Informe de openMP

Jhoel Christian Ccari Quispe
Alumno de Ciencia de la Computación
Universidad Catolica San Pablo

I. CONSIDERACIONES PREVIAS

Algunas consideraciones que se deben tomar en cuenta son:

- El presente trabajo se **divide en dos partes**, **la primera** es un tabla obtenida de medir los tiempos de ejecución de un algoritmo de multiplicacion de una matriz por un vector y **la segunda** una tabla de medición de tiempos de ejecución del algoritmo Odd-Even Sort.
- Ambos algoritmos son paralelizados usando **openMP**.
- **Hardware**
  `Marca:` HP Probook
  `Procesador:` Intel® Core™ i5-4200M CPU
  `Velocidad de Procesador:` 2.5Ghz
  `Número de núcleos:` 4
  `Disco Duro:` 750 GB
  `Arquitectura:` 64 Bits

- **Sistema Operativo**
  `Tipo:` Fedora 27 - Linux

- **tamañi de cache**
  `L1:` 32KBytes
  `L2:` 256KBytes
  `L3 or LL:` 3MBytes

II. ODD-EVEN SORT

Este algoritmo es un algoritmo parecido al Bubble Sort pero tiene más oportunidades de paralelizar.

thread count	1	2	3	4
Two paralell for directives	0,925	0,449	0,537	0,486
Two for Directives	0,911	0,434	0,511	0,42

Figura 1. Tabla tiempos de ejecución

II-A. conclusión

Como se puede ver en el grafico y en las tablas la forma "Two parallel for directives.[es] mas lenta que la otra, y eso de debe a:

- En cada fin de un loop hay una barrera implícita, porque se necesita que todos los threads de una fase P terminen antes que cualquier thread comienze la fase P+1. Por tanto esto hace que los tiempos sean un mas lentos.

La segunda dorme "Two for directives"también es mas rápida porque:

- Al usar una directiva "parallel for.[en] el for externo OpenMP crea los threads y hace que los for interiores reutilizen los mismo threads en cada iteración.

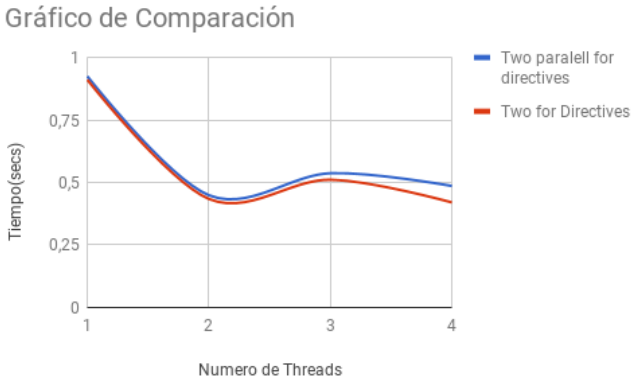

Figura 2. Gráfico de comparación de tiempos

III. MULTIPLICACIÓN MATRIZ-VECTOR

	Dimensiones de Matrices		
Threads	8000000x8	8000x8000	8x8000000
1	0,282	0,218	0,274
2	0,137	0,117	0,239
4	0,129	0,114	0,225

Figura 3. Tabla tiempos de ejecución

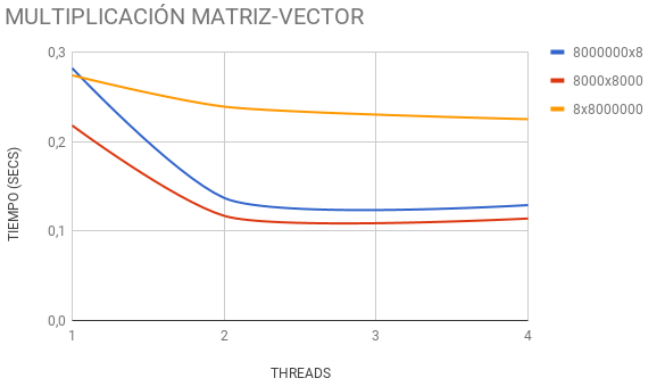

Figura 4. Gráfico de comparación de tiempos

III-A. conclusión

Los mejores tiempos se obtienen en la matriz ”8000x8000ƶ esto se debe a:

- Al uso de la caché, en específico a los principios de **localidad espacial** y **localidad temporal** de los datos al momento de realizar las operaciones.
- En las operaciones de matrices **8000000x8** se tienen muchos **write misses** (Cuando un thread trata de actualizar una variable y esta no esta en la caché y tiene que acceder a la memoria principal).
- En las operaciones de matrices de **8x8000000** se tienen muchos **read misses**(Cuando un thread trata de leer una variable y esta no esta en la caché y tiene que acceder a la memoria principal).