Sistemas paralelos 2015
Entrega 2

Integrantes:

Cuitiño, Juan Alfonso - 10581/3
Keegan, Christian Jonathan - 10694/2

Estrategias:

El problema se basa en tres partes, cada cual tiene una estrategia diferente. En términos generales, el sistema se paraleliza con cuatro threads, ya que es el hardware sobre el que se va a trabajar.

La primera parte consta de la multiplicación de tres matrices, con el detalle de que la última matriz es diagonal. Lo primero es optimizar la multiplicación por la matriz diagonal; es muy ineficiente tratarla como una matriz normal. Lo ideal es tratarla como un arreglo de N-valores y aplicar las modificaciones necesarias en la multiplicación.
En la parte paralela se trabaja con submatrices, lo que evita cualquier problema con secciones críticas ya que la escritura se da en distintas celdas.

La segunda parte consta de sacar el máximo, mínimo y promedio de las primeras dos matrices. La solución paralela trae problemas con secciones críticas, ya que varios threads/procesos pueden escribir en el máximo, mínimo y total al mismo tiempo. Los métodos se encargan de buscar un máximo y un mínimo local y compararlo con una variable compartida.

La tercer parte es el ordenamiento. El hecho de ordenar por columnas y mover toda la fila, hace que sea imposible utilizar caché en este caso. Esto además hace que cada columna sea dependiente de la anterior, así que esa parte no se puede paralelizar, lo único que queda es paralelizar el ordenamiento interno de cada columna, dividiéndola en cuatro sub-arreglos, ordenarlos y hacer un merge entre los sub-arreglos. En el caso secuencial, se copia en un arreglo todos los valores con su respectiva posición y se los ordena, con eso se aprovecha la localidad. Luego se hacen los movimientos basándose en el arreglo ordenado, lo que reduce la cantidad de movimientos en matrices grandes.

Algoritmo secuencial:

512	1024	2048
1.343129s	10.997027s	85.431948s

Paralelizado con Pthread:

	512	**1024**	**2048**
Tiempo en seg	0.424557	3.477573	26.051826
Speedup	3.16360112	3.16227064	3.27930749
Eficiencia	0.79090028	0.79056766	0.81982687

Paralelizado con OpenMP:

	512	**1024**	**2048**
Tiempo en seg	0.390424	3.213060	25.508733
Speedup	3.44018042	3.42260244	3.34912549
Eficiencia	0.8600451	0.85565061	0.83728137

Conclusión:

Observamos que con Pthreads se puede lograr un control más fino de la ejecución, aunque OMP es más amigable, ya que las herramientas están englobadas en instrucciones más fáciles de comprender.
Creemos que con Pthreads podemos lograr mucho más rendimiento si utilizamos los mismos 4 threads durante toda la ejecución, en vez de crearlos y destruirlos en cada paso. Con esos cambios el tiempo del algoritmo pthreads va a acercarse o superar al código con OMP.
También vemos que a medida que aumenta el tamaño de los datos, nuestro código de Pthreads es más rápido que el de OMP.