Departamento de Matemática y Ciencia de la Computación

Primer examen

Segundo Semestre 2018

Algoritmos Distribuidos 26106
Licenciatura en Ciencia de la Computación

Rafael A. Castillo López
rafael@trfs.me

Problema 1

1. Descripción del problema

Enunciado Haz un estudio del comportamiento del algoritmo que determina la distancia de estrellas a partir de una matriz de intensidad de luz. La implementación vista en clase recorre la matriz por filas. Construya una implementación que la recorra por columnas y haga una comparación entre las secuenciales y las paralelas. Recuerde que en clase se vieron dos metricas para medir el rendimiento.

2. Algoritmo

El algoritmo por filas es el mismo visto en clase, y solo se necesitan realizar cambios triviales para obtener la versión por columnas.

3. Analisís de complejidad

Ambos algoritmos secuenciales tienen complejidad lineal con respecto al numero de elementos de la matriz de entrada. Es decir que para una matríz de tamaño $N \times M$. El algoritmo será $\mathcal{O}(MN)$.

4. Experimentos

Todos los experimentos se corrieron en un servidor con las siguientes especificaciones

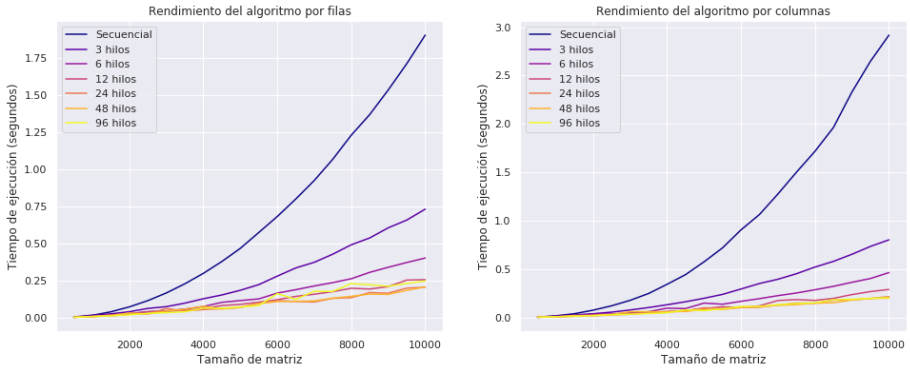
Procesador	Dos Intel Xeon Gold 6126 CPU @ 2.60GHz
Numero de nucleos	12 por procesador, 24 en total (48 hilos con Hyper Threading)
Memoria ram	128 GiB
Sistema operativo	Ubuntu 16.04 x86-64
Memoria secundaria	SSD Samsung MZ7KM240HMHQ0D3 de 240 GiB
Nucleo	Linux help 4.4.0-138

Se corrío cada algoritmo con matrices cuadradas de $N \times N$ para $N = 500, 1000, 1500 \ldots 9500, 10000$. Como escalamos ambas dimensiones de la matríz al mismo tiempo y en las graficas de rendimiento usaremos solo N como eje x, para efectos de este experimento el algoritmo se comportará con complejidad $\mathcal{O}(N^2)$.

Cada punto de prueba se ejecutó con ambos algoritmos secuenciales, y con ambos algoritmos paralelos con 3, 6, 12, 24, 48 y 96 hebras (empezamos de 3 ya que nuestro procesador tiene un número de nucleos multiplo de 3). Se realizaron ademas 40 repeticiones del experimento y se tomo el valor mediano para presentar resultados. Las tablas con los resultados resumidos se encuentran en el apéndice.

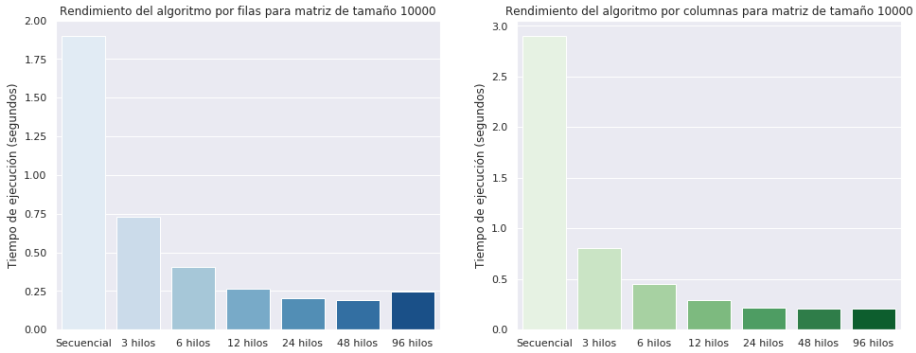
En la figura 1.1 se muestran los tiempos de ejecución de ambos algoritmos. Observamos que aunque teóricamente ambos algoritmos tengan comportamientos asimptoticos exactamente iguales, ¡el algoritmo por columnas toma casi el doble de tiempo que el algoritmo por filas! Esto probablemente se debe a que la baja localidad de datos en el caso por columnas lleva a muchos fallos de cache. Tambien se vé que los algoritmos paralelos ofrecen una mejora dramatica sobre los secuenciales, aunque su comportamiento no sea tan estable.

Figura 1.1: Izquierda: rendimiento de los algoritmos por fila. Derecha: rendimiento de los algoritmos por columna.



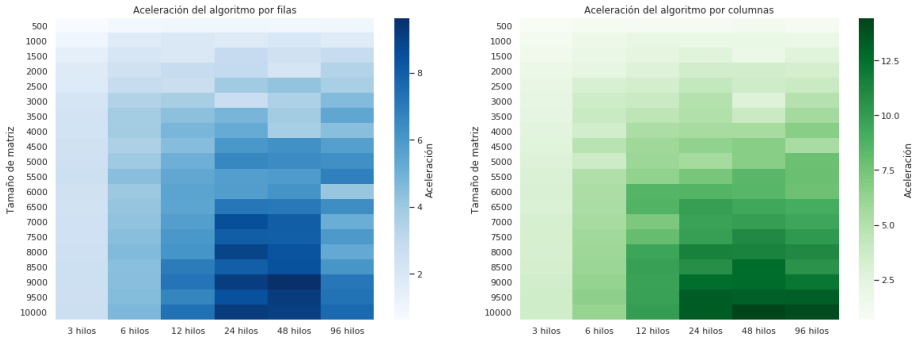
La figura 1.2 nos muestra el rendimiento para el caso especifico de una matríz de 10000×10000. Aqui se aprecia mejor como mejora el rendimiento con mas hilos. Curiosamente podemos ver en el algoritmo por filas que el rendimiento con 96 hilos es peor que el rendimiento con 48. 48 hilos es el maximo que el procesador usado en las pruebas puede correr en paralelo (contando los nucleos virtuales o HyperThreads). Esto no se presenta en la versión por columnas. Hipotetizo que la razón de esto es que aunque el mas rápido algoritmo por filas llegue a un cuello de botella ocasionado por el cambio de contexto entre hilos, el algoritmo por columnas aún es limitado por su mas lento acceso a memoria.

Figura 1.2: Rendimiento para una matriz de 10000×10000



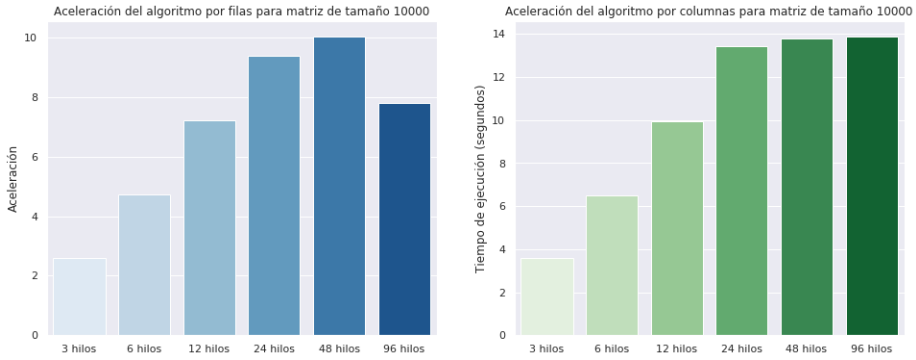
En el mapa de calor de la figura 1.3 podemos ver la aceleración de los algoritmos probados. Podemos ver la tendencia general de que la aceleración crezca con el numero de hilos y el tamaño del problema, con la excepción previamente mencionada del algoritmo por filas usando 96 hilos.

Figura 1.3: Mapa de calor de aceleración



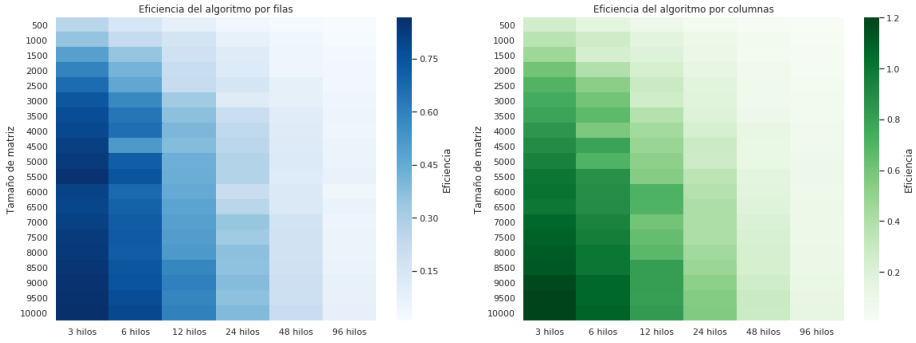
De nuevo acercandonos a nuestro caso extremo de la mátriz de 10000×10000 en la figura 1.4, observamos que en ambos casos nuestra aceleración crece hasta cierto punto mientras mas hilos agreguemos, aunque nuestra delta de aceleración baje entre mas hilos empleemos. Tambien vemos el comportamiento con 96 hilos en ambos casos: en el algoritmo por columnas, no vemos mejora al agregar estos hilos, mientras que en el algoritmo por filas, perdemos tiempo, como se habia mencionado antes.

Figura 1.4: Aceleración para mátriz de 10000×10000



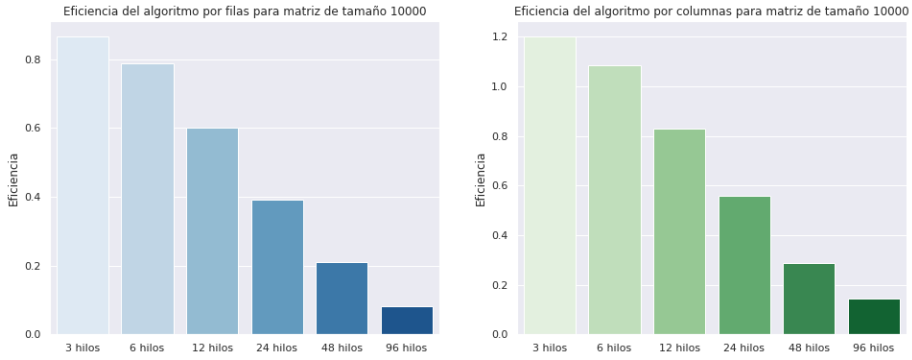
En el caso de la eficiencia, vemos una historia muy parecida para ambos algoritmos. Aunque el tiempo que toman menos tiempo real, la eficiencia del uso de recursos en realidad baja dramaticamente aumentando el número de hilos. La figura 1.5 muestra este comportamiento.

Figura 1.5: Mapa de calor de eficiencia



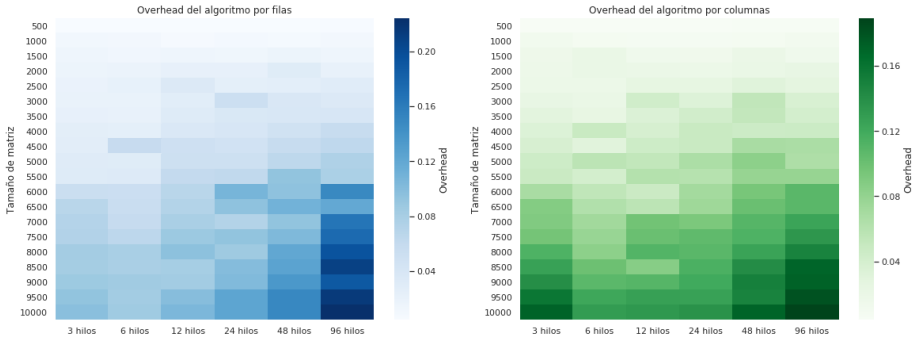
En nuestra figura 1.6, mostrando la eficiencia para nuestra mátriz de 10000×10000, vemos como nuestra eficiencia baja constantemente, culminando en la eficiencia con 96 hilos, que se vé dramaticamente reducida por el tiempo gastado en cambios de contexto.

Figura 1.6: Eficiencia para mátriz de 10000 × 10000



Para acabar, obervamos el overhead que sufren los algoritmos. Los resultados son congruentes con la historia hasta ahora. El overhead aumenta con el número de hilos, y ve un incremento dramatico en las corridas con 96 hilos. Estos resultados estan graficados en la figura 1.7

Figura 1.7: Overhead de los algoritmos



5. Conclusión

Ambos algoritmos de detección de estrellas nos muestran propiedades típicas de los algoritmos distribuidos. Vemos que el algoritmo paralelo en general es mucho más rapido que el secuencial. Pero tambien vemos que la eficiencia de uso de recursos baja rapidamente entre mas paralelizemos el algoritmo.

Vemos tambien que hay cierto punto cuando agregar mas hilos se vuelve contraproducente cuando la cantidad de hilos supere el numero de núcleos de procesador con los que contamos. Este punto obviamente depende del procesador, pero es importante estar consciente de su existencia.

Problema 2

1. Introducción

El objetivo de este trabajo es construir un algoritmo que dadas las coordenadas de dos piezas de ajedrez, de las cuales una es un alfil, determine si el alfil puede atacar a la otra pieza.

2. Procedimiento

Un alfil ataca a la pieza contraria si esta se encuentra en alguna de las diagonales Dado que cada pieza ocupa una posición en el tablero es suficiente con calcular la pendiente de la recta que pasa por las coordenadas en las cuales se encuentran ambas piezas.

Si el valor de dicha pendiente es 1 o -1, entonces el alfil ataca a la pieza contraria.

2.1. Restricciones

Se asume que las coordenadas de ambas piezas son válidas y que cada ambas piezas no ocupan la misma posición.

3. Estructuras de Datos Utilizadas

Para este problema no se ocupan estructuras de datos.

4. Algoritmo

El algoritmo para ...

Algorithm Bootstrap

Input: X, Y set of n observations
nbi Number of bootstrap iterations
p percent of the confidence interval
Output: x_l, x_r Confidence interval

```
1  ρ ← CorrelationCoef(X, Y)
2  for i ← 1 to nbi do
3      X', Y' ← SampleFrom(X, Y, n)
4      cint_i ← CorrelationCoef(X', Y')
5  Sort(cint)
6  x_l, x_r ← ConfidenceInterval(cint, p)
7  return x_l, x_r
```

El algoritmo primero verifica si ambas piezas se encuentran en la misma fila (línea 1) o en la misma columna (línea 3); en ambos casos el resultado es *Fail*. Si ambas piezas no se encuentran en la misma fila o columna se calcula la pendiente de la recta que pasa por las coordenadas en las cuales se encuentran ambas piezas (línea 5) y luego se verifica si el valor de la pendiente es ± 1 (línea 6).

4.1. Análisis de Complejidad

El algoritmo propuesto tiene costo constante.

5. Implementación

El algoritmo está implementado en lenguaje C como se muestra en la figura a continuación:

5.1. Plataforma Computacional

El programa fue ejecutado eun un computador con las siguientes características:

- **Procesador:** Intel(R) Pentium(R) 4 CPU 3.00GHz
- **Memoria RAM:** 994312 kB
- **Sistema Operativo:** Linux - Ubuntu 9.04

6. Resultados Experimentales

En este caso es trivial ...

Datos experimentales del problema 1