Informe de OpenMP Odd-Even Sort

Fabrizio Paul Rosado Málaga
Universidad Católica San Pablo
Arequipa, Perú
Email: fabrizio.rosado@ucsp.edu.pe

I. Introducción

A continuación se muestra el desarrollo de la ejecución del algoritmo Odd-Even Sort utilizando la librería OpenMP en el lenguaje C.

II. Desarrollo

Primero, expliquemos como funciona el Odd-Even Sort, este es un algoritmo de ordenamiento bastante parecido al Bubble Sort, pero se ordena en fases que pueden ser pares o impares. En cada una de estas fases, se seleccionan pares de número para ordenarse, pero estos pares cambian dependiendo de la fase. Las fases pares, ordenan el par cero, el par dos, y así. Mientras que en las fases impares, se ordenan el par uno, el par tres hasta acabar los pares disponibles en el arreglo.

Para paralelizar este ordenamiento, se tiene en cuenta la implementación inicial del ordenamiento secuencial que esta conformado por un bucle principal con dos bucles anidados en su interior. El bucle principal no se puede paralelizar debido a que este realiza el cambio entre fases del ordenamiento, y existe dependencia entre una fase y la siguiente. Pero los bucles en el interior que seleccionan los pares a ordenar, pueden ser distribuidos en threads para ser realizados en paralelo.

En este caso se presentan dos implementaciones utilizando OpenMP para realizar la paralelización. Esta biblioteca tiene sus directivas

Para ser objetivos con los resultados, cabe mencionar que varían dependiendo de la computadora en la que se esté, debido al número de threads y núcleos en el CPU. En este caso, se está ejecutando en una computadora con las siguientes especificaciones:

Procesador Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
RAM instalada 16.0 GB (15.8 GB utilizable)
Tipo de sistema Sistema operativo de 64 bits, procesador x64

	Pruebas con arreglos de 200000 números aleatorios			
	N° de Threads	8	16	32
Método	Two parallel for directives	36.776 seg.	26.538 seg.	29.088 seg.
	Two for directives	34.539 seg.	22.835 seg.	23.687 seg.

Fig. 1. Tiempos de Ejecución del Odd-Even Sort con OpenMP

Como se puede ver en la figura 1, se muestran los tiempos de ejecución de las dos formas presentadas en el libro de Pacheco. Los parámetros utilizados para evaluar los códigos, fueron 200000 números generados aleatoriamente, con diferentes números de threads, entre 8, 16 y 32.

Los tiempos del primer método con dos directivas parallel for en la primera fila, muestran un desempeño decente, pero debido al número de núcleos en los 32 threads, la eficiencia decae para ambos métodos.

Cabe destacar que el segundo método usando las dos directivas for, es más eficiente que el anterior en todos los resultados. Esto tiene su explicación, y es que el primer método y el segundo son bastante parecidos, lo único que cambia es la directiva utilizada para paralelizar los pares a ordenar. Y es que el primer método, para cada iteración de los bucles internos se realiza un fork de los threads a utilizar. Mientras que en el segundo método, los threads a los que se les hizo un fork al inicio, se siguen utilizando hasta acabar las iteraciones de los bucles internos.

Al final cuando se terminan de realizar las operaciones de los bucles internos paralelizados, existe una barrera en la cual, los threads que acaban antes esperan al resto para terminar y realizar un join, de esa forma se finaliza la fase actual.

El código puede ser encontrado en el repositorio de Github al cual se accede cliqueando en el hipervínculo: Github/FPRM2021/Comp_Paralela

III. Conclusiones

Se demostró el que utilizar las dos directivas for en el Odd-Even Sort es más eficiente que utilizar la directiva parallel for. Esto debido a la reutilización de

threads a los que se habían hecho fork anteriormente, en lugar de hacer fork a los threads cada vez que hay una iteración en uno de los bucles internos.