PRACTICA 1

Programación Paralela

PEDRO PIZARRO HUERTAS - DAVID MARTIN VILAR

Referente al punto 2.

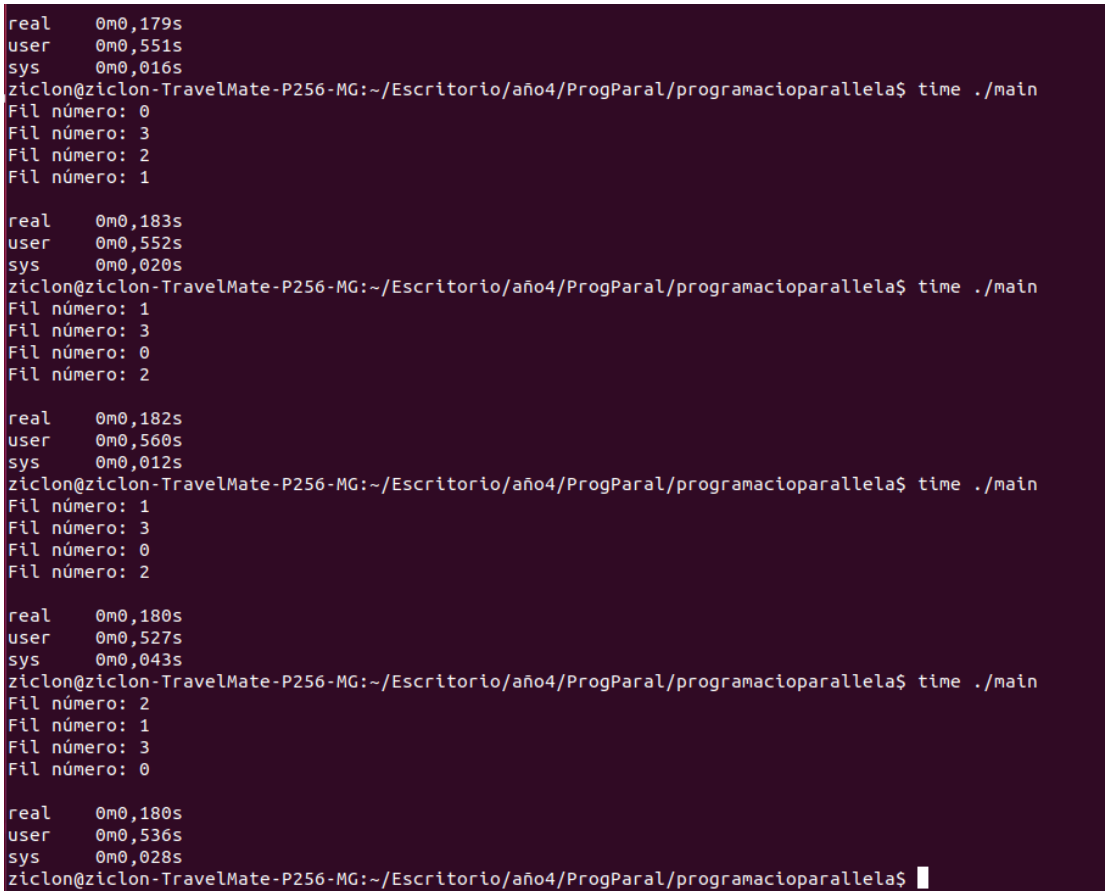
Se ha hecho uso de dos iteraciones “for” para llevar a cabo este punto.

Primero, se ha hecho que el primer for (con el valor ‘x’) recorra la anchura de la imagen y el segundo (con el valor ‘y’) recorra la altura de la imagen. Si lo hacemos de esta forma, va a recorrer todos los valores de y por cada x, es decir, cogerá un dato de cada fila en cada iteración, por lo que estará cogiendo datos que están separados en la memoria y le tomará más tiempo.

Los tiempos conseguidos de esta forma son estos:

```
real    0m0,541s
user    0m1,680s
sys     0m0,040s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 3
Fil número: 1
Fil número: 2

real    0m0,556s
user    0m1,683s
sys     0m0,033s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 3
Fil número: 1
Fil número: 2

real    0m0,536s
user    0m1,678s
sys     0m0,044s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 3
Fil número: 2
Fil número: 1

real    0m0,552s
user    0m1,751s
sys     0m0,016s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 2
Fil número: 1
Fil número: 0
Fil número: 3

real    0m0,534s
user    0m1,680s
sys     0m0,024s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$
```

Para poder mejorar esto se ha hecho que se recorran toda la fila en cada iteración, haciendo que el primer for recorra las y y el segundo recorra las x, es decir, que cuando se itere por cada fila de la imagen se recorran todas las columnas consiguiendo así que acceda a datos que se encuentran a menos distancia entre ellos haciendo este proceso más eficiente:



Referente al punto 3.

En este caso hacemos que las tareas que se llevan a cabo en esta transformación se lleve a cabo por distintos hilos de ejecución (Tantos como el procesador permita).

De esta forma logramos estos tiempos:

```
real    0m0,083s
user    0m0,161s
sys     0m0,024s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 3
Fil número: 1
Fil número: 2

real    0m0,084s
user    0m0,155s
sys     0m0,029s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 3
Fil número: 2
Fil número: 1

real    0m0,083s
user    0m0,156s
sys     0m0,029s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 1
Fil número: 3
Fil número: 2

real    0m0,083s
user    0m0,157s
sys     0m0,029s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 2
Fil número: 1
Fil número: 3

real    0m0,087s
user    0m0,177s
sys     0m0,013s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$
```

Comentar que se ha llevado a cabo de la misma forma que la segunda forma comentada en el apartado 2, el resto de pruebas también lo estarán.

Se puede notar una mejoría notable en cuestión de eficiencia.

Referente al punto 5.

Tenemos 3 metodos de schedule aplicados:

-static: La division entre tareas e hijos se determina de forma estatica (no varía) desde inicio a final de la ejecución del programa ejecutado de esta forma.

```
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 1
Fil número: 3
Fil número: 0
Fil número: 2

real    0m0,085s
user    0m0,148s
sys     0m0,040s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 0
Fil número: 3
Fil número: 2
Fil número: 1

real    0m0,085s
user    0m0,150s
sys     0m0,036s
ziclon@ziclon-TravelMate-P256-MG:~/Escritorio/año4/ProgParal/programacioparallela$ time ./main
Fil número: 1
Fil número: 0
Fil número: 3
Fil número: 2

real    0m0,082s
user    0m0,134s
sys     0m0,050s
```

También podemos definir el tamaño de chunks, los chunks son fragmentos de información, en este caso, fragmentos de las iteraciones a procesar que se asignan a los hilos con la tecnica round robin en el caso del static(Una cola circular en la cual se le asigna a cada proceso un tiempo determinado), esto lo reflejaremos en la tabla de a continuación.

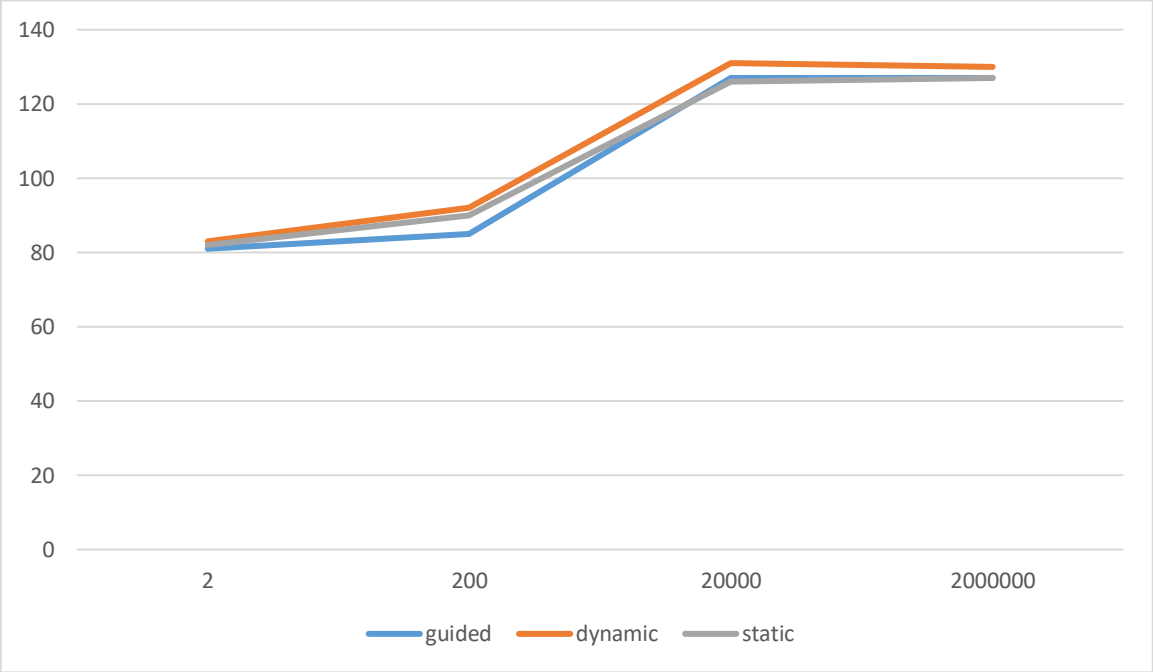
-dynamic y guided:

Se ha hecho una grafica con las medidas de tiempo (en ms) para 2,200,20000 y 2000000 (Hemos decidido parar en 2000000 debido a que hacemos 3840×2160 operaciones, que es aproximadamente esa cifra.) de chunk size, para los casos de guided y dynamic.

En ambos metodos la asignacion de tareas a los hijos se realiza a medida que se ejecuta el programa, pero el chunk size toma distintos comportamientos en las dos planificaciones.

En el guided cuando especificas el chunk size especificas la medida MINIMA a la que va a poder reducirse el chunk.

En el dynamic las iteraciones se dividen en chunks del tamaño indicado de forma estricta.



Podemos ver que no existe una diferencia notable entre ellos, pero en este caso el mejor que podríamos usar sería el static, debido a que todas las iteraciones son la misma operación repetida, y todos los chunks serán de un mismo tamaño, por eso conviene que la medida sea estatica, que la medida varíe no nos aporta nada en este caso.