Ayudante Alumno. Becario Alumno LIDI pnovarini@lidi.info.unlp.edu.ar
[2] Becario Alumno LIDI lucianoiglesias@flashmail.com
[3] Ayudante Alumno. Becario Alumno LIDI barbieri@lidi.info.unlp.edu.ar
[4] Ayudante Alumno jcorta@casaresnet.com.ar
[5] LIDI. Calle 50 y 155 1er. Piso (1900) La Plata, Bs.As. Tel/fax: +54 221 422 7707
WEB:www-lidi.info.unlp.edu.ar

Introducción

Definimos *procesamiento paralelo* como la ejecución concurrente (en el mismo instante de tiempo) sobre distintos componentes físicos (procesadores) [Tin98]. Los objetivos del procesamiento paralelo son disminuir los tiempos de ejecución, incrementar la eficiencia y atender tareas de control y procesamiento en tiempo real [Tin98].

El procesamiento paralelo resulta necesario en problemas con gran cantidad de cálculo y cuando los tiempos de respuesta requeridos no pueden ser alcanzados mediante el procesamiento secuencial [Lei92].

Vale la pena aclarar que hay problemas cuya naturaleza permite obtener una solución paralela, y otros que no.

Una *computadora paralela* consiste de dos o más procesadores independientes con capacidad de comunicarse entre ellos.

El número de procesadores, sus características, la topología de interconexión, los mecanismos de acceso a la memoria y los protocolos de comunicación son los aspectos básicos que definen el comportamiento de una arquitectura paralela.

Para poder llevar a cabo la implementación de un algoritmo sobre una arquitectura de este tipo se requiere que el problema admita una descomposición en múltiples procesos que se comuniquen y sincronicen. Esto involucra encontrar una forma conveniente de distribuir las acciones para utilizar al máximo los recursos disponibles. Es importante tener en cuenta que la implementación de un algoritmo paralelo está fuertemente ligado a la arquitectura empleada.

Un concepto importante a tener en cuenta para el análisis de los resultados obtenidos es el de speed-up [Tin98]. La relación entre el tiempo de ejecución de un algoritmo sobre un procesador (T_1) y el tiempo de ejecución sobre una arquitectura paralela con N procesadores (T_N) se denomina factor de speed-up.

En el campo del *procesamiento digital de imágenes*, la cantidad de información que se procesa es de gran magnitud. Desde los algoritmos más simples, como aplicar un filtro, hasta los más complejos, como el reconocimiento de patrones en tiempo real o la segmentación e identificación de objetos dentro de una imagen, requieren mucho procesamiento debido a la gran cantidad de información [Gon92].

A partir de lo anterior surge inmediatamente la idea de dividir la “carga” para ser tratada por más de un procesador. La paralelización de algoritmos de tratamiento de imágenes es un campo importante para el estudio dada la complejidad de procesamiento que generalmente se requiere.

Idea del trabajo

El objetivo de este trabajo es hacer un análisis de los resultados obtenidos de la implementación de un algoritmo paralelo que cuenta la cantidad de objetos de una imagen binaria.

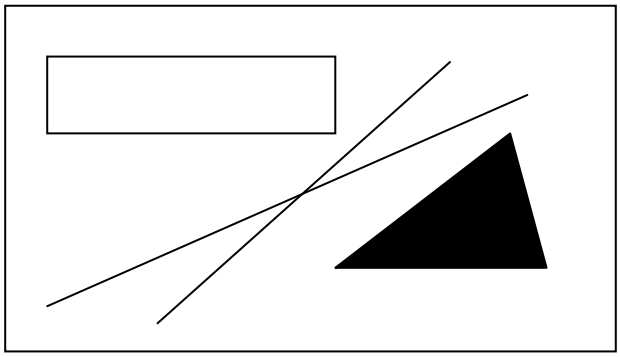
Se realizan dos implementaciones, una de ellas sobre una arquitectura paralela de 32 procesadores, y la otra sobre un sólo procesador de las mismas características; con el fin de realizar comparaciones en cuanto al rendimiento de los algoritmos con imágenes con diferente distribución del histograma de color, con dfirente número de objetos a identificar y con formas y tamaños (relativos a la imagen total) distintos de los objetos en cuestión.

El parámetro principal de estudio ha sido la función de speed-up alcanzable.

Análisis del problema

El problema consiste en contar los objetos en una imagen binaria. Los pixeles que componen una imagen binaria tienen dos valores posibles (blanco o negro). Se entiende por objeto a un conjunto de pixeles de color negro conectados entre sí.

A continuación se muestra un ejemplo de una imagen binaria con tres objetos (las líneas cruzadas, el rectángulo y el triángulo relleno):



La resolución de este problema se puede dividir en dos etapas. La primera consiste en encontrar la forma de identificar dentro de una imagen el conjunto de pixeles que conforman un mismo objeto (algoritmo de labeling).

Una vez solucionado esto, se debe resolver la forma de paralelizarlo. A primera vista, surge la idea de dividir la imagen en diferentes partes, y realizar simultáneamente la búsqueda de objetos en las diferentes partes de la imagen.

La consecuencia obvia de esta solución es que un objeto puede quedar en más de una parte, de manera que la cantidad de objetos total de la imagen no es la suma de las cantidades encontradas en cada porción de ella (“*el todo no es mayor que la suma de las partes”*). Se necesita entonces determinar si objetos detectados en diferentes partes de la imagen en realidad conforman un mismo objeto en la imagen total.

Arquitectura de hard y algoritmo utilizado

Para resolver el problema se utilizó un hipercubo de 32 transputers. Cada transputer posee cuatro links físicos que lo conectan con otros nodos del multiprocesador [Tra90]. Los links físicos de los transputers son reconfigurables, lo que permite trabajar con diferentes topologías.

Se realizó la implementación en el lenguaje de programación C utilizando una librería que permite establecer una comunicación sincrónica entre nodos conectados del cubo multiprocesador [Tra94].

Como se desea analizar el speed-up que se puede alcanzar utilizando un algoritmo paralelo para resolver el problema, se realizó también una implementación utilizando un solo procesador del hipercubo. De esta forma las mediciones de los tiempos de ejecución de ambas implementaciones son comparables ya que utilizan procesadores del mismo tipo.

Para la implementación paralela, debido a que se dispone de 32 procesadores, se divide la imagen en 32 fragmentos horizontales. Cada uno se asigna a un procesador para que etiquete (labeling) los objetos encontrados. Como partes de un objeto pueden quedar en varios fragmentos, deben analizarse los bordes de fragmentos adyacentes de la imagen para determinar si hay objetos partidos.

La imagen se divide horizontalmente para realizar de forma sencilla el análisis de los bordes, entre dos fragmentos adyacentes de la imagen particionada. De esta forma cada fragmento tiene solamente dos bordes a tratar, el superior y el inferior. Notar que si la imagen se dividiera en forma de cuadrícula, por ejemplo, cada fragmento de la imagen tendría más de dos adyacentes, aumentando la complejidad algorítmica.

Es evidente que dos procesadores que tratan fragmentos adyacentes de la imagen, deberían tener la posibilidad de comunicarse directamente, para que uno de ellos analice los bordes.

Se elige la configuración de la arquitectura que se muestra en la siguiente figura, la cual permite distribuir los datos de manera conveniente entre los procesadores.

La distribución de la imagen en los procesadores se asocia con el número identificatorio de cada transputer (esto se logra modificando la topología de conexionado).

De esta forma se facilita el análisis de los bordes. Al estar la imagen distribuida así, dos procesadores con regiones de imagen consecutivas son siempre vecinos.

Gráfico de la imagen partida

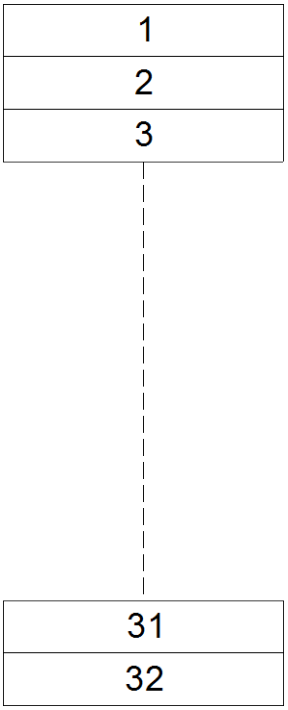
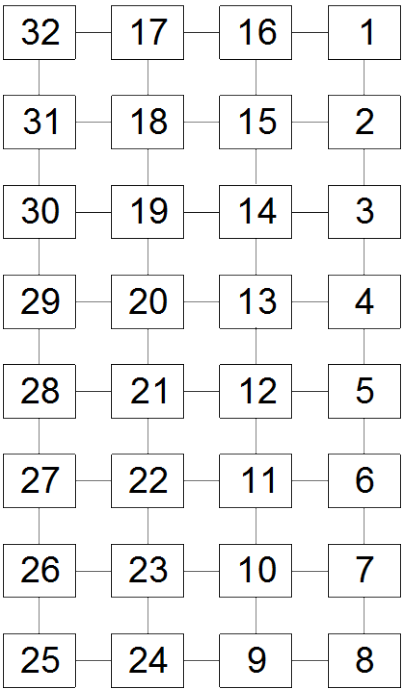


Gráfico de la red de transputers



Una vez realizado el labeling, cada procesador se comunica con los vecinos que tienen una parte de la imagen adyacente a la suya, para realizar el análisis de los bordes; por ejemplo, el transputer que tiene la parte 9 se comunica con los que tienen la parte 8 y 10.
Por último, se reúnen todos los resultados parciales para obtener el resultado buscado.

En la implementación con un solo procesador, se realiza el labeling sobre toda la imagen obteniendo así el resultado final.

Portabilidad del algoritmo

Si bien el algoritmo está implementado para la arquitectura mencionada con una topología particular, la idea puede ser utilizada para la implementación sobre otras arquitecturas multiprocesador, ya que el algoritmo es paralelizable por naturaleza.

Resultados obtenidos

Se realizaron pruebas con numerosas imágenes, variando el tamaño, el histograma (porcentaje de pixeles en negro), y la cantidad y forma de los objetos.

De acuerdo a las mediciones del tiempo de ejecución realizadas, *notamos que el algoritmo responde con similar tiempo de ejecución para imágenes de igual tamaño y mismo histograma, sin importar la cantidad y forma de los objetos.*

Por este motivo, en las siguientes tablas se muestran los tiempos de respuesta para diferentes entradas en función del tamaño e histograma únicamente.

Cada entrada de las tablas de pruebas realizadas corresponde a la ejecución del algoritmo con diversas imágenes, variando la cantidad y la forma de los objetos.

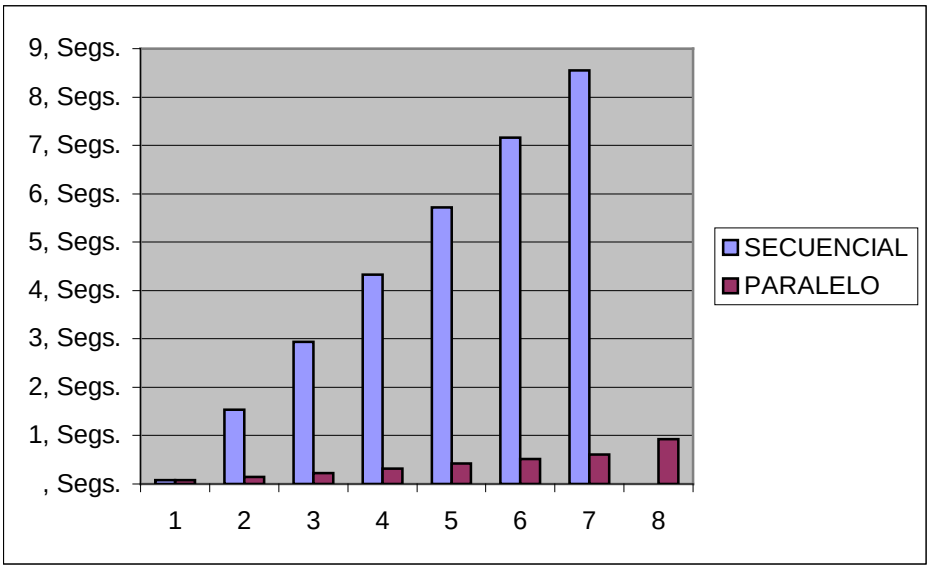
En las tablas existe una columna para los tiempos de ejecución del algoritmo secuencial y otra para los del paralelo. Además hay, una columna que tiene el speed-up alcanzado y otra con la cantidad de puntos de la imagen (área).

Histograma = 0 % Negro

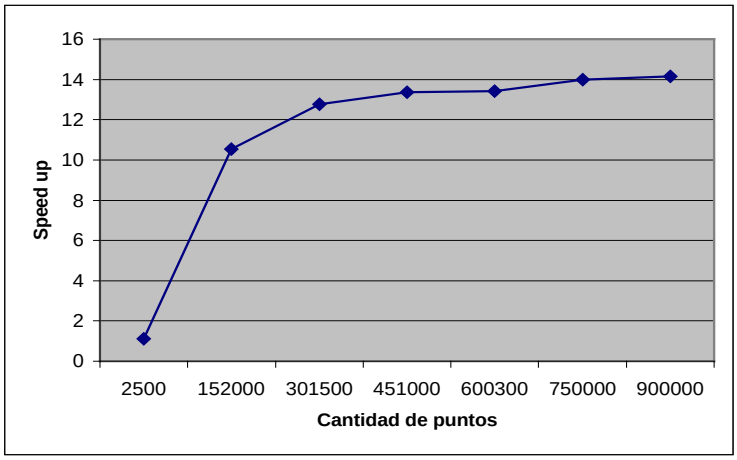
- **Pruebas realizadas:**

Nº	SECUENCIAL	PARALELO	SPEED UP	CANT. PUNTOS
1	,086 Segs.	,077 Segs.	1,117	2500
2	1,53 Segs.	,145 Segs.	10,552	152000
3	2,939 Segs.	,23 Segs.	12,778	301500
4	4,33 Segs.	,324 Segs.	13,364	451000
5	5,718 Segs.	,426 Segs.	13,423	600300
6	7,163 Segs.	,512 Segs.	13,990	750000
7	8,546 Segs.	,604 Segs.	14,149	900000
8	(*)	,928 Segs.	#########	1500000

- **Relación de tiempo entre el secuencial y el paralelo**:



- **Relación entre el tamaño de la imagen con el speed-up:**

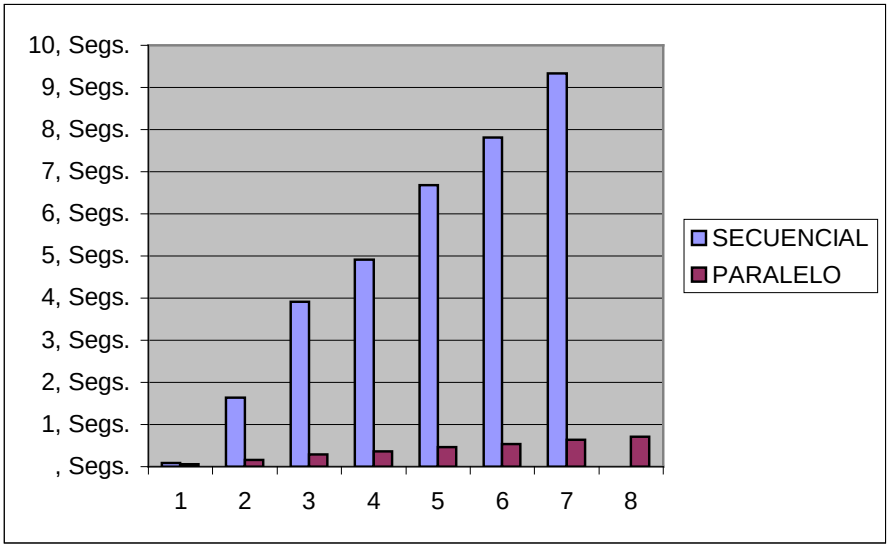


Histograma = 10 % Negro

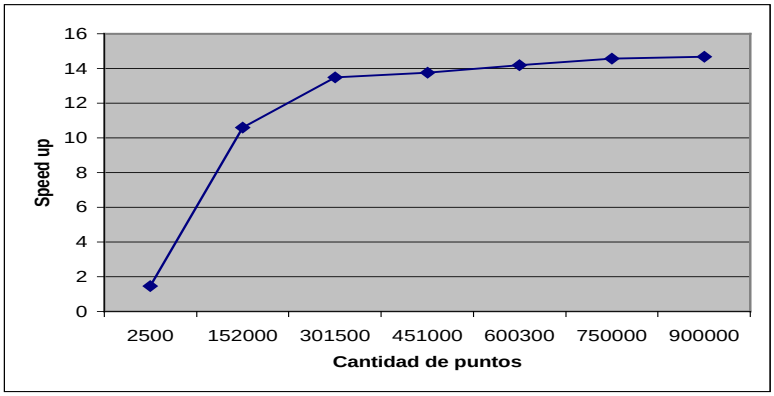
- **Pruebas realizadas:**

Nº	SECUENCIAL	PARALELO	SPEED UP	CANT. PUNTOS
1	,091 Segs.	,062 Segs.	1,468	2500
2	1,644 Segs.	,155 Segs.	10,606	152000
3	3,91 Segs.	,29 Segs.	13,483	301500
4	4,912 Segs.	,357 Segs.	13,759	451000
5	6,674 Segs.	,47 Segs.	14,200	600300
6	7,818 Segs.	,537 Segs.	14,559	750000
7	9,327 Segs.	,636 Segs.	14,665	900000
8	**(*)**	,716 Segs.	#########	1500000

- **Relación de tiempo entre el secuencial y el paralelo**:



- **Relación entre el tamaño de la imagen con el speed-up:**



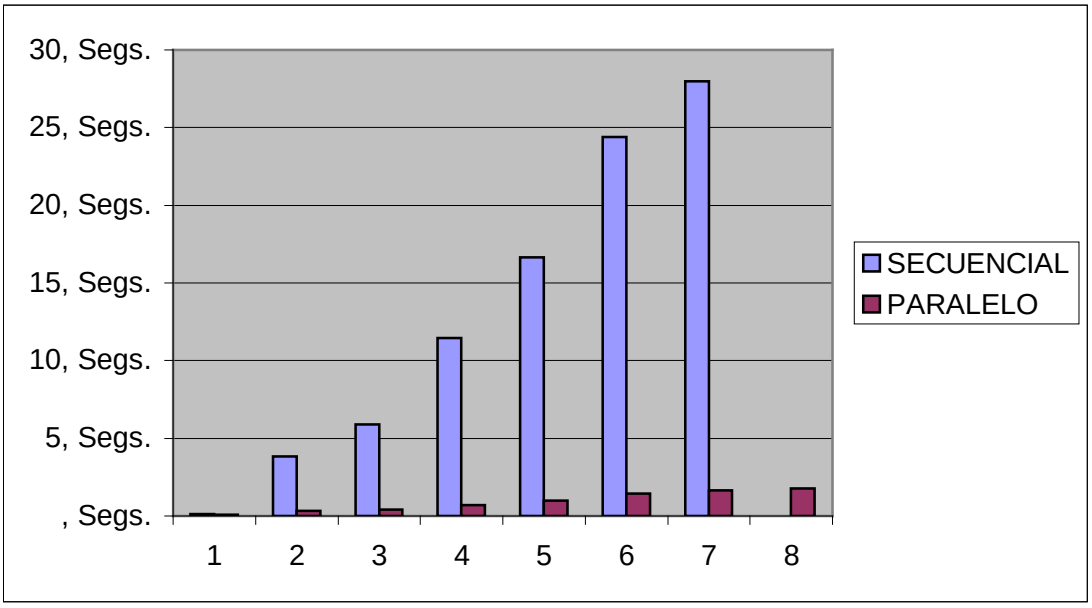
(*) No se pudo realizar la prueba por falta de memoria. **(#)** No están disponibles los datos necesarios.

Histograma = 50 % Negro

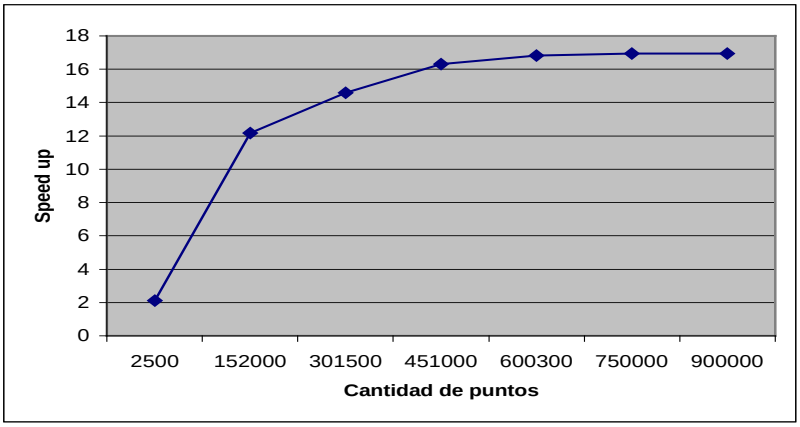
- **Pruebas realizadas:**

Nº	SECUENCIAL	PARALELO	SPEED UP	CANT. PUNTOS
1	,138 Segs.	,065 Segs.	2,123	2500
2	3,818 Segs.	,314 Segs.	12,159	152000
3	5,94 Segs.	,4 Segs.	14,85	301500
4	11,769 Segs.	,698 Segs.	16,861	451000
5	16,8 Segs.	,98 Segs.	17,142	600300
6	24,43 Segs.	1,42 Segs.	17,204	750000
7	27,996 Segs.	1,59 Segs.	17,607	900000
8	**(*)**	1,79 Segs.	#########	1500000

- **Relación de tiempo entre el secuencial y el paralelo**:



- **Relación entre el tamaño de la imagen con el speed-up:**



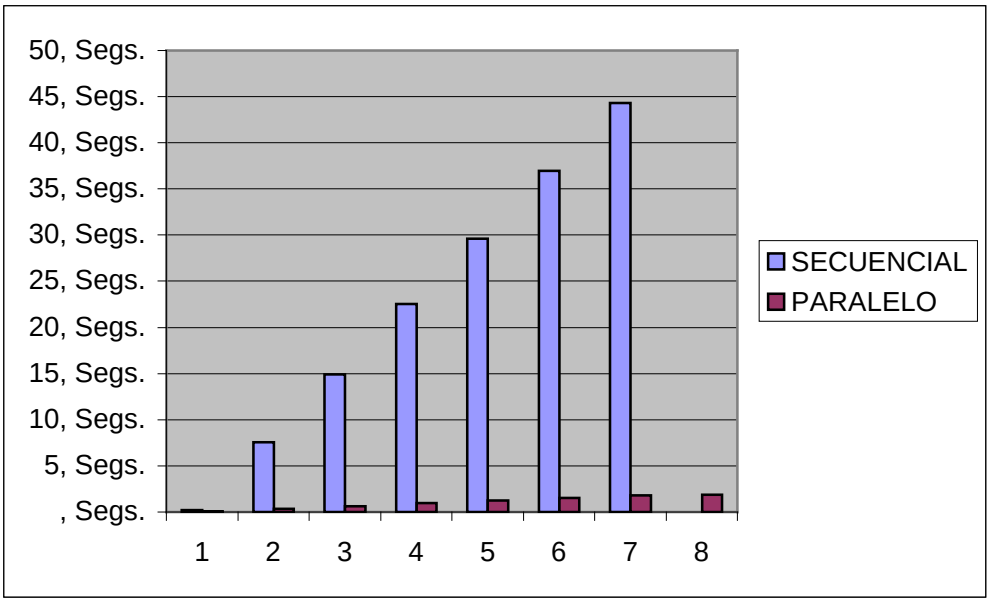
(*) No se pudo realizar la prueba por falta de memoria. **(#)** No están disponibles los datos necesarios.

Histograma =100 % Negro

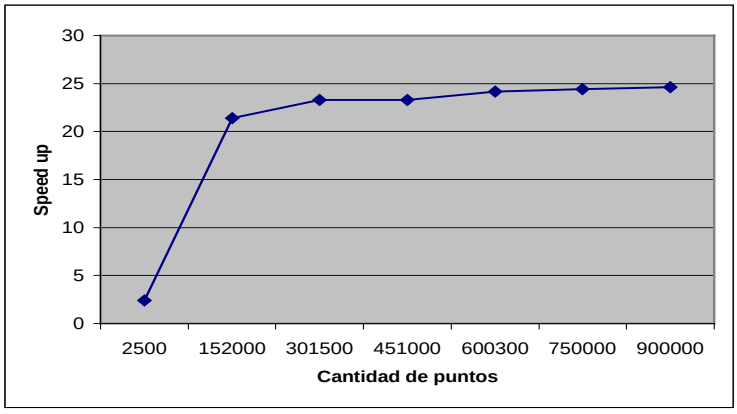
- **Pruebas realizadas:**

Nº	SECUENCIAL	PARALELO	SPEED UP	CANT. PUNTOS
1	,196 Segs.	,082 Segs.	2,390	2500
2	7,577 Segs.	,354 Segs.	21,404	152000
3	14,917 Segs.	,641 Segs.	23,271	301500
4	22,516 Segs.	,967 Segs.	23,284	451000
5	29,597 Segs.	1,226 Segs.	24,141	600300
6	36,939 Segs.	1,512 Segs.	24,431	750000
7	44,318 Segs.	1,799 Segs.	24,635	900000
8	**(*)**	1,868 Segs.	#########	1500000

- **Relación de tiempo entre el secuencial y el paralelo:**



Relación entre el tamaño de la imagen con el speed-up:



(*) No se pudo realizar la prueba por falta de memoria. **(#)** No están disponibles los datos necesarios.

Observando los resultados de las pruebas mostradas anteriormente, se llega a la conclusión de que cuanto mayor es el tamaño de la imagen tratada y mayor es el área ocupada por los objetos (cantidad de pixeles en negro), se obtiene un mejor speed-up.

Se puede ver en la gráfica que relaciona el speed-up con la cantidad de pixeles de la imagen, que las curvas crecen más lentamente en aquellas imágenes con bajo porcentaje de negro.

El speed-up tiende a estabilizarse a partir de un cierto tamaño de imagen, creciendo muy lentamente, como se puede observar en los gráficos. El valor donde el speed-up se estabiliza, difiere, en gran medida, según el histograma. El speed-up se acerca al ideal cuando las imágenes tienen un porcentaje grande de pixeles negros y se aleja cuando el porcentaje es bajo. Esto se nota claramente comparando los casos extremos donde la imagen es toda blanca o toda negra.

Conclusiones

Este trabajo fue realizado dentro de la asignatura Procesamiento Paralelo de la Licenciatura en Informática de la UNLP. El enfoque es totalmente práctico. Se implementó un algoritmo en una arquitectura real y se hicieron pruebas para observar si los resultados obtenidos eran los esperados según la teoría de procesamiento paralelo.

Este trabajo puede ser tomado como la base para trabajos futuros entre los cuales se encuentra la posibilidad de estudiar el tema probando el mismo algoritmo con diferente cantidad de procesadores.

El material relacionado con este trabajo se encuentra disponible en el LIDI para ser consultado.

Bibliografía

[Dav95] David C. Kay & John R. Levine. "Graphics File Formats". 2nd Edition, Windcrest/McGRAW-HILL.
[Hee91] D. W. Heermann, A. N. Burkitt, "Parallel Algorithms in Computational Science", Springer-Verlag, 1991.
[Kum94] Kumar V., Grama A., Gupta A., Karypis G., “Introduction to Parallel Computing. Desing and Analysis of Algorithms”, Benjamin/Cummings, 1994.
[Lei92] F. T. Leighton, “Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes”, Morgan Kaufmann Publishers, 1992.
[Tin98]Tinetti F., De Giusti A. “Procesamiento Paralelo. Conceptos de Arquitecturas y Algoritmos”. Editorial Ciencias Exactas. 1998.
[Tra90] "Transputer Architecture and Overview. Transpurer Technical Specifications. ", Computer System Architects,1990.
[Tra94] "Toolset Reference. Parallel “C” for the transputer".
[Gon92] "Tratamiento Digital de Imágenes". Gonzalez & Woods Adison-Wesley.
[Par95] "Parallel Programing in C for the Transputers", D. Thiébaut.