LABORATORIO: Trabajo sobre OpenMP 1

Christian David Ytuza Cusirramos
Departamento de Ciencia de la Computación
Universidad Católica San Pablo
Arequipa 2020
christian.ytuza@ucsp.edu.pe

1 Introducción

En este informe probaremos el algoritmo modificado de bubble sort usando OpenMP, las especificaciones de la computadora en la cual se realizo el experimento son:

- Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz
- Memoria Ram: 16 GB
- Tarjeta Grafica NVIDIA GeForce RTX 2070

2 Implementación

La implementación de los algoritmos difiere en la parte justamente del OpenMP la cual crea threads antes del primer for y el segundo el cual crea los threads entre el primer y el segundo for, esto provoca en el primer de los casos que cada vuelta del primer for se cree de nuevo los threads, a diferencia del segundo que usa los threads ya creados, esto puede significar un tiempo extra para la creación de los threads en cada iteración. (Pacheco, 1997)

3 Resultados

Las pruebas se realizaron con un array de 10 000 enteros generados de forma aleatoria, variando la cantidad de threads que se crearan. Probando con 1, 2, 4, 6, 8, 10 ,12, 16 y 32. Los tiempos serán expresados en milisegundos.

Thread_count	1	2	4	6	8	10	12	16	32
Two parallel for directives	271.1	134.5	86.9	87.8	73	60.8	72	287	594.1
Two for directives	250	124.5	84.2	67	55.5	49.5	53.1	163.6	307.3

Figure 1: Tabla de tiempos en milisegundos

Podemos observar que el segundo algoritmo modificado del BubbleSort tiene un mayor desempeño a comparación del primero en todos los casos

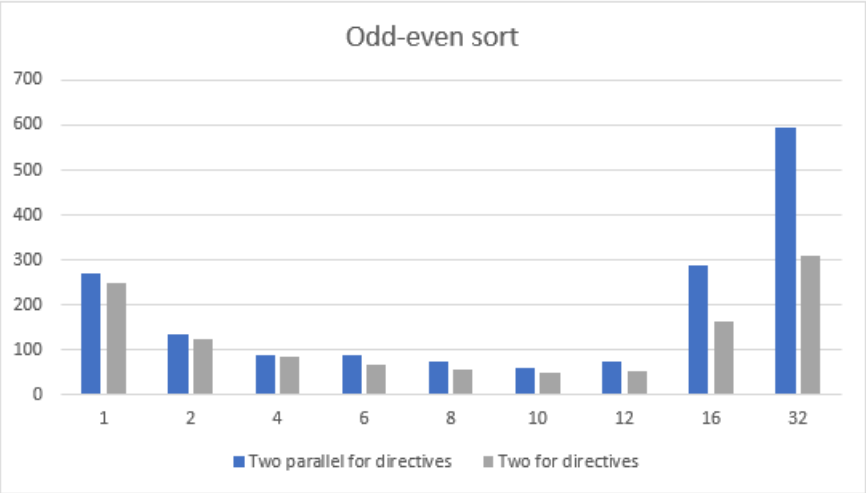

Figure 2: Grafico de Tiempos en milisegundos

Observamos también que al incrementar el numero de procesos aumenta el tiempo considerablemente esto debido a que dicho procesador solo tiene 6 cores y 12 procesadores lógicos.

4 Conclusión

La mejora en la modificación paralela es evidente y en ambos casos se observa una mejora al aumentar el numero de procesadores en uso. Y considerar que el costo de crear un thread también se ve reflejado en los tiempos por ello la diferencia de algoritmos, es necesario crear los necesarios y utilízalos para todo lo q se pueda, al igual que el overhead de crear mas procesos que cores disponibles eso solo aumentara el tiempo de ejecución. Esto ultimo lo podemos evitar consultando mediante codigo el numero de procesadores disponibles y así mandar la cantidad correcta.

5 Repositorio

https://github.com/ytuza/BubbleSortOpenMP

References

Peter Pacheco. 1997. *Parallel programming with MPI.* Morgan Kaufmann.