

Memoria de programacion concurrente y distribuida

Sistemas concurrentes y distribuidos



26 de febrero de 2017

Xabier jauregi

Universidad de Mondragón



Índice

[1. EL MICOPROCESADOR DE MI EQUIPO 3](#_Toc477530479)

[2. EJECUCION DE PROBLEMAS SECUENCIALES 3](#_Toc477530480)

[2.1 Multiplicación de matrices: 3](#_Toc477530481)

[2.2 Cálculo de números primos 4](#_Toc477530482)

[2.3 Cálculo del número pi 4](#_Toc477530483)

[2.4 Cálculo del camino más corto 4](#_Toc477530484)

[2.4.1 Resultados de ejecución con muro tipo 1 4](#_Toc477530485)

[2.4.2 Resultados de ejecución con muro tipo 2 4](#_Toc477530486)

[2.5 Difusión del calor en una malla 5](#_Toc477530487)

[3 EJECUCIÓN DE PROBLEMAS SINCRONIZACIÓN 5](#_Toc477530488)

[4. EJECUCIÓN DE PROBLEMAS PARALELIZADOS 6](#_Toc477530489)

[4.1 Lock-Atomic-Synchonized: 6](#_Toc477530490)

[4.2 Multiplicación de matrices: 6](#_Toc477530491)

[4.2.1 Resultados de ejecución con 2 hilos(Executor) 6](#_Toc477530492)

[4.2.2 Resultados de ejecución con 4 hilos(Executor) 7](#_Toc477530493)

[4.2.3 Resultados de ejecución con 6 hilos(Executor) 7](#_Toc477530494)

[4.2.4 Resultados de ejecución con 2 hilos(Runnable) 7](#_Toc477530495)

[4.2.5 Resultados de ejecución con 4 hilos(Runnable) 7](#_Toc477530496)

[4.2.6 Resultados de ejecución con 6 hilos(Runnable) 7](#_Toc477530497)

[4.3 Cálculo de números primos 8](#_Toc477530498)

[4.3.1 Resultados de ejecución con 2 hilos(CompletionService) 8](#_Toc477530499)

[4.3.2 Resultados de ejecución con 4 hilos(CompletionService) 8](#_Toc477530500)

[4.3.3 Resultados de ejecución con 6 hilos(CompletionService) 8](#_Toc477530501)

[4.3.4 Resultados de ejecución con 100 hilos(CompletionService) 8](#_Toc477530502)

[4.3.5 Resultados de ejecución con 2 hilos(Runnable) 9](#_Toc477530503)

[4.3.6 Resultados de ejecución con 4 hilos(Runnable) 9](#_Toc477530504)

[4.3.7 Resultados de ejecución con 6 hilos(Runnable) 9](#_Toc477530505)

[4.4 Cálculo del número pi 9](#_Toc477530506)

[4.4.1 Resultados de ejecución con 2 hilos(CompletionService) 9](#_Toc477530507)

[4.4.2 Resultados de ejecución con 4 hilos(CompletionService) 9](#_Toc477530508)

[4.4.3 Resultados de ejecución con 6 hilos(CompletionService) 10](#_Toc477530509)

[4.4.4 Resultados de ejecución con 100 hilos(CompletionService) 10](#_Toc477530510)

[4.4.5 Resultados de ejecución con 2 hilos(ForkJoin) 10](#_Toc477530511)

[4.4.6 Resultados de ejecución con 4 hilos(ForkJoin) 10](#_Toc477530512)

[4.4.7 Resultados de ejecución con 6 hilos(ForkJoin) 10](#_Toc477530513)

[4.4.8 Resultados de ejecución con 2 hilos(Thread) 10](#_Toc477530514)

[4.4.9 Resultados de ejecución con 4 hilos(Thread) 11](#_Toc477530515)

[4.4.10 Resultados de ejecución con 6 hilos(Thread) 11](#_Toc477530516)

[4.5 Cálculo del camino más corto 11](#_Toc477530517)

[4.5.1 Resultados de ejecución con 2 hilos (CompletionService) 11](#_Toc477530518)

[4.5.2 Resultados de ejecución con 4 hilos (CompletionService) 11](#_Toc477530519)

[4.5.3 Resultados de ejecución con 6 hilos (CompletionService) 12](#_Toc477530520)

[4.6 Difusión del calor en una malla 12](#_Toc477530521)

[4.6.1 Resultados de ejecución con 2 hilos (CompletionService) 12](#_Toc477530522)

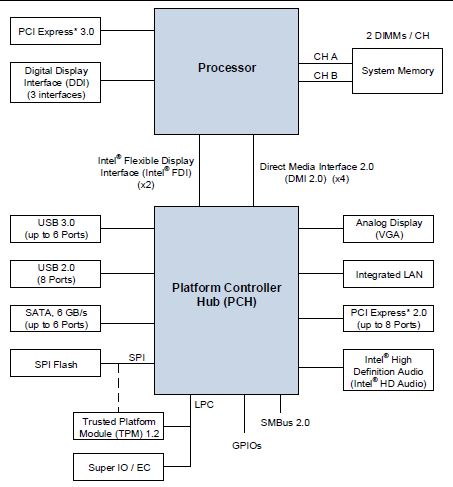
[4.6.2 Resultados de ejecución con 4 hilos (CompletionService) 12](#_Toc477530523)

[4.6.3 Resultados de ejecución con 6 hilos (CompletionService) 12](#_Toc477530524)

[5. CONCLUSIONES 13](#_Toc477530525)

[6. CODIGO DEL PROGRAMA 15](#_Toc477530526)

# 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).

### 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?**

Tomando como ejemplo la prueba que he llevado a cabo para responder a esta respuesta, no he podido apreciar una diferencia notable.  
La prueba realizada ha sido incrementar un valor por medio de 1000000 hilos donde cada hilo accedía al recurso compartido y lo incrementaba por uno.

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

Después de realizar varias 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 diferentes tipos de locks o llaves que tenemos. En definitiva, para comparar los diferentes tipos de métodos de sincronización en cuanto a tiempos de ejecucion se refiere.

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(CompletionService)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10.000 | 12 | 12 | 11 | 9 | 23 |
| Lim 100.000 | 26 | 34 | 114 | 33 | 70 |
| Lim 1.000.000 | 391 | 488 | 363 | 460 | 407 |
| Lim 100.000.000 | +15min | +15min | +15min | +15min | +15min |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10.000 | 16 | 0 | 2 | 9 | 16 |
| Lim 100.000 | 26 | 16 | 55 | 22 | 16 |
| Lim 1.000.000 | 283 | 350 | 399 | 335 | 343 |
| Lim 100.000.000 | +15min | +15min | +15min | +15min | +15min |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10.000 | 0 | 0 | 0 | 0 | 6 |
| Lim 100.000 | 41 | 15 | 16 | 41 | 29 |
| Lim 1.000.000 | 335 | 355 | 319 | 325 | 301 |
| Lim 100.000.000 | 136186 | 125052 | 125435 | 125417 | 119179 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º | Ejecución  4º | Ejecución  5º |
| Lim 10.000 | 36 | 15 | 47 | 15 | 32 |
| Lim 100.000 | 37 | 54 | 115 | 47 | 84 |
| Lim 1.000.000 | 300 | 255 | 245 | 256 | 235 |
| Lim 100.000.000 |  |  |  |  |  |

\*\*El ordenador ha dejado re responder\*\*

### 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

A la hora de ejecutar este programa para obtener los tiempos de ejecución tenia abierto y en uso Spotify y este mismo documento Word.

En esta implementación he paralelizado la función que calcula los nodos adyacentes desde un nodo y una ruta determinada. Para ello he usado el CompletionService donde una vez calculados los nodos adyacentes de cada nodo devuelvo un HasMap de la ruta y sus nodos.

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| espacio1.txt | 42578 | 44034 | 44838 |
| espacio1Muro2.txt | 47982 | 46508 | 42206 |
| espacio2.txt | 27510 | 26942 | 27475 |
| espacio2Muro2.txt | 26640 | 24546 | 27299 |
| espacio3.txt | 480716 | 450603 | 489020 |
| espacio3Muro3.txt | 482381 | 481547 | 498257 |

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| espacio1.txt | 43965 | 45519 | 44888 |
| espacio1Muro2.txt | 42235 | 45148 | 42935 |
| espacio2.txt | 25989 | 26830 | 24960 |
| espacio2Muro2.txt | 26313 | 28341 | 25491 |
| espacio3.txt | 456802 | 458102 | 459220 |
| espacio3Muro3.txt | 468768 | 468547 | 469214 |

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| espacio1.txt | 42890 | 48597 | 45634 |
| espacio1Muro2.txt | 42000 | 47671 | 44225 |
| espacio2.txt | 24651 | 26041 | 24797 |
| espacio2Muro2.txt | 27722 | 24880 | 29420 |
| espacio3.txt | 444803 | 528669 | 527219 |
| espacio3Muro3.txt | 420591 | 422165 | 425251 |

## Difusión del calor en una malla

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

Esta implementación no ha sido realizada con el “exchanger”. Lo que he hecho ha sido paralelizar la función difundir el cual el encargado de hacer el cálculo de la expansión del calor.

Usando el CompletionService he dividido la malla en filas donde cada hilo se encarga de una determinada cantidad de filas. La idea que he usado es implementar dos mallas, en el cual en cada ciclo guardo el estado de la malla anterior y hago el cambio de la difusión de calor.

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| Malla 50 x50 | 123690 |  |  |
| Malla 100 x 100 | 484885 | 492560 | 481394 |
| Malla 150 x 150 | 1084496 | 1088730 | 1090234 |

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

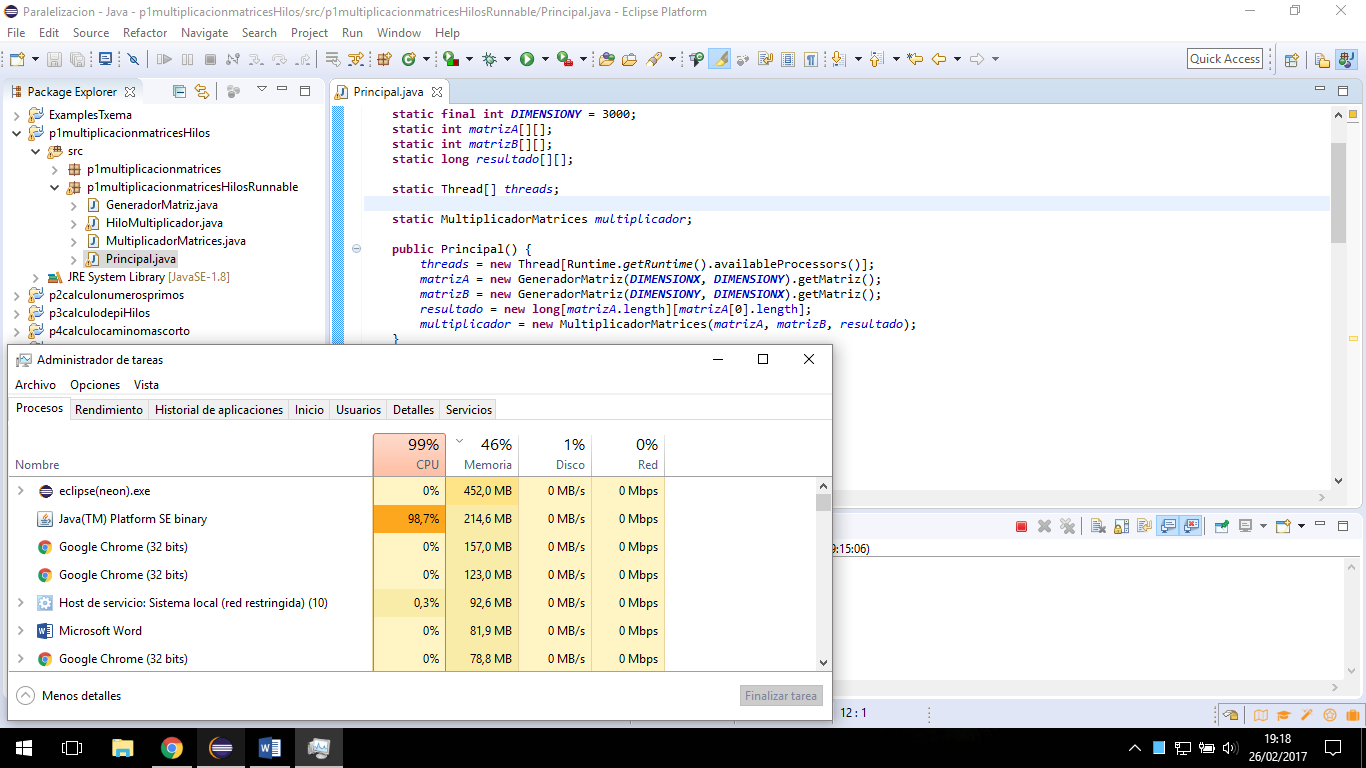
|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| Malla 50 x50 | 107388 | 100870 | 100870 |
| Malla 100 x 100 | 415859 | 415254 | 426526 |
| Malla 150 x 150 | 897147 | 891482 | 891601 |

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ejecución  1º | Ejecución  2º | Ejecución  3º |
| Malla 50 x50 | 105158 | 102309 | 104701 |
| Malla 100 x 100 | 404648 | 389977 | 384006 |
| Malla 150 x 150 | 885460 | 885460 | 1072308 |

## 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 costo 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 multi-hilos, 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. Centrándonos en el ejercicio del cálculo de los números primos en la prueba que se ha llevado a cabo con los 100 hilos en un rango de 0 a 10.000 he podido observar el peso que tiene la tarea del content switch.   
   Es decir, a la hora de dividir un trabajo pequeño, en este caso un rango muy pequeño en muchas tareas que en este caso han sido 100. Los tiempos de ejecución se han disparado por el peso que el cambio de contenido implica.
5. En el programa de la búsqueda de camino más corto, más concretamente en la prueba de paralización con CompletionService y Executor he podido observar que la paralización no siempre acelera el proceso. En este caso si nos centramos en la prueba de CompletionService he paralelizado la función para obtener los nodos adyacentes de un determinado nodo y una determinada ruta. El problema por el cual va más lento es porque cada hilo analiza su propio camino más corto y por lo tanto la creación de los nodos adyacentes tarda más en analizar que ejecutándolo en un solo hilo.
6. En el ejercicio del cálculo de Pi en la prueba donde he ejecutado el programa con 100 hilos y un rango de 100.000.000 valores el ordenador ha dejado de funcionar, se ha quedado conjelado y HE PERDIDO TODOS LOS DATOS no guardados del documento de Word.
7. 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.
8. Como conclusión del último ejercicio puedo decir que el uso de los hilos manuales no es nada rentable. Es decir, el tiempo de ejecución se dispara, ya que en cada ciclo estaríamos creando un hilo esperando a que finalice su tarea y destruyéndolo. Dicho en otras palabras, no estaríamos paralelizando nada, solo ralentizándolo.
9. En el ejercicio de la difusión de calor con la implementación de CompletionService se ha paralelizado la función difundir, el cual he creído que deterioraba mucho la velocidad del programa.  
   En este ejercicio tal y como se ha explicado en el apartado de ese ejercicio he dividido la malla en filas y no en fragmentos más pequeños(mallas más pequeñas). A cada ciclo de actualización le he enviado un referencia de la malla en el ciclo anterior y después lo he modificado, guardando el mismo para enviarlo al ciclo siguiente.  
   Creo que el sistema se podía acelerar un poco más mediante el uso de “Runnable” en vez de usar un “Callable”, ya que el uso del future veo algo innecesario.
10. ¿Cuantos mas hilos mejor? Bueno, esto depende del sistema que tengamos y como implementemos la paralización. Generalmente se dice que para poder usar la paralización con el mayor rendimiento posible tenemos que poner tantos hilos como procesadores tenga nuestro sistema.  
    Pero hay unos casos por muy pocos que sean donde poner más hilos nos puede salir rentable. Uno de esos casos es cuando nuestro sistema está a la espera de un I/O y otro es cuando un hilos espera por un tiempo prolongado a un “mutex”.  
    El caso es que si cuando todos los hilos de una aplicación están completos computacionalmente hablando, es absurdo o ineficiente pones usar más hilos que numero de procesadores tengamos.

## CODIGO DEL PROGRAMA

Todo el código generado que corresponde a los tiempos de ejecución de los programas esta disponible en GitHub:

<https://github.com/jaure96/Paralelizacion/tree/master>