

## **Programación de Servicios y Procesos**

# **Análisis del Ejercicio 2**

Ismael Delgado Rodríguez

# Propiedades de la CPU

- **Tipo de CPU**

*Mobile QuadCore Intel Core i7-2675QM, 3100 MHz (31 x 100)*

- **Alias de la CPU**

*Sandy Bridge-MB*

- **Escalonamiento de la CPU**

*D2*

- **Conjunto de instrucciones**

*x86, x86-64, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AVX, AES*

- **Reloj original**

*2200 MHz*

- **Multiplic. de CPU Mín / Máx**

*8x / 22x*

- **Engineering Sample**

*No*

- **Caché de código L1**

*32 kB per core*

- **Caché de datos L1**

*32 kB per core*

- **Caché L2**


*256 kB per core (On-Die, ECC, Full-Speed)*

- **Caché L3**

*6 MB (On-Die, ECC, Full-Speed)*

# Información física de la CPU

- **Tipo de paquete**  
*1023/1224 Ball BGA*
- **Tamaño del paquete**  
*31 mm x 24 mm*
- **Transistores**  
*1160 millón(es)*
- **Tecnología del proceso**  
*32 nm, CMOS, Cu, High-K + Metal Gate*
- **Tamaño interno**  
*216 mm<sup>2</sup>*
- **Potencia típica**  
*45 W*

CPU	Caches	Mainboard	Memory	SPD	Graphics	Bench	About
Processor							
Name	Intel Core i7 2670QM						
Code Name	Sandy Bridge	Max TDP	45.0 W				
Package	Socket 1023 FCBGA						
Technology	32 nm	Core VID	1.166 V				
Specification	Intel(R) Core(TM) i7-2675QM CPU @ 2.20GHz						
Family	6	Model	A	Stepping	7		
Ext. Family	6	Ext. Model	2A	Revision	D2		
Instructions	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX						
Clocks (Core #0)							
Core Speed	2793.33 MHz						
Multiplier	x 28.0 ( 8 - 31 )						
Bus Speed	99.76 MHz						
Rated FSB							
Cache							
L1 Data	4 x 32 KBytes			8-way			
L1 Inst.	4 x 32 KBytes			8-way			
Level 2	4 x 256 KBytes			8-way			
Level 3	6 MBytes			12-way			
Selection	Processor #1			Cores	4	Threads	8

# Pruebas de rendimiento

Las pruebas constarán de 4 tests a cada programa (secuencial y Multi Hilo). Cada uno de ellos irá aumentando de forma progresiva el tamaño de las matrices a multiplicar.

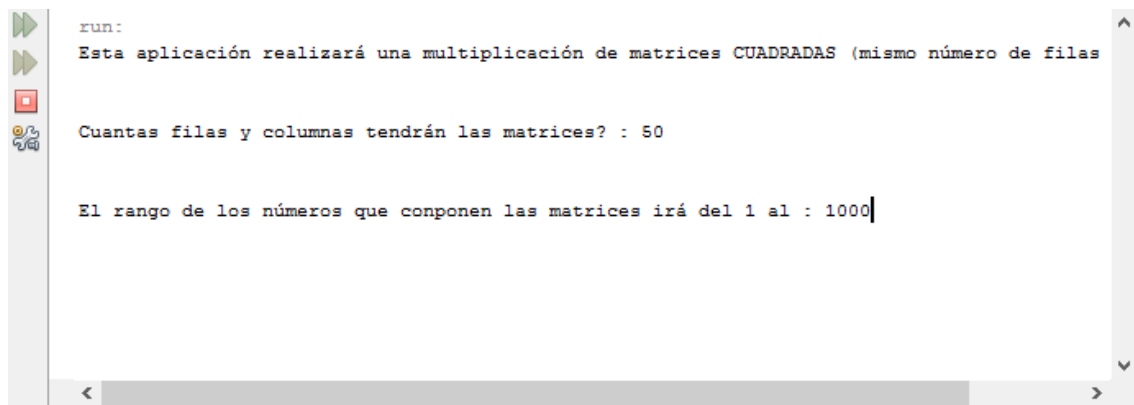
Todos los tests se han realizado en condiciones (carga soportada por el procesador) idénticas.

## SECUENCIAL

### TEST 1

Tamaño de la matriz: 50 x 50

Rango de los números que las componen: 1--->1000

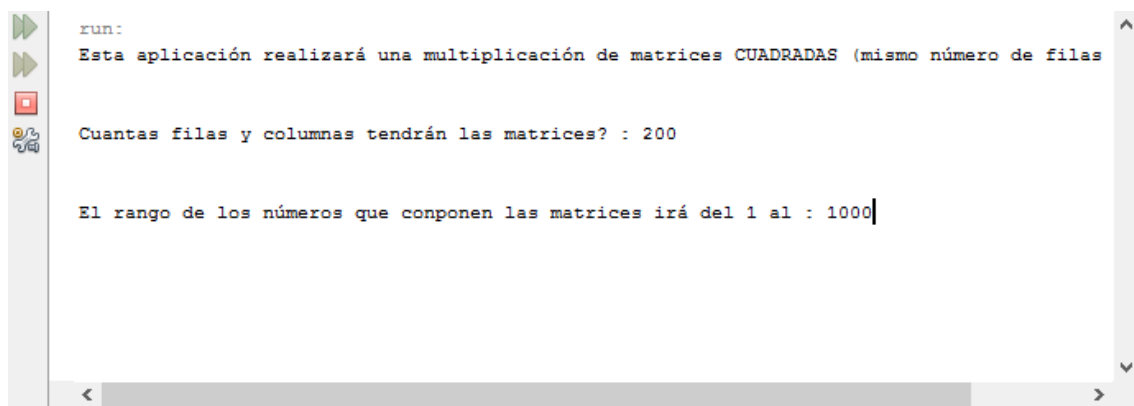


TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 16 milisegundos

### TEST 2

Tamaño de la matriz: 200 x 200

Rango de los números que las componen: 1--->1000

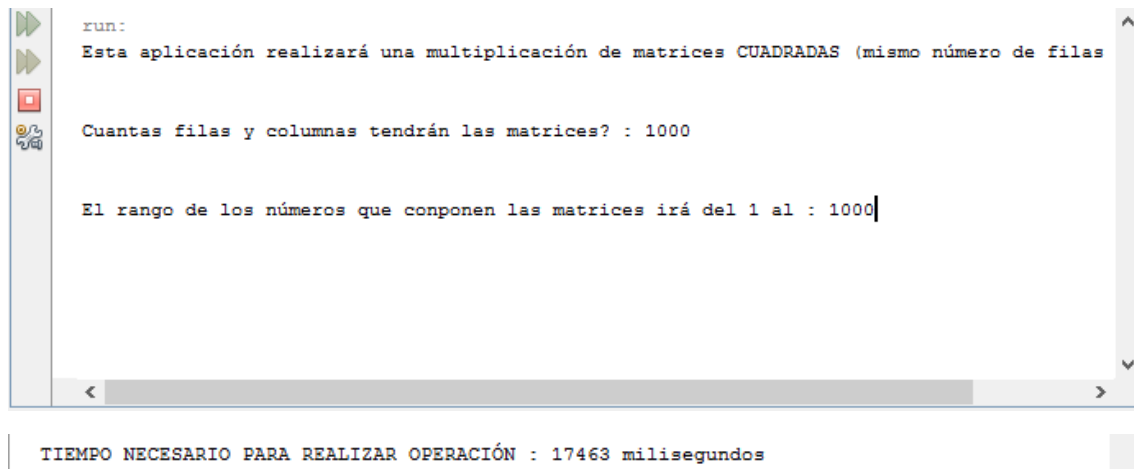


TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 328 milisegundos

### TEST 3

Tamaño de la matriz: 1000 x 1000

Rango de los números que las componen: 1--->1000



```
run:
Esta aplicación realizará una multiplicación de matrices CUADRADAS (mismo número de filas

Cuántas filas y columnas tendrán las matrices? : 1000

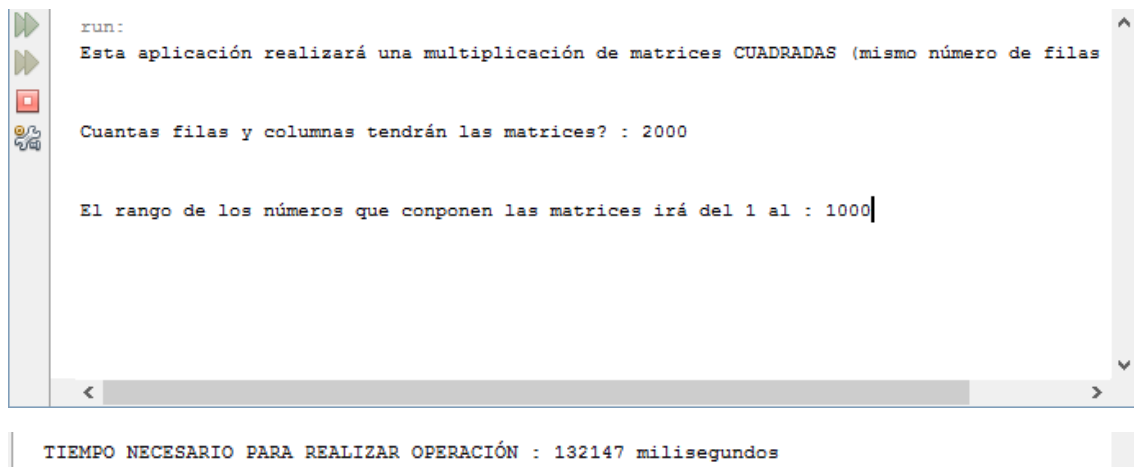
El rango de los números que componen las matrices irá del 1 al : 1000

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 17463 milisegundos
```

### TEST 4

Tamaño de la matriz: 2000 x 2000

Rango de los números que las componen: 1--->1000



```
run:
Esta aplicación realizará una multiplicación de matrices CUADRADAS (mismo número de filas

Cuántas filas y columnas tendrán las matrices? : 2000

El rango de los números que componen las matrices irá del 1 al : 1000

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 132147 milisegundos
```

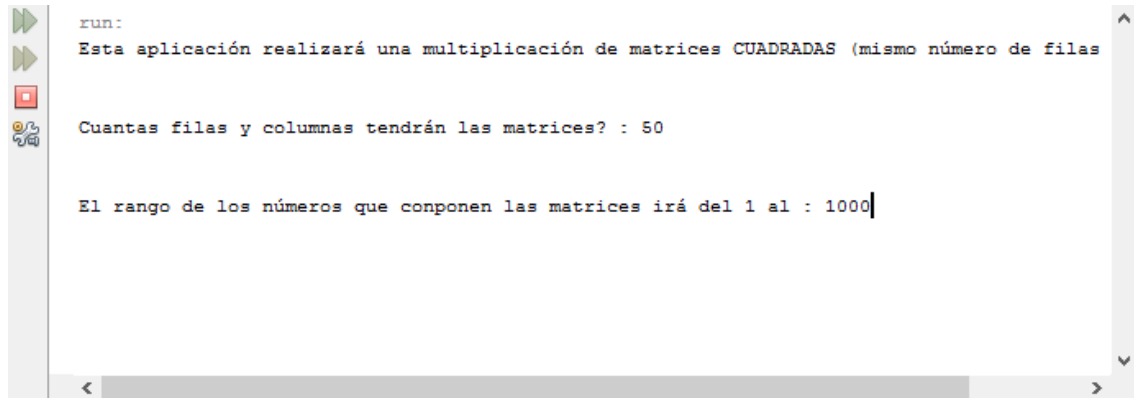
CPU # 1 / núcleo # 1 / Unidad HTT # 1 0%  
CPU # 1 / núcleo # 1 / Unidad HTT # 2 0%  
CPU # 1 / núcleo # 2 / Unidad HTT # 1 0%  
CPU # 1 / núcleo # 2 / Unidad HTT # 2 0%  
CPU # 1 / núcleo # 3 / Unidad HTT # 1 100%  
CPU # 1 / núcleo # 3 / Unidad HTT # 2 0%  
CPU # 1 / núcleo # 4 / Unidad HTT # 1 0%  
CPU # 1 / núcleo # 4 / Unidad HTT # 2 0%

## USO DE 4 THREADS

### TEST 1

Tamaño de la matriz: 50 x 50

Rango de los números que las componen: 1--->1000



```
run:
Esta aplicación realizará una multiplicación de matrices CUADRADAS (mismo número de filas
y columnas)

Cuántas filas y columnas tendrán las matrices? : 50

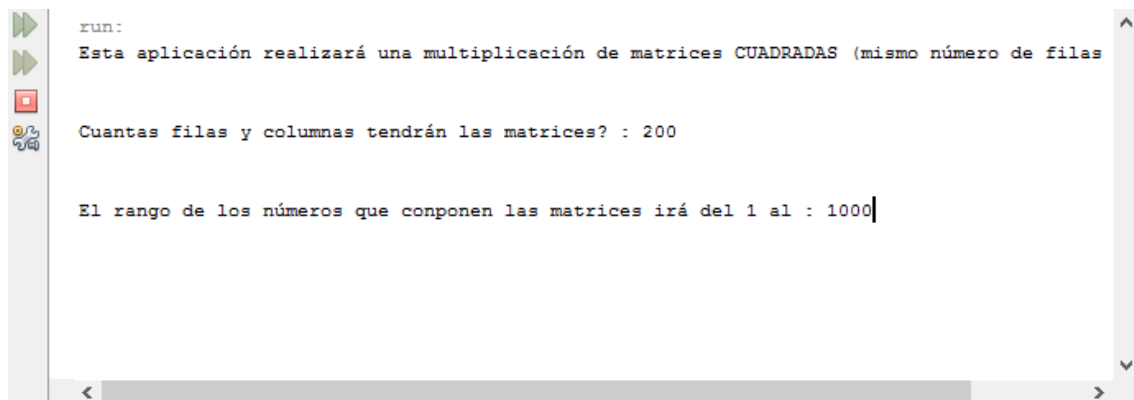
El rango de los números que componen las matrices irá del 1 al : 1000
```

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 15 milisegundos

### TEST 2

Tamaño de la matriz: 200 x 200

Rango de los números que las componen: 1--->1000



```
run:
Esta aplicación realizará una multiplicación de matrices CUADRADAS (mismo número de filas
y columnas)

Cuántas filas y columnas tendrán las matrices? : 200

El rango de los números que componen las matrices irá del 1 al : 1000
```

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 375 milisegundos

### TEST 3

Tamaño de la matriz: 1000 x 1000

Rango de los números que las componen: 1--->1000

```
run:
Esta aplicación realizará una multiplicación de matrices CUADRADAS (mismo número de filas

Cuántas filas y columnas tendrán las matrices? : 1000

El rango de los números que componen las matrices irá del 1 al : 1000|

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 9720 milisegundos
```

### TEST 4

Tamaño de la matriz: 2000 x 2000

Rango de los números que las componen: 1--->1000

```
run:
Esta aplicación realizará una multiplicación de matrices CUADRADAS (mismo número de filas

Cuántas filas y columnas tendrán las matrices? : 2000

El rango de los números que componen las matrices irá del 1 al : 1000|

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 57420 milisegundos
```

CPU # 1 / núcleo # 1 / Unidad HTT # 1 0%  
CPU # 1 / núcleo # 1 / Unidad HTT # 2 100%  
CPU # 1 / núcleo # 2 / Unidad HTT # 1 100%  
CPU # 1 / núcleo # 2 / Unidad HTT # 2 100%  
CPU # 1 / núcleo # 3 / Unidad HTT # 1 0%  
CPU # 1 / núcleo # 3 / Unidad HTT # 2 0%  
CPU # 1 / núcleo # 4 / Unidad HTT # 1 100%  
CPU # 1 / núcleo # 4 / Unidad HTT # 2 0%

## Conclusiones

*En las 2 primeras pruebas prácticamente no hay diferencia entre multiplicar las matrices de forma secuencial y hacerlo a través de 4 Hilos. Incluso hay una pequeña demora en la operación paralela que intuyo se podría atribuir al tiempo que tarda en montar la estructura de los Threads, aunque insisto en que es prácticamente imperceptible.*

### Tests 1 y 2 en Secuencial

1. TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 16 milisegundos
2. TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 328 milisegundos

### Tests 1 y 2 con 4 Threads

1. TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 15 milisegundos
2. TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 375 milisegundos

*En la 3ª ya empieza a notarse una diferencia notable a favor del programa Multiproceso, esta tarda prácticamente la mitad de tiempo en realizar la misma operación pero las matrices aún no tienen suficiente tamaño cómo para aprovechar completamente su implementación.*

### Test 3 en Secuencial

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 17463 milisegundos

### Test 3 con 4 Threads

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 9720 milisegundos

*En la prueba nº 4 ya se aprecian de forma clara las ventajas que ofrece dividir la operación en 4 partes iguales y usar 1 núcleo distinto en cada una de ellas. Realizar la operación con un sólo procesador ocupa casi 3 veces más que repartiendo el trabajo entre 4.*

### Test 4 en Secuencial

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 132147 milisegundos

### Test 4 con 4 Threads

TIEMPO NECESARIO PARA REALIZAR OPERACIÓN : 57420 milisegundos

*Por lo tanto, no resulta difícil imaginar que si seguimos incrementando el tamaño de las matrices al final llegaríamos a tardar una cuarta parte en el programa que usa 4 Threads.*