**TRABAJO PRÁCTICO**

**M O N I T O R E S**

**Integrantes:** Marcos Ibarra, Gonzalo Verón, Graciela Daglio

Teniendo en cuenta los cálculos que deben realizar los threads dependiendo de la cantidad de tareas a realizar y el número de orden de cada una de ellas, nos llamó la atención la variación de tiempos de ejecución modificando las variables mencionadas antes, como así también, haciendo variar el tamaño del buffer y la cantidad de latinWorkers.

**Con respecto al código implementado:**

El input se logra dando el path del archivo de entrada, y el output de la misma forma.

Recordemos que una matrix es latina si: sus filas y columnas no tienen elementos repetidos ni mayores a la dimensión de la misma.

El paso a paso para corroborar esto es:

1. Se crea una ParseMatrix con el dato a analizar, su número de identificación, la lista ordenada (OrderList) donde debe agregarse el índice si es una matrix latina, y el CountDown.
2. Se analiza si en el parseo del punto anterior, algún elemento es mayor que el size pasado.
3. Se construye una matriz a partir del dato obtenido mediante la generación de sublistas cuyo size es el orden de la matriz cuadrada.
4. Se analiza cada fila para averiguar si hay repetidos.
5. Se transpone la matriz para analizar el comportamiento en las columnas, y se repite el análisis del paso anterior.
6. Luego de obtener el resultado, si es latina, la agrega a la OrderList.
7. Decrementa el CountDown.
8. Busca la siguiente task.
9. Se escribe en el archivo output con la matríz ordenada con los índices de las matrices que son latinas.

**Con respecto los resultados obtenidos:**

Observemos la siguiente tabla (teniendo un buffer de tamaño 10) y su representación gráfica con los tiempos empleados de ejecución:

|  |  |
| --- | --- |
| cantidad latinWorkwers | tiempo (seg) |
| 1 | 6,18 |
| 2 | 7,64 |
| 3 | 8,38 |
| 4 | 6,62 |
| 5 | 7,21 |
| 6 | 6,52 |
| 7 | 7,47 |
| 8 | 6,68 |
| 9 | 9,43 |
| 10 | 8,21 |
| 11 | 9,40 |
| 12 | 7,39 |
| 13 | 7,91 |
| 14 | 7,92 |
| 15 | 7,56 |
| 16 | 8,97 |
| 17 | 6,14 |
| 18 | 9,30 |
| 19 | 7,94 |
| 20 | 8,46 |

En la siguiente tabla consideramos una cantidad de latinWorkers igual a 5, y se observa la diferencia de tiempo cuando varía el tamaño del buffer:

|  |  |
| --- | --- |
| Capacidad del buffer | tiempo(seg) |
| 1 | 7,64 |
| 2 | 10,84 |
| 3 | 9,53 |
| 4 | 8,46 |
| 5 | 7,83 |
| 6 | 7,49 |
| 7 | 7,49 |
| 8 | 6,44 |
| 9 | 9,00 |
| 10 | 6,54 |
| 11 | 6,15 |
| 12 | 6,33 |
| 13 | 8,46 |
| 14 | 9,97 |
| 15 | 7,95 |

Estas observaciones nos llevan a concluir que:

1. Considerando un buffer con tamaño de 10 y un rango de 1 a 20 threads:
   1. Cuando se consideraron 9 threads, éstos tardan más que el resto, con un tiempo de 9,43 seg..
   2. En cambio, el menor tiempo empleado se logró con 17 threads, 6,14 seg.
   3. El tiempo promedio fue de 7,77 seg. dentro de un rango entre 6,14 seg. y 9,43 seg.
   4. Teniendo en cuenta el promedio de tiempos de 7,77 seg, la cantidad óptima de latinWorkers es de 2 o 13 latinWorkers.
2. Ahora considerando la cantidad de threads fija en 5, observamos el tiempo empleado variando el tamaño del buffer, pudimos concluir que:
   1. El menor tiempo empleado fue de 6,15 seg., considerando un tamaño de buffer de 11.
   2. El máximo empleado fue de 10,84 seg considerando una capacidad de 2 en el buffer.
   3. El tiempo promedio es de 8,01 seg, en un rango entre 6,15 y 10,84 seg. Cabe destacar que este promedio es mayor con respecto a las consideraciones del primer inciso.
   4. Considerando el promedio de 8,01 seg, la cantidad óptima tendría que ser una capacidad de 15 (7,95 seg)

**NOTA:** Datos del equipo donde se realizó la toma de los tiempos necesarios para elaborar las tablas anteriores

