OpenMP Odd Even Sort

Iñigo Manuel Diez Canseco Fuentes
Univesidad Católica San Pablo
Arequipa, Perú
Email: inigo.diezcanseco@ucsp.edu.pe

I. Trabajo de OpenMP

El trabajo de OpenMP es para implementar el método de ordenamiento ODD-EVEN SORT usando las dos formas presentadas en el capítulo 4 del libro de Peter Pacheco.[1]

Además, se debe explicar cuál es la diferencia de cada forma implementada y replicar el cuadro de resultados, en su caso deben hacer los experimentos con mayor cantidad de hilos, 8, 16, 32 y mayor cantidad de datos.

El *link* del repositorio se encontrará en el siguiente enlace[1].

II. Arquitectura del Computador

La aquitectura de mi computadora es la siguiente: Procesador Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz 2.50 GHz
RAM instalada 16.0 GB.
Tipo de sistema Sistema operativo de 64 bits.

III. Explicación

Para comenzar, se tiene que mencionar que el funcionamiento de Odd-Even sort esta basado en 3 bucles, primero esta el bucle más grande donde seguirá ejecutándose hasta que este ordenado y los otros bucles se tratan de ordenar utilizando los índices, es muy parecido a Buble sort.

Paralelizar este ordenamiento se sabe que el bucle principal no puede paralelizarse ya que realiza el cambio de odd a even, entonces lo que puede paralelizarse sería los otros dos bucles que están a dentro para distribuirlos en threads.

IV. Desarollo

Se tiene que mencionar que ambos métodos son similares excepto por la directiva utilizada para paralelizar los pares a ordenar.

Para comenzar la forma *two paralel directives*, cada iteración de los bucles dentro del bucle principal, se realiza un fork de los threads mientras que *two for directives*, reutiliza threads hasta terminar con todas las iteraciones.

V. Cuadro Comparativo

Aquí esta el cuadro comparativo de los códigos odd-even sort diferentes para poder saber cual es el más eficiente y la razón:

VI. Conclusión

Podemos concluir que utilizar las dos directivas for es mucho más eficiente debido a que reutiliza threads que ya habían hecho fork, en vez de un fork a cada thread en cada iteración en los bucles que están adentro del bucle principal.

Referencias

[1] P. Pacheco, *An introduction to parallel programming*. Elsevier, 2011.

[1] https://github.com/inigomanuel/Computacion_Paralela_y_Distribuida/tree/main/05_Quinta_Tarea

Pruebas con 20000 (segundos)			
Threads	8	16	32
Two for directives	33.916	21.384	20.987
Two paralel for directives	38.249	28.926	26.808

Figura 1. Cuadro Comparativo de los Códigos.