David Martos
Mariona Farré
Grupo: paco1208
Curso: 2022-23

PACO:
Laboratorio 3

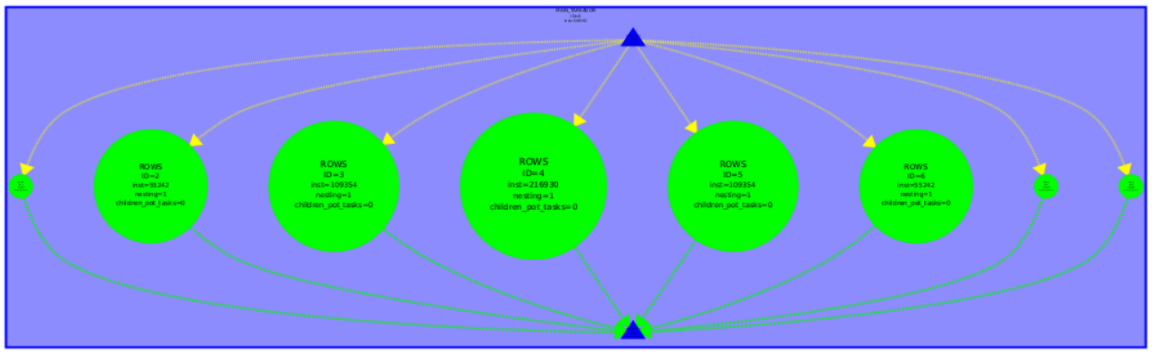
Índice:

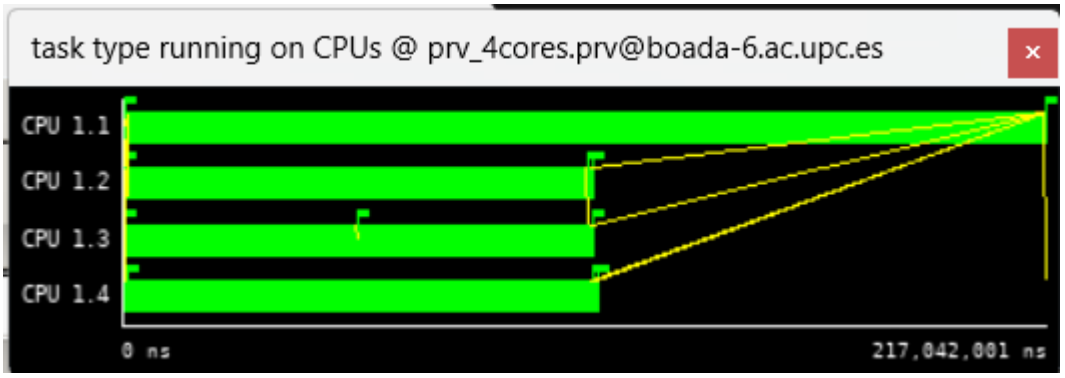
1. Task decomposition analysis for the Mandelbrot set computation
    1.1. Task decomposition analysis with Tareador
        1.1.1. Row strategy

For the deliverable: Which are the two most important characteristics for the task graph that is generated? Save the TDG generated for later inclusion in the deliverable.

Una de las características de el código original es que algunas las tareas verdes tienen diferentes tamaños y son mas grandes entre otras y la otra característica es que entre tascas no hay dependencias, solo tienen una dependencia con la tasca inicial que es la de color azul.

./run-tareador.sh mandel-tar

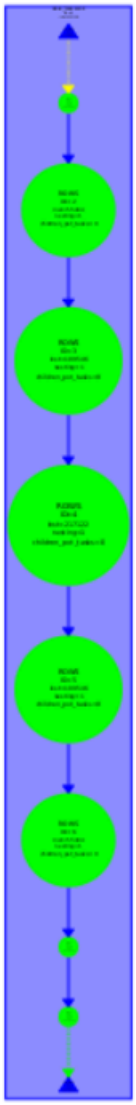


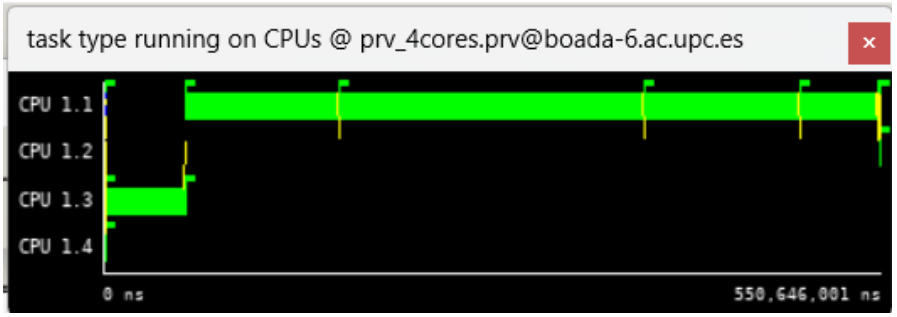


For the deliverable: Which are the two most important characteristics for the task graph that is generated? Which part of the code is making the big difference with the previous case? How will you protect this section of code in the parallel OpenMP code that you will program in the next sessions? Save the new TDG generated for later inclusion in the deliverable.

Ahora una de las características del grafo es que aun hay algunas las tareas verdes tienen diferentes tamaños però ahora la otra característica es que entre tascas si hay dependencias, una dependencia que forma una serialización entre todas las tascas.
Esta zona de código necesita ser protegida para prevenir que diferentes threads accedan en la misma variable y cambien el color, ya que este sería modificado repetidas veces cuando debe estar protegido.

./run_tareador.sh mandel-tar -d

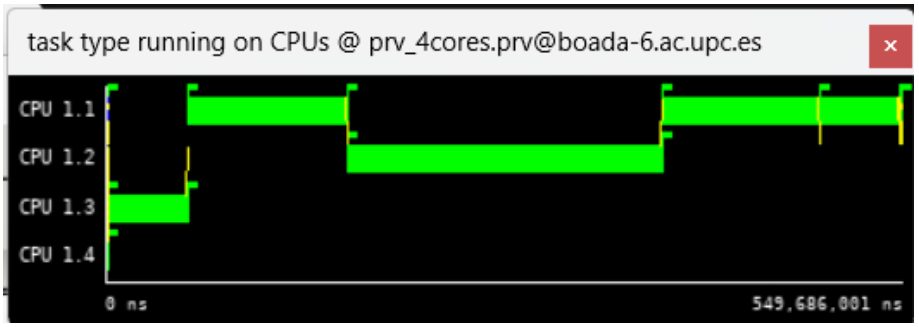




For the deliverable: What does each chain of tasks in the task graph represents? Which part of the code is making the big difference with the two previous cases? How will you protect this section of code in the parallel OpenMP code that you will program in the next sessions? Don't forget to save the new TDG generated for later inclusion in the deliverable.

Como los últimos dos grafos, hay algunas las tareas verdes tienen diferentes tamaños y son más grandes entre otras y si hay dependencias en el grafo, una dependencia que forma una serialización entre todas las tascas però ahora con más dependencias entre si.
En este código, los threads acceden en k-1 posiciones en el histograma y el código se vuelve secuencial porque no pueden acceder a un elemento que aún no existe.
Y tambien ocurre lo mismo que el código anterior y es que los threads tendrán acceso a una posición de memòria todos a la vez haciendo que de un resultado incorrecto, porque sólo un thread debe acceder en esa zona, y no todos a la vez lo que dará errores en la solución.

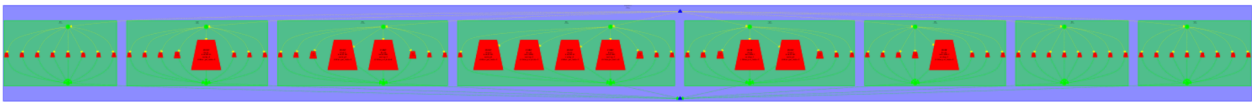
./run_tareador.sh mandel-tar -h

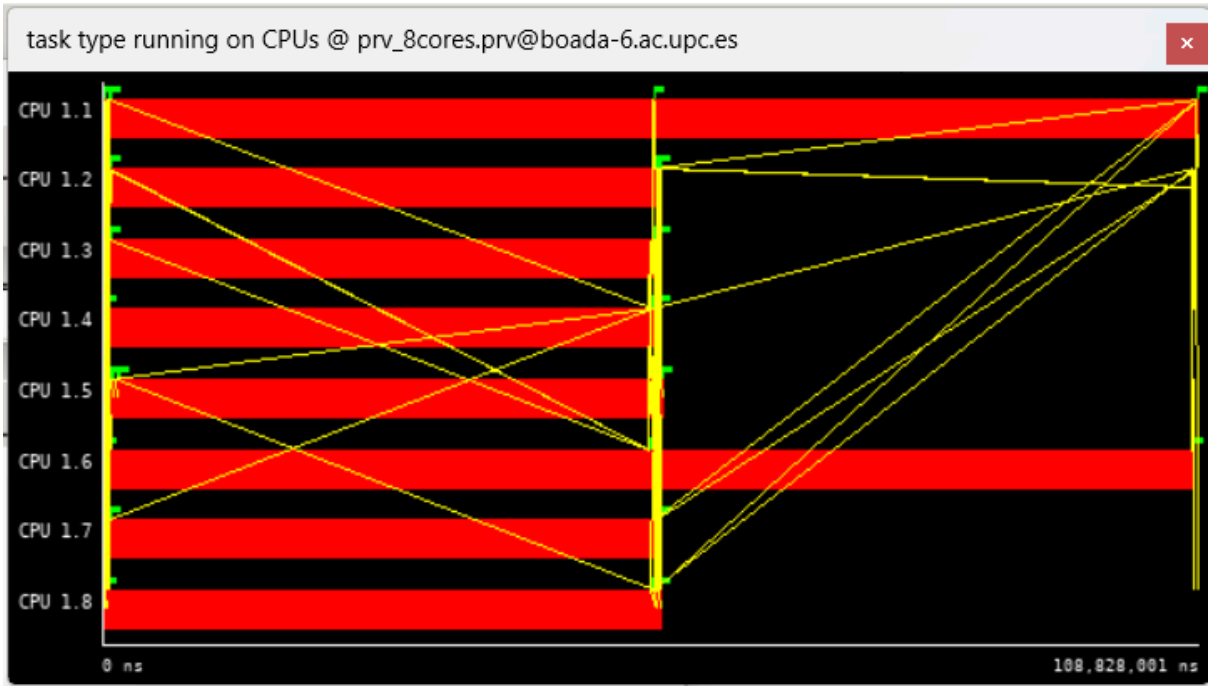


1.1.2. Point strategy

For the deliverable: Which is the main change that you observe with respect to the Row strategy?
Tenemos las mismas características que la anterior estrategia, de no tener dependencias entre tascas y tener tascas que son más grandes que otras.

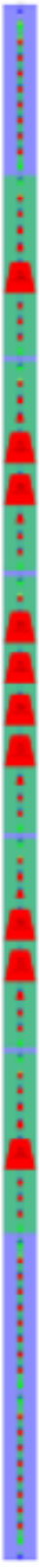
./run_tareador.sh mandel-tar

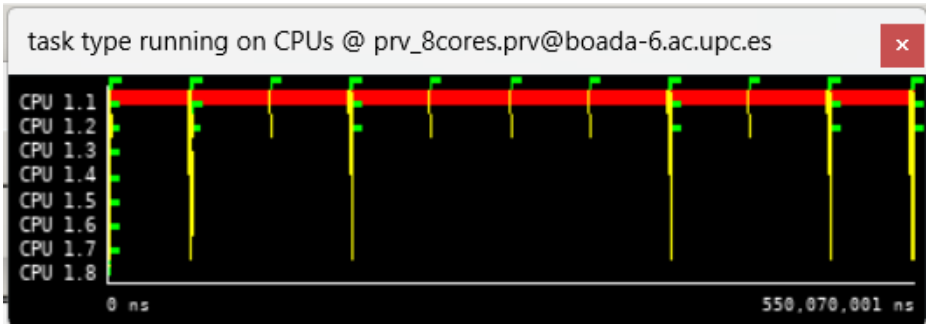




En el siguiente grafo también tiene las mismas características que el anterior, pero ahora con muchas más tascas, ahora con dependencias entre tascas y tascas con diferentes tamaños.

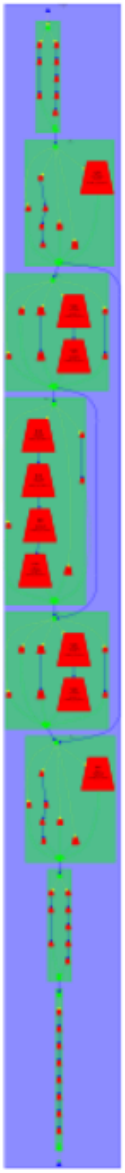
./run_tareador.sh mandel-tar -d

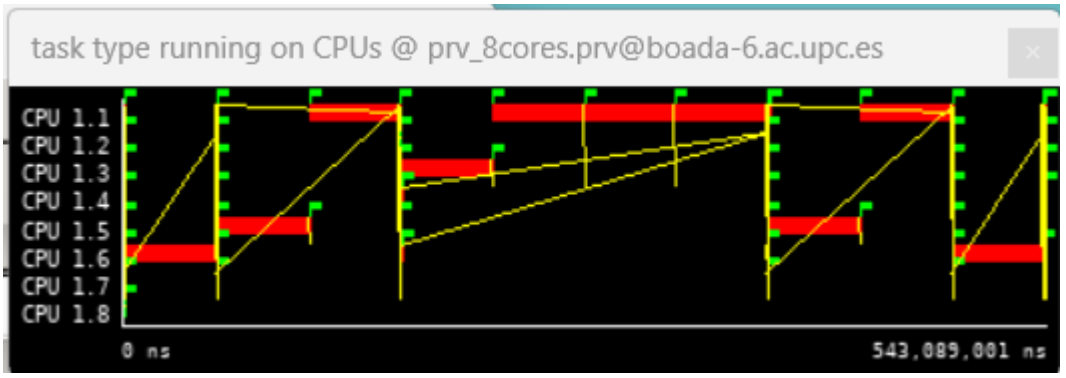




En el último grafo de esta estrategia, podemos ver que es el más diferente de los anteriores ya que parezca que las tascas se ejecuten casi paralelamente, continúan siendo ejecutadas secuencialmente por culpa de las grandes tascas que tienen dependencias de datos.

./run_tareador.sh mandel-tar -h





For the deliverable: Which is the main change that you observe with respect to the Row strategy?

La diferencia más grande entre las dos estrategias es la cantidad de tareas que se crean, ya que en la Row strategy se crean muchísimas más tareas por culpa de la tarea que está dentro del loop.

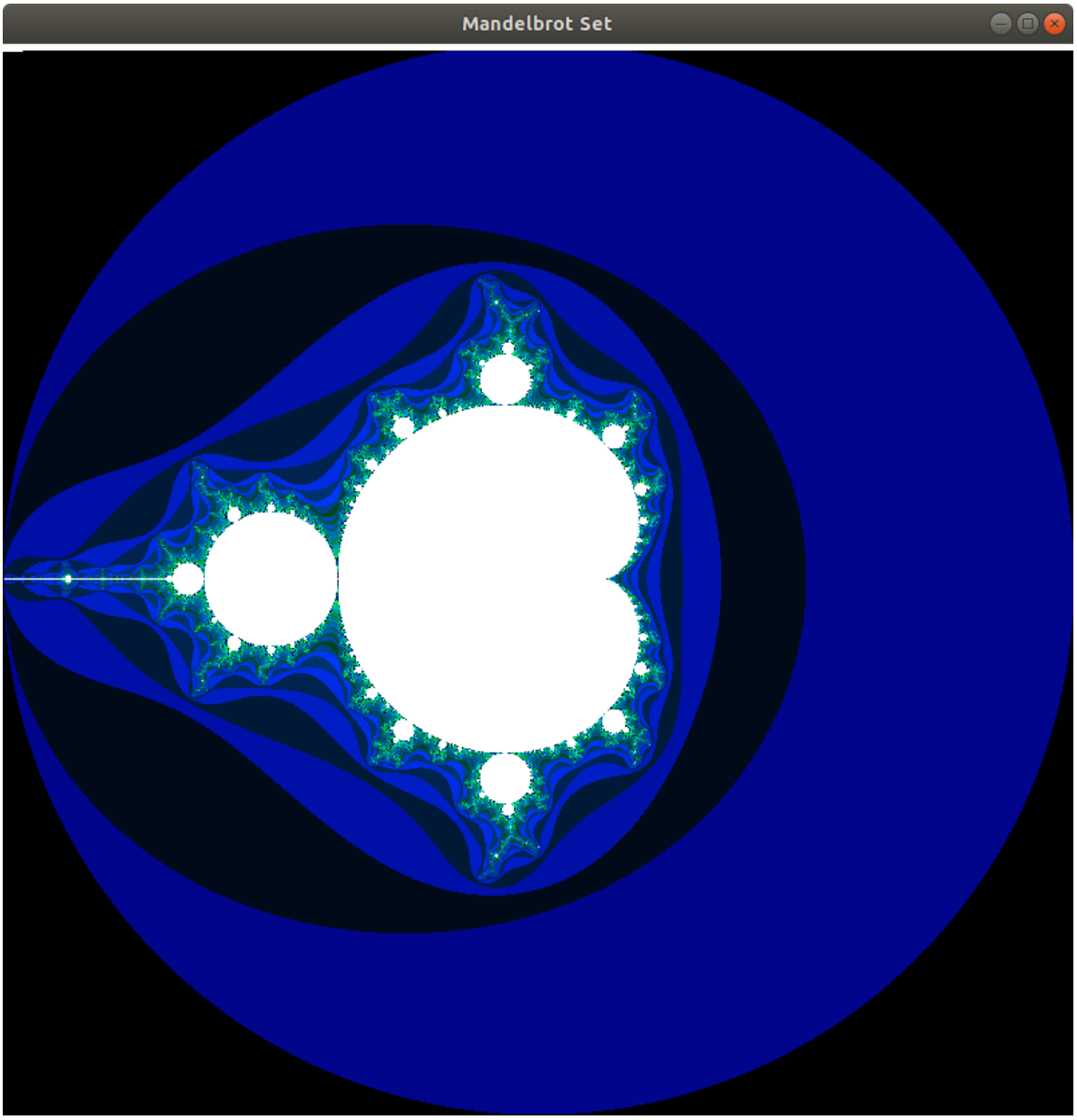
De ambas estrategias, la más adecuada en este caso es la de Row strategy, ya que hay tareas mucho más grandes que otras, y en cada ejecución, ya que se ejecutan secuencialmente, tiene muchas menos tareas para calcular, además, la estrategia Point tiene problemas con los el número de tareas al crear las tareas cada iteración en ambos bucles.

2. Implementation and analysis of task decompositions in OpenMP

2.1. Point decomposition strategy

Is the image generated correct?

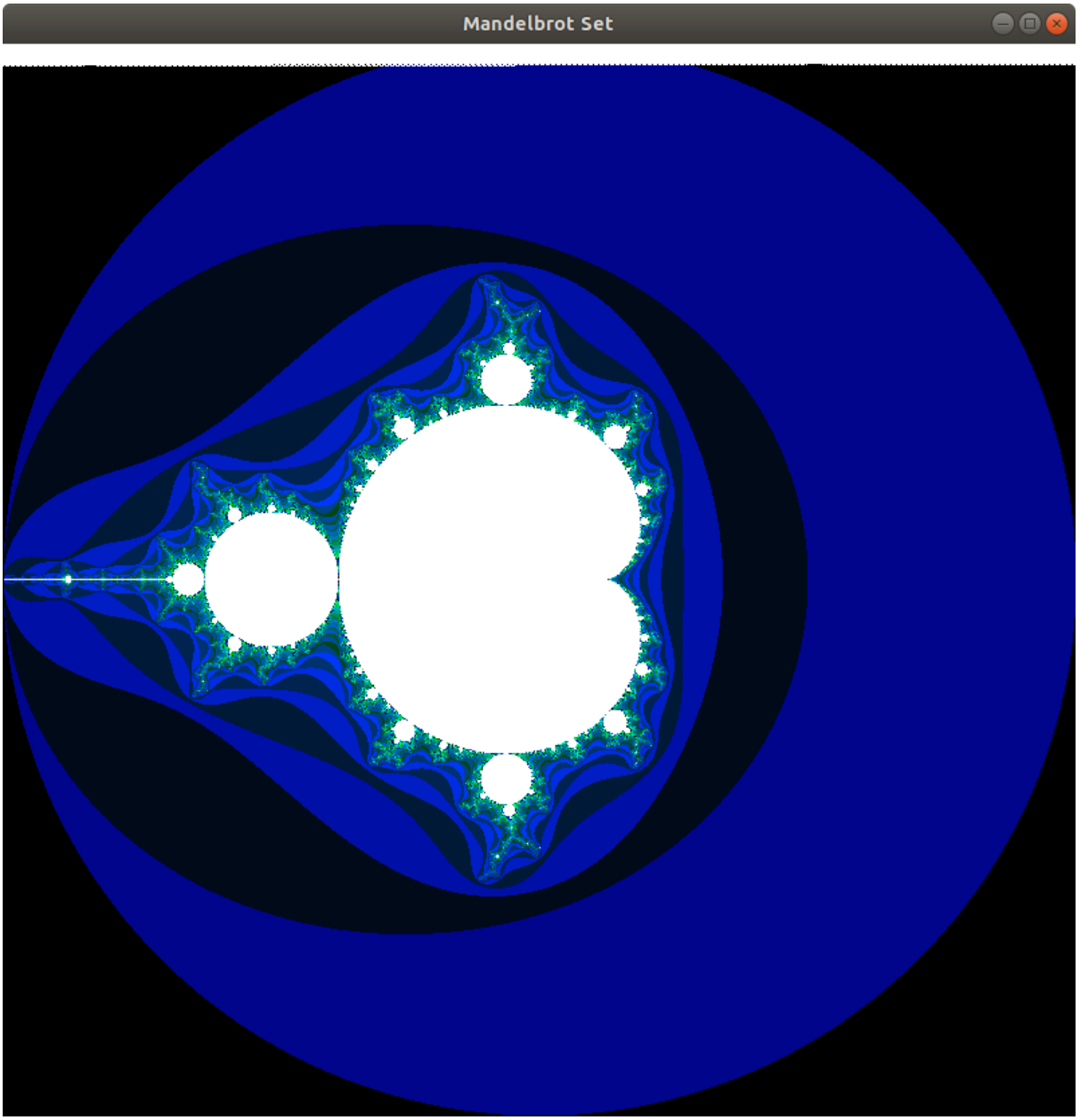
Si la imagen se genera correctamente y comparando los ficheros creados -o no hay ninguna diferencia




```
paco1208@boada-6:~/lab3/lab3$ OMP_NUM_THREADS=1 ./mandel-omp -o -d -h -i 10000

Computation of the Mandelbrot set with:
    center = (0, 0)
    size = 2
    maximum iterations = 10000

Mandelbrot set: Computed
Histogram for Mandelbrot set: Computed
Writing output file to disk: output_omp_1.out

Click on a point in the display to get its coordinates
Press any key (with focus in display) to end the program
X connection to localhost:17.0 broken (explicit kill or server shutdown).
paco1208@boada-6:~/lab3/lab3$ cmp output_originalseq.out output_omp_1.out
paco1208@boada-6:~/lab3/lab3$
```

Ejecución con 2 threads y con 10000 iteraciones por punto:
OMP_NUM_THREADS=2 ./mandel-omp -d -h -i 10000

Is the image generated still correct?
La imagen se continúa creando correctamente, podemos ver diferencias en los colores pero se genera bien igualmente.




```
paco1208@boada-6:~/lab3/lab3$ OMP_NUM_THREADS=2 ./mandel-omp -o -d -h -i  10000

Computation of the Mandelbrot set with:
    center = (0, 0)
    size = 2
    maximum iterations = 10000

Mandelbrot set: Computed
Histogram for Mandelbrot set: Computed
Writing output file to disk: output_omp_2.out

Click on a point in the display to get its coordinates
Press any key (with focus in display) to end the program
coordinates = (1.3, 0.34)
X connection to localhost:17.0 broken (explicit kill or server shutdown).
paco1208@boada-6:~/lab3/lab3$ cmp output_originalseq.out output_omp_2.out
paco1208@boada-6:~/lab3/lab3$
```