Informe 5 Computación Paralela y Distribuida - OpenMP

Valdivia Begazo Sharon Daniela
sharon.valdivia@ucsp.edu.pe
Universidad Católica San Pablo

1 de octubre de 2023

1. Descripción

Para esta práctica, se implementó el algoritmo **Odd-even transposition sort** en 2 versiones, la primera con dos directivas for paralelas y la segunda con 2 directivas for.

2. Especificaciones técnicas de la computadora

La ejecución del código se realizó con Ubuntu 18.04.5, con un procesador *Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz.* con las siguientes características:

- **RAM** : 8 GB
- **Caché L1** : 256 KB
- **Caché L2** : 1.0 MB
- **Caché L3** : 6.0 KB
- **Núcleos** : 4
- **Procesadores Lógicos** : 8

3. Explicación del Algoritmo

La idea general del algoritmo consiste en ir intercambiando por fases (según sean par o impar) 2 números según el ordenamiento (ascendente o descendente); en la fase par, cada 2 elementos se comparan: el número actual y el número a su izquierda, y en la fase impar, cada 2 elementos menos el último, se comparan: el número actual y el número a su derecha. El algoritmo descrito se muestra en la figura 1.

```
for (phase = 0; phase < n; phase++)
   if (phase % 2 == 0)
      for (i = 1; i < n; i += 2)
         if (a[i-1] > a[i]) Swap(&a[i-1],&a[i]);
   else
      for (i = 1; i < n-1; i += 2)
         if (a[i] > a[i+1]) Swap(&a[i], &a[i+1]);
```

Figura 1: Odd-even transposition sort

4. Análisis

4.1. Primera versión - Dos directivas for paralelas

En esta versión se presenta una barrera implícita al final del bucle (es decir por cada fase), asi que ningún thread creado podrá avanzar a la siguiente fase hasta que todos no hayan concluido su tarea.
Lo malo es que se presentan problemas al realizar fork y join, ya que los realiza en cada etapa del bucle.

4.2. Segunda versión - Dos directivas for

Por otro lado en esta versión, OpenMP nos brinda una opción para evitar este problema de fork y join, permitiéndonos reutilizar los threads ya creados, en vez de generar unos nuevos.
De igual forma trabaja con una barrera implícita, pero ahorrando más tiempo, por lo que es mejor que la primera versión.

5. Resultados

A continuación en las figuras 2, 3 y 4 se muestran los resultados de ambas versiones, con pruebas de 1 a 4 threads, para listas de 10000 a 50000 elementos.

	Primera Versión			
	1	2	3	4
10000	0.2846739	0.1378716	0.1186668	0.1093726
20000	1.0165390	0.5107890	0.4038800	0.4173265
30000	2.3383650	1.3364240	0.9658064	0.6881648
40000	5.3828490	1.8674560	1.6078000	1.5367840
50000	11.2253500	3.0797890	2.6285610	2.1875300

	Segunda Versión			
	1	2	3	4
10000	0.2393668	0.1111841	0.1034193	0.0717229
20000	0.8419112	0.4362609	0.2949944	0.2896063
30000	1.9913440	0.9039724	0.7257944	0.6316292
40000	4.0948960	1.7304580	1.3636370	1.3945730
50000	8.9840880	2.6617800	2.0934810	1.8601500

Figura 2: Tablas de resultados de ambas versiones

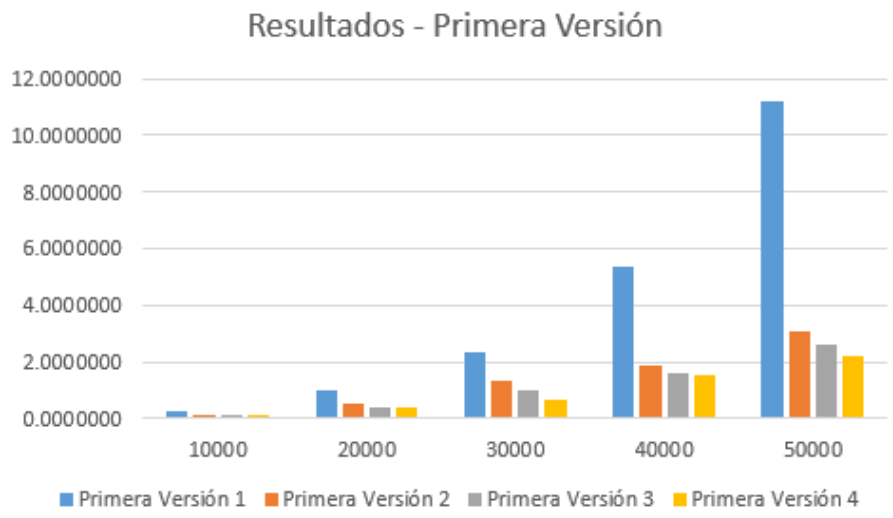

Figura 3: Resultados - Primera Versión

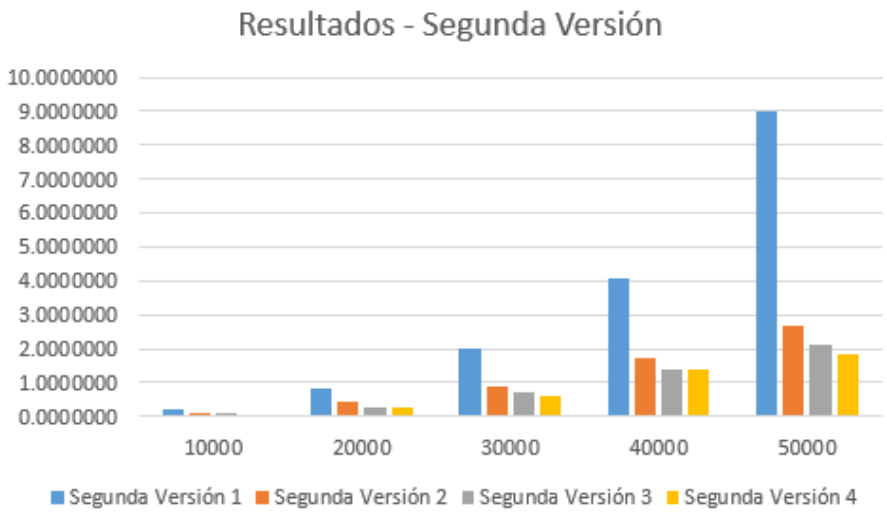

Figura 4: Resultados - Segunda Versión

Como se observa, ambas versiones presentan un mismo comportamiento, según la cantidad de threads y elementos, sin embargo, la segunda versión, muestra menores tiempos de ejecución.

6. Código Fuente

El código fuente se encuentra en: `https://github.com/valdiviasharon/Paralela/tree/main/OpenMP`