

Memoria de programacion concurrente y distribuida

Sistemas concurrentes y distribuidos



26 de febrero de 2017

Xabier jauregi

Universidad de Mondragón



Índice

[EL MICOPROCESADOR DE MI EQUIPO 2](#_Toc477201576)

[EJECUCION DE PROBLEMAS SECUENCIALES 2](#_Toc477201577)

[Multiplicación de matrices: 2](#_Toc477201578)

[Cálculo de números primos 3](#_Toc477201579)

[Cálculo del número pi 3](#_Toc477201580)

[Cálculo del camino más corto 3](#_Toc477201581)

[Resultados de ejecución con muro tipo 1 3](#_Toc477201582)

[Resultados de ejecución con muro tipo 2 3](#_Toc477201583)

[Difusión del calor en una malla 4](#_Toc477201584)

[EJECUCIÓN DE PROBLEMAS SINCRONIZACIÓN 4](#_Toc477201585)

[EJECUCIÓN DE PROBLEMAS PARALELIZADOS 5](#_Toc477201586)

[Lock-Atomic-Synchonized: 5](#_Toc477201587)

[Multiplicación de matrices: 5](#_Toc477201588)

[Resultados de ejecución con 2 hilos(Executor) 5](#_Toc477201589)

[Resultados de ejecución con 4 hilos(Executor) 6](#_Toc477201590)

[Resultados de ejecución con 6 hilos(Executor) 6](#_Toc477201591)

[Resultados de ejecución con 2 hilos(Runnable) 6](#_Toc477201592)

[Resultados de ejecución con 4 hilos(Runnable) 6](#_Toc477201593)

[Resultados de ejecución con 6 hilos(Runnable) 7](#_Toc477201594)

[Cálculo de números primos 7](#_Toc477201595)

[Resultados de ejecución con 2 hilos(Runnable) 7](#_Toc477201596)

[Resultados de ejecución con 4 hilos(Runnable) 7](#_Toc477201597)

[Resultados de ejecución con 6 hilos(Runnable) 7](#_Toc477201598)

[Cálculo del número pi 8](#_Toc477201599)

[Resultados de ejecución con 2 hilos(CompletionService) 8](#_Toc477201600)

[Resultados de ejecución con 4 hilos(CompletionService) 8](#_Toc477201601)

[Resultados de ejecución con 6 hilos(CompletionService) 8](#_Toc477201602)

[Resultados de ejecución con 100 hilos(CompletionService) 8](#_Toc477201603)

[Resultados de ejecución con 2 hilos(ForkJoin) 8](#_Toc477201604)

[Resultados de ejecución con 4 hilos(ForkJoin) 8](#_Toc477201605)

[Resultados de ejecución con 6 hilos(ForkJoin) 8](#_Toc477201606)

[Resultados de ejecución con 2 hilos(Thread) 9](#_Toc477201607)

[Resultados de ejecución con 4 hilos(Thread) 9](#_Toc477201608)

[Resultados de ejecución con 6 hilos(Thread) 9](#_Toc477201609)

[Cálculo del camino más corto 9](#_Toc477201610)

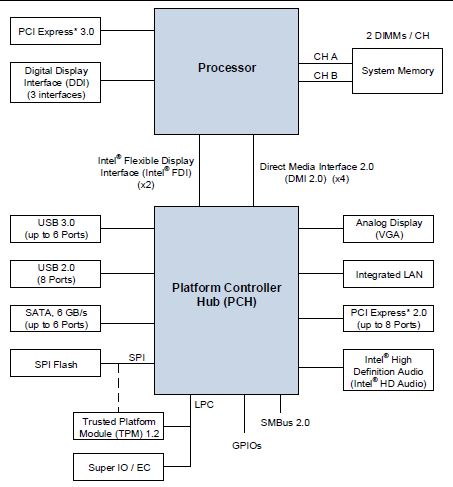
[Implementación de hilos mal hecho 9](#_Toc477201611)

[Implementación de hilos “bien” hecho 10](#_Toc477201612)

[Difusión del calor en una malla 10](#_Toc477201613)

[CONCLUSIONES 10](#_Toc477201614)

# EL MICOPROCESADOR DE MI EQUIPO

* Mi ordenador dispone de un Intel® Core™ i5-4200U que está compuesto por 4 procesadores.
* Tiene una frecuencia base de 1.60GHz y un máximo de 2.60GHz.
* Tiene 2 cores y dispone de una cache de 3MB.
* Esquema general del procesador:

# EJECUCION DE PROBLEMAS SECUENCIALES

## Multiplicación de matrices:

*A la hora de ejecutar el programa cabe destacar que tenía abierto este mismo Word y tres pestañas de Google Chrome.*

*Todos los tiempos que se muestran en la tabla de los resultados están en milisegundos y los valores obtenidos son pruebas hechas con el ordenador en la misma situación a la hora de la ejecución.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100 x 100 | 16 | 0 | 15 | 16 | 16 |
| 500 x 500 | 301 | 302 | 329 | 301 | 301 |
| 1000 x 100 | 3234 | 3149 | 3044 | 3078 | 3127 |

## Cálculo de números primos

*A la hora de ejecutar el programa cabe destacar que tenía abierto este mismo Word y tres pestañas de Google Chrome.*

*Todos los tiempos que se muestran en la tabla de los resultados están en milisegundos y los valores obtenidos son pruebas hechas con el ordenador en la misma situación a la hora de la ejecución.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 100 | 0 | 0 | 0 | 0 | 0 |
| Lim 10000 | 3 | 0 | 0 | 4 | 0 |
| Lim 100000 | 32 | 31 | 37 | 31 | 16 |

## Cálculo del número pi

*A la hora de ejecutar el programa cabe destacar que tenía abierto este mismo Word y tres pestañas de Google Chrome.*

*Todos los tiempos que se muestran en la tabla de los resultados están en milisegundos y los valores obtenidos son pruebas hechas con el ordenador en la misma situación a la hora de la ejecución.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 1620 | 1646 | 1620 | 1635 | 1636 |
| 10.000.000 ciclos | 170 | 169 | 177 | 184 | 165 |
| 10.000.000.000 ciclos | 162721 | 162596 | 162603 | 162643 | 164278 |

## Cálculo del camino más corto

A la hora de ejecutar este programa tenia abierto este documento Word el cual lo estaba editando. Y al mismo tiempo estaba ejecutando Spotify (escuchando música).

Para ver el condigo usado en estas diferentes pruebas tenéis aquí el código usado:

https://github.com/jaure96/Paralelizacion/tree/master/p4calculocaminomascorto/bin/p4calculocaminomascorto

### Resultados de ejecución con muro tipo 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Espacio 80 x80 | 2048 | 2020 | 2011 | 2065 | 2065 |
| Espacio 100 x 100 | 1829 | 1863 | 1832 | 2103 | 1896 |
| Espacio 150 x 150 | 4846 | 4846 | 4846 | 4846 | 4846 |

### Resultados de ejecución con muro tipo 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Espacio 80 x80 | 2015 | 2017 | 2080 | 2125 | 2015 |
| Espacio 100 x 100 | 1956 | 1910 | 1823 | 1849 | 1883 |
| Espacio 150 x 150 | 9921 | 9873 | 9753 | 10240 | 9810 |

## Difusión del calor en una malla

La cantidad de ejecución ha sido reducida a 3, ya que las ejecuciones de dicho programa eran largas.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| Malla 50 x50 | 143755 | 136673 | 143211 |
| Malla 100 x 100 | 607731 | 608854 | 608261 |
| Malla 150 x 150 | 1272415 | 1372723 | 1251541 |

# EJECUCIÓN DE PROBLEMAS SINCRONIZACIÓN

1. **Explica y cuantifica la ventaja de la utilización de un Thread Pool frente a la utilización de Threads. Describe el ejemplo a utilizar y proporciona los datos obtenidos.**

Una de las mayores ventajas que he podido deducir a la ora de hacer pruebas con threads simples creados a mano y threads que están dentro de un thread pool ha sido la velocidad a la hora de ejecutar programas cortos.

Una de las ventajas que nos ofrece el thread pool es que no crea thread nuevos si no que reutiliza los ya creados. Me he dado cuenta de que la creación de un thread nuevo es considerable en cuanto a rendimiento se refiere.

Por lo tanto, como conclusión crear uno o unos threads manuales considero necesario en caso de que el programa sea un programa de larga ejecución y que no necesite iniciar muchos hilos.

El thread pool digamos que esta optimizado para los programas cortos y que generan una carga grande en cuanto a la creación de hilos se refiere.

1. **¿Existen diferencias apreciables en la utilización de los distintos elementos de sincronización: monitor Object, Lock, Semaphore?**

La respuesta de esta pregunta se puede ver en la tabla de tiempos del punto posterior ya que los tiempos de ejecución obtenidos están documentados en ese apartado.

1. **Qué es mejor, hacer un objeto ThreadSafe o limitar un acceso a un hilo o un singleThreadPool**

Después de realizar varia pruebas y ver las diferencias de los tiempos de ejecución de cada programa y cada sistema diremos que la utilización de cada una de ellas depende mucho del caso en el que nos situemos.

Por ejemplo si nos centramos en el tema del ThreadPool podemos decir que es muy útil cuando la vida del programa es muy corta el mismo crea un numero de hilos muy “grande” para llevar acabo la resolución del problema. La ventaja de esta es que en vez de crear los hilos en cada ejecución reutiliza los ya creados anteriormente y que no están en funcionamiento.

El caso de limitar el acceso a un objeto por medio de locks o sistemas parecidos como semáforos en cambio depende del tiempo en el que el recurso está ocupado por el hilo que esté usándolo. Es decir, si un hilo coge la llave de un recurso, hace con él una tarea pesada y no devuelve la llave hasta terminar esa tarea estaremos prolongando mucho el tiempo de ejecución de nuestro sistema, por lo que es más óptimo poner estas exclusiones a la hora de leer y escribir en las variables compartidas.

# EJECUCIÓN DE PROBLEMAS PARALELIZADOS

## Lock-Atomic-Synchonized:

Esta demo es para casar las conclusiones de uso de los diferente tipos de locks o llaves que tenemos.

A la hora de mirar a la tabla de tiempos es importante saber que tenían ejecutando en paralelo este documento Word.

En esta prueba se han lanzado 1000000 threads que acceden a una misma variable y lo incrementan. Para ello se han utilizado diferentes métodos de sincronización y así evitar la corrupción de la variable compartida que es incrementada.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Sistema Lock | 9102 | 9415 | 9172 | 9511 | 9266 |
| Sistema Semaphore | 10958 | 9199 | 9286 | 9730 | 9264 |
| Sistema Synchronized | 9156 | 9171 | 9147 | 9379 | 10365 |
| Sistema Atomic | 9181 | 11627 | 9208 | 9167 | 9225 |

## Multiplicación de matrices:

*A la hora de ejecutar el programa cabe destacar que tenía abierto este mismo Word y tres pestañas de Google Chrome.*

*A la hora de la ejecución el ordenador estaba conectado a la fuente de alimentación el cual acelera el rendimiento del ordenador de una manera considerable.*

*Todos los tiempos que se muestran en la tabla de los resultados están en milisegundos y los valores obtenidos son pruebas hechas con el ordenador en la misma situación a la hora de la ejecución.*

### Resultados de ejecución con 2 hilos(Executor)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 500 x 500 | 478 | 388 | 399 | 462 | 441 |
| 1000 x 1000 | 4075 | 3997 | 3577 | 3459 | 4225 |
| 1500 x 1500 | 32954 | 32246 | 38025 | 32243 | 33326 |
| 2500 x 2500 | 192976 | 196124 | 196466 | 196512 | 206215 |
| 3000 x 3000 | ERROR | ERROR | ERROR | ERROR | ERROR |

### Resultados de ejecución con 4 hilos(Executor)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 500 x 500 | 301 | 281 | 297 | 313 | 328 |
| 1000 x 1000 | 5287 | 5168 | 5289 | 5164 | 5351 |
| 1500 x 1500 | 30365 | 31594 | 30277 | 30154 | 31154 |
| 2500 x 2500 | 183953 | 166764 | 142124 | 181547 | 182348 |
| 3000 x 3000 | ERROR | ERROR | ERROR | ERROR | ERROR |

### Resultados de ejecución con 6 hilos(Executor)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 500 x 500 | 278 | 305 | 280 | 298 | 358 |
| 1000 x 1000 | 5203 | 5403 | 5455 | 5189 | 5255 |
| 1500 x 1500 | 35723 | 32487 | 25866 | 32205 | 32165 |
| 2500 x 2500 | 167747 | 171423 | 167245 | 168215 | 162517 |
| 3000 x 3000 | ERROR | ERROR | ERROR | ERROR | ERROR |

### Resultados de ejecución con 2 hilos(Runnable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 500 x 500 | 217 | 218 | 216 | 200 | 201 |
| 1000 x 1000 | 1988 | 2121 | 2075 | 2051 | 2042 |
| 1500 x 1500 | 32544 | 34164 | 32335 | 32872 | 33088 |
| 3000 x 3000 | 347038 | 347697 | 341338 | 345471 | 348654 |

### Resultados de ejecución con 4 hilos(Runnable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 500 x 500 | 210 | 147 | 147 | 218 | 147 |
| 1000 x 1000 | 4606 | 5190 | 5182 | 4744 | 4593 |
| 1500 x 1500 | 26336 | 25185 | 26414 | 25514 | 25801 |
| 3000 x 3000 | 264953 | 255663 | 260090 | 259600 | 259613 |

### Resultados de ejecución con 6 hilos(Runnable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 500 x 500 | 153 | 164 | 115 | 149 | 146 |
| 1000 x 1000 | 5167 | 5119 | 5108 | 4859 | 5104 |
| 1500 x 1500 | 25287 | 25137 | 24782 | 24968 | 25674 |
| 3000 x 3000 | 240289 | 241914 | 242853 | 245258 | 243198 |

## Cálculo de números primos

*A la hora de ejecutar el programa cabe destacar que tenía abierto este mismo Word y tres pestañas de Google Chrome.*

*A la hora de la ejecución el ordenador estaba conectado a la fuente de alimentación el cual acelera el rendimiento del ordenador de una manera considerable.*

*Todos los tiempos que se muestran en la tabla de los resultados están en milisegundos y los valores obtenidos son pruebas hechas con el ordenador en la misma situación a la hora de la ejecución.*

*ejecución.*

### Resultados de ejecución con 2 hilos(Runnable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10.000 | 3 | 3 | 4 | 3 | 4 |
| Lim 100.000 | 31 | 28 | 16 | 30 | 33 |
| Lim 1.000.000 | 523 | 542 | 535 | 528 | 505 |
| Lim 100.000.000 | 202762 | 236675 | 237805 | 246883 | 24818 |

### Resultados de ejecución con 4 hilos(Runnable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10000 | 0 | 0 | 0 | 16 | 0 |
| Lim 100.000 | 16 | 16 | 15 | 11 | 16 |
| Lim 1.000.000 | 394 | 449 | 383 | 383 | 390 |
| Lim 100.000.000 | 247167 | 247511 | 246351 | 247846 | 246057 |

### Resultados de ejecución con 6 hilos(Runnable)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10.000 | 0 | 0 | 0 | 16 | 0 |
| Lim 100.000 | 32 | 28 | 37 | 31 | 32 |
| Lim 1.000.000 | 280 | 263 | 253 | 260 | 254 |
| Lim 1.00.000.000 | 151959 | 165365 | 128825 | 149738 | 161844 |

## Cálculo del número pi

### Resultados de ejecución con 2 hilos(CompletionService)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 918 | 904 | 919 | 901 | 925 |
| 1.000.000 ciclos | 16 | 16 | 16 | 16 | 15 |
| 10.000.000.000 ciclos | 91099 | 91469 | 91262 | 91356 | 90689 |

### Resultados de ejecución con 4 hilos(CompletionService)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 492 | 490 | 520 | 481 | 489 |
| 1.000.000 ciclos | 16 | 16 | 15 | 16 | 11 |
| 10.000.000.000 ciclos | 46515 | 46394 | 46607 | 48394 | 46621 |

### Resultados de ejecución con 6 hilos(CompletionService)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 568 | 505 | 505 | 507 | 521 |
| 1.000.000 ciclos | 3 | 15 | 16 | 16 | 16 |
| 10.000.000.000 ciclos | 48011 | 46973 | 48642 | 47532 | 47677 |

### Resultados de ejecución con 100 hilos(CompletionService)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 531 | 581 | 531 | 532 | 531 |
| 1.000.000 ciclos | 171 | 47 | 47 | 47 | 47 |
| 10.000.000.000 ciclos | 47660 | 46488 | 46450 | 47421 | 46689 |

### Resultados de ejecución con 2 hilos(ForkJoin)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 8404 | 8423 | 9243 | 8916 | 8720 |
| 1.000.000 ciclos | 105 | 100 | 100 | 85 | 115 |
| 10.000.000.000 ciclos | 85551 | 84778 | 84569 | 84102 | 84986 |

### Resultados de ejecución con 4 hilos(ForkJoin)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 4601 | 4979 | 4677 | 4581 | 4521 |
| 1.000.000 ciclos | 54 | 47 | 54 | 47 | 53 |
| 10.000.000.000 ciclos | 45947 | 45307 | 45437 | 45637 | 45500 |

### Resultados de ejecución con 6 hilos(ForkJoin)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 285 | 321 | 285 | 301 | 300 |
| 1.000.000 ciclos | 11 | 38 | 1 | 15 | 15 |
| 10.000.000.000 ciclos | 30377 | 30864 | 28427 | 28542 | 28533 |

### Resultados de ejecución con 2 hilos(Thread)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 933 | 938 | 934 | 937 | 933 |
| 1.000.000 ciclos | 16 | 16 | 22 | 16 | 16 |
| 10.000.000.000 ciclos | 91285 | 91220 | 91133 | 90637 | 91509 |

### Resultados de ejecución con 4 hilos(Thread)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 608 | 486 | 540 | 484 | 487 |
| 1.000.000 ciclos | 18 | 15 | 31 | 9 | 19 |
| 10.000.000.000 ciclos | 46358 | 47792 | 47169 | 47102 | 47656 |

### Resultados de ejecución con 6 hilos(Thread)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| 100.000.000 ciclos | 504 | 494 | 502 | 503 | 501 |
| 1.000.000 ciclos | 18 | 16 | 31 | 0 | 3 |
| 10.000.000.000 ciclos | 46492 | 47745 | 50161 | 47409 | 48000 |

## Cálculo del camino más corto

### Implementación de hilos mal hecho

La implementación de este programa con hilos no ha sido nada viable. He intentado meter una paralización a la hora de buscar las rutas adyacentes.

La implementación ha sido llevada a cabo de una mala manera ya que en cada ciclo de la búsqueda creo un nuevo hilo para que busque los “vecinos”. Por lo tanto se ha paralelizado nada el sistema sino que se ha ralentizado mucho más el sistema de un único hilo.

#### Resultados de ejecución con muro tipo 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Espacio 80 x80 | 4101 | 3874 | 3853 | 4015 | 3920 |
| Espacio 100 x 100 | 2719 | 2752 | 3285 | 3034 | 2939 |
| Espacio 150 x 150 | 2712 | 2939 | 3041 | 2955 | 2666 |

#### Resultados de ejecución con muro tipo 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Espacio 80 x80 | 3706 | 4107 | 4376 | 4229 | 3892 |
| Espacio 100 x 100 | 2660 | 2990 | 2930 | 3138 | 2937 |
| Espacio 150 x 150 | 2643 | 2643 | 2937 | 2905 | 2764 |

### Implementación de hilos “bien” hecho

Después de estar un buen rato intentando paralelizar el programa he llegado a la conclusión de que la paralización en este programa no tendría ventajas respecto a ejecutarlo en un hilo.

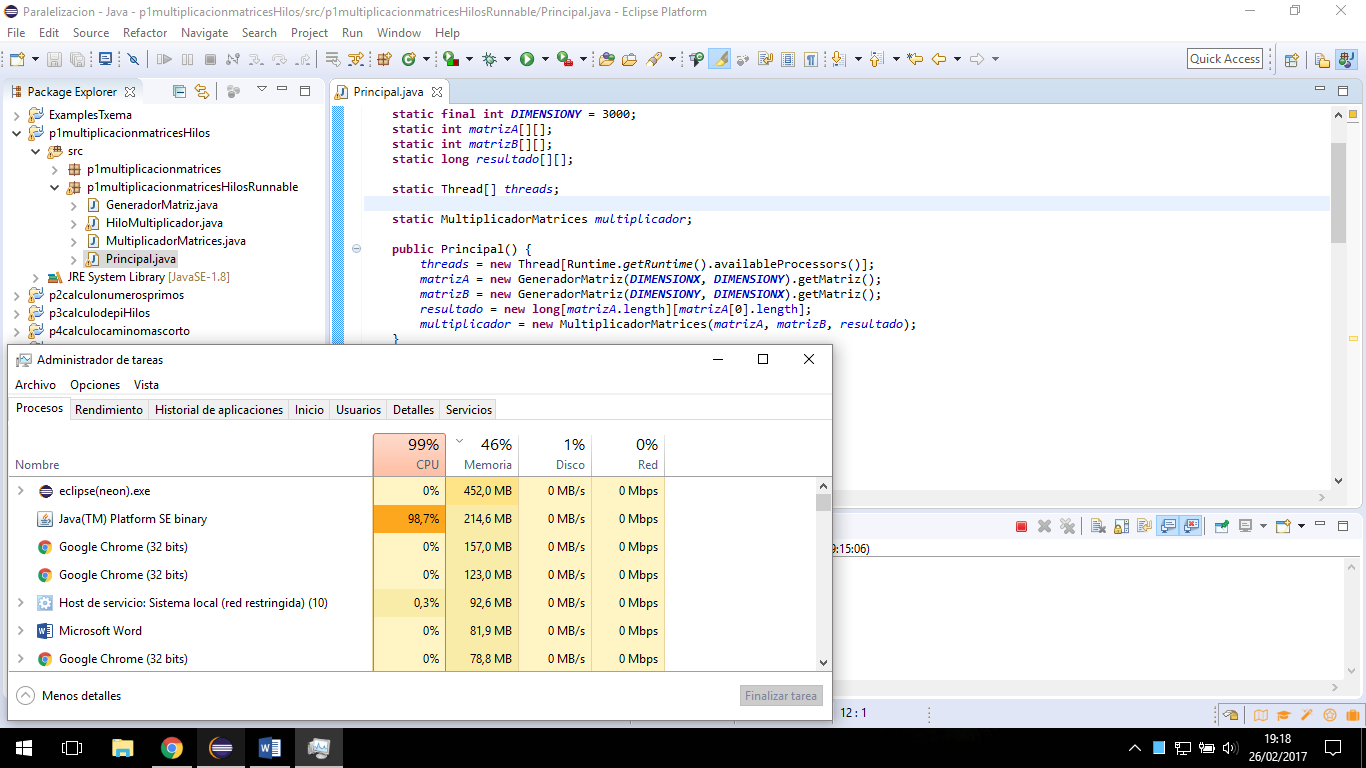
La busque del algoritmo está reflejada en el despliegue de un árbol de busque como las que se usan en inteligencia artificial. Por este motivo todos los nodos que se abren tienen que ir asta el fondo del árbol para determinar si esa ruta es buena o no, en este caso cada nodo es dependiente del nodo padre menos el nodo principal que es el estado inicial del sistema o el punto donde se inicia la búsqueda.

Esto implica que esta implementación de búsqueda necesita tener una imagen completa del árbol de búsqueda con el fin de determinar cuál es, en este caso, el camino más corto al objetivo final.

## Difusión del calor en una malla

## CONCLUSIONES

1. Es curioso ver como aumenta el uso de la CPU cuando ejecutas los programas. He ejecutado un programa con 4 hilos(numero de procesadores de mi ordenador)C y la CPU se ha disparado al 99% de su capacidad, tal y como se puede ver en la imagen posterior



También es interesante ver que cuando ejecutas dos hilos la carga del procesador baja al 50% y aumenta de una manera considerable los tiempos de ejecución.

1. Tal y como se ha hablado en uno de los puntos anteriores me he dado cuenta que la creación de los hilos es una función bastante considerable en cuanto al rendimiento del sistema se refiere. Por lo tanto, tenemos que andar con cuidado a la hora de definir el sistema de paralización que vayamos a implementar, ya que esto puede afectar al programa y en vez de conseguir acelerarlo conseguiremos ralentizarlo.
2. Se debe reflexionar y analizar muy bien el código antes de empezar a paralelizar cualquier programa, ya que la creación de los hilos en el lenguaje Java tiene un consto computacional muy elevado tal y como se ha podido observar en la tabla de tiempos del programa donde se comparaban los diferentes usos de sincronización en el programa “Lock-Atomic-Synchonized”.
3. Cuando se elaboran programas multihilos, se deben repetir las pruebas infinidad de veces para prever cualquiera condición de que pudiera alterar el resultado final bajo condiciones que no hemos tenido en cuenta. Estas condiciones excepcionales se descubren después de múltiples ejecuciones del programa con diferentes juegos de datos de entrada. En los casos que hemos probado, cada programa ha sido ejecutado por lo general cinco veces con diferentes números de hilos y diferentes tipos de datos de entrada.
4. Las ventajas de la paralización son evidentes, pero en muchas ocasiones suele ser complicado o casi imposible encontrar la manera de paralelizar los procesos dentro de una aplicación sin que el resultado final de la aplicación se vea afectado. Por lo tanto, aunque el concepto se fácil de entender el aplicarlo a un caso práctico puede ser una tarea uy complicada, tal y como se ha podido demostrar en ciertos programas de esta memoria.