# DATOS

Las pruebas están hechas con matrices creadas al azar de dimensión 1000x1000, comprobamos lo que pasa en tres casos:

1. Proyecto extremo🡪 Creamos 1 hilo por cada celda, por lo tanto tendremos 1.000.000 de hilos para hacer la cuenta.
2. Proyecto secuencial🡪 En este caso no creamos ningún hilo.
3. Proyecto por selección de hilos🡪 Elegimos en cuantos hilos queremos dividir la matriz.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Proyecto Extremo** | | | | | |
| 30213 ms | 45545 ms | 37749 ms | 45980 ms | 33579 ms | 38495 ms |
| **Proyecto Secuencial** | | | | | |
| 1744 ms | 1808 ms | 1782 ms | 1822 ms | 3320 ms | 1924 ms |
| **Proyecto por selección de hilos** | | | | | |
| **1 hilo** | 1679 ms | 1089 ms | 1804 ms | 1723 ms | 1752 ms |
| **4 hilos** | 480 ms | 464 ms | 454 ms | 559 ms | 465 ms |
| **8 hilos** | 483 ms | 514 ms | 454 ms | 540 ms | 487 ms |
| **100 hilos** | 510 ms | 535 ms | 480 ms | 540 ms | 596 ms |
| **1000 hilos** | 519 ms | 622 ms | 711 ms | 583 ms | 592 ms |

# CONCLUSION

Como conclusión podemos asegurar para empezar que la programación concurrente es mucho más efectiva que la secuencial, pero en su justa medida, ya que en el caso extremo usamos hilos pero al ser tantísimos propicia que el programa disminuya su rendimiento en gran cantidad.

Por lo tanto, podemos asumir que todo lo que sea no más de un hilo por fila, será mejor que secuencial, y sobre todo por las pruebas he podido observar que entre 4 y 8 hilos, la ejecución obtendría el menor tiempo.