
**DEPARTAMENTO DE INGENIERÍA DE SISTEMAS**

**SISTEMAS OPERATIVOS**

**JOHN CORREDOR**

# TALLER 3
# EVALUACIÓN
# DE
# RENDIMIENTO

**SEBASTIÁN SÁNCHEZ OLAYA**
**TOMÁS OSPINA ULLOA**
**DAVID SANTIAGO RODRÍGUEZ PRIETO**
**IVÁN CORTÉS CONSTAIN**

**Link Repositorio : https://github.com/sebas0430/Taller-Rendimiento-SO**

**6 DE MAYO DE 2025**
**BOGOTÁ D.C.**

# TABLA DE CONTENIDOS

## I. INTRODUCCIÓN

Este taller tiene como objetivo aplicar los conocimientos sobre un algoritmo en serie y uno paralelo midiendo el rendimiento en distintos sistemas de cómputo. Donde para probar esto se utilizan diferentes programas para multiplicar matrices de diferentes tamaños y mediremos el tiempo de ejecución de cada uno en los distintos ambientes de cómputo, modificando los métodos y los hilos con los que se ejecutan estos programas.

## II. DOCUMENTACIÓN DETALLADA

Para este taller contamos con 4 sistemas de cómputo distintos, 3 máquinas virtuales y uno nativo, donde sus características se pueden evidenciar en las siguientes figuras:

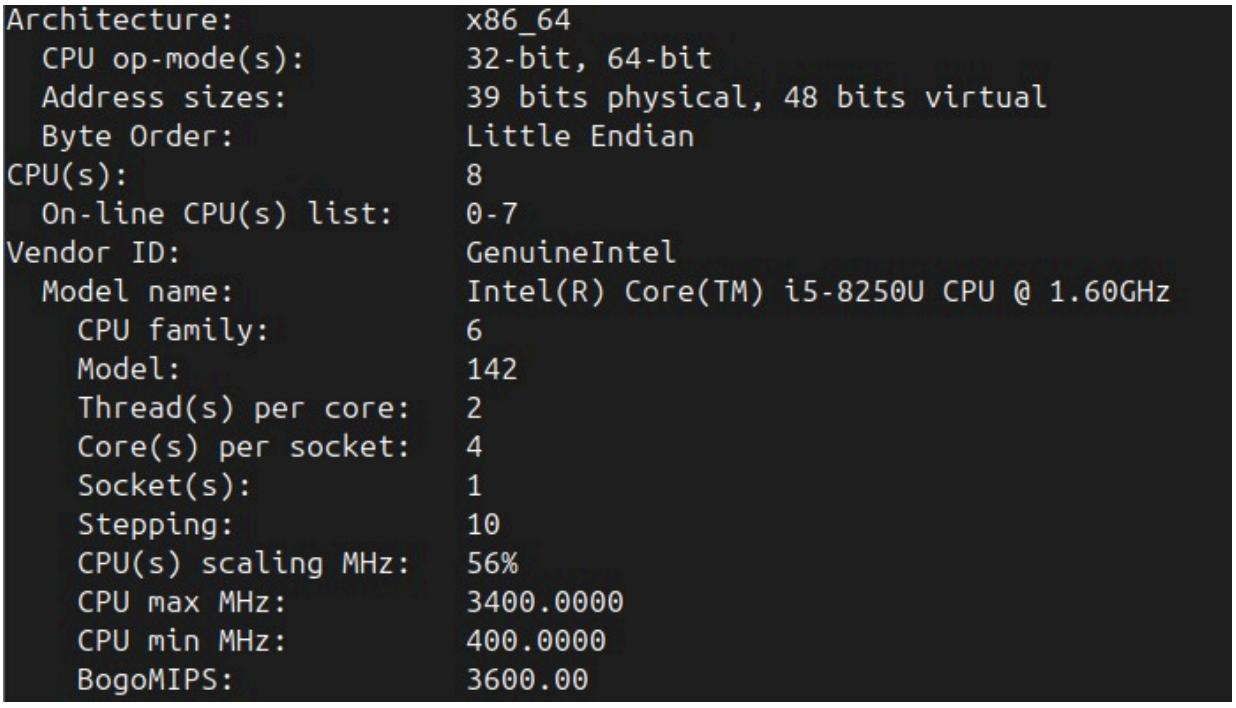


***Figura 1:*** *Especificaciones de la máquina nativa.*

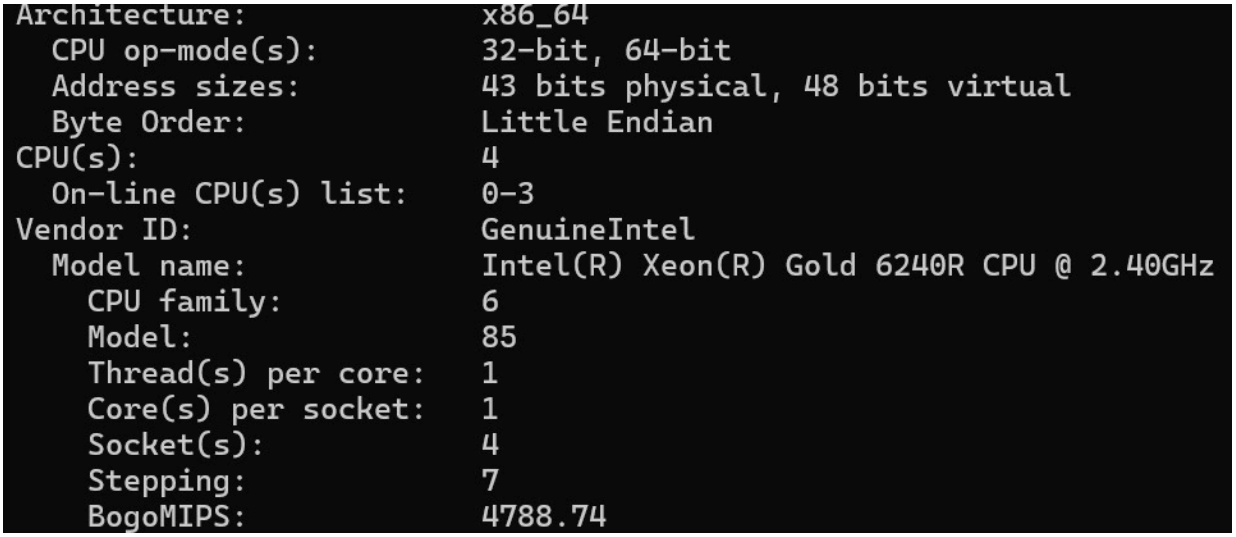


***Figura 2:*** *Rendimiento Máquinas Virtuales.*

A su vez, se trabaja con tres programas, cada uno manejando las matrices de diferentes maneras: Uno usa *fork()*, otro usa *OpenMP*, y el último usa *POSIX*. El objetivo es comparar su rendimiento con diferentes tamaños de matriz a procesar y diferentes hilos con los que se

procesa la multiplicación de matrices (ver la sección III. PLAN DE PRUEBAS para más información sobre los valores de entrada). Para ello, se utiliza un programa llamado *lanza.pl* con un lenguaje conocido como *PERL*, el cual nos permite ejecutar todas las repeticiones y pruebas de manera automática sin necesidad de ejecutarla una por una, ahorrando tiempo y recursos. Ahora, para extraer los tiempos de ejecución (en ms) de cada programa se siguen los siguientes pasos:

1. Editar el programa *lanza.pl* con los valores de entrada deseados, el programa al cual se le extraen los tiempos de ejecución y la carpeta de destino de los archivos resultantes (para efectos prácticos se cambia la extensión a .csv).
2. Ejecutar *lanza.pl* y esperar a que todas las repeticiones finalicen, cuando lo haga, aparecen 15 archivos .csv en la carpeta de destino definida.
3. Exportar cada .csv a Excel para sacar el promedio de cada una de las listas de valores para pegar ese valor resultante a una tabla que recopila los tiempos de cada manera de manejo de la matriz.
4. Repetir el proceso con las otras dos maneras restantes de manejar las matrices.

## III. PLAN DE PRUEBAS

*Resultados para las Máquinas Virtuales*

<table>
<tr><th>Descripción del Caso</th><th>Valores de Entrada</th><th>Valor Esperado</th><th>Valor Obtenido</th></tr>
<tr><td>La matriz se multiplica utilizando fork() con la variación de hilos y tamaños.</td><td rowspan="3">- Repeticiones: 30<br>- Hilos: 1 (en serie) 2,4,8,16 (en paralelo)<br>- Tamaños de Matriz: 100, 500, 1000<br><br>Los valores del tiempo en que se demora cada hilo en operar cada tamaño de matriz</td><td rowspan="3">Se espera que en cierto punto de uso de hilos se consiga el menor tiempo posible (en ms), es decir, si se hace uso excesivo de hilos, se espera que se demore más tiempo; al igual que si usamos muy pocos hilos. Requerimos los hilos necesarios para conseguir el mejor tiempo estimado.</td><td>
<table>
<tr><th colspan="4">MMCLASICAFORK</th></tr>
<tr><td>Tamaño de Matriz/ Hilos</td><td>100</td><td>500</td><td>1000</td></tr>
<tr><td>1</td><td>6225,5</td><td>574735,83</td><td>5290303,3</td></tr>
<tr><td>2</td><td>2942,56667</td><td>294355,5</td><td>2550252,83</td></tr>
<tr><td>4</td><td>2265,1</td><td>150684,73</td><td>1166550,5</td></tr>
<tr><td>8</td><td>2949,53</td><td>150663,3</td><td>1160108,13</td></tr>
<tr><td>16</td><td>3911,7</td><td>152301,267</td><td>1184698,63</td></tr>
</table>
</td></tr>
<tr><td>La matriz se multiplica utilizando comandos de OpenMP con la variación de hilos y tamaños.</td><td>
<table>
<tr><th colspan="4">MMCLASICAOPENMP</th></tr>
<tr><td>Tamaño de Matriz/ Hilos</td><td>100</td><td>500</td><td>1000</td></tr>
<tr><td>1</td><td>1138,5</td><td>159354,233</td><td>1803910,4</td></tr>
<tr><td>2</td><td>1074,23333</td><td>87966,1667</td><td>928156,7</td></tr>
<tr><td>4</td><td>3351,06667</td><td>65266,5667</td><td>480515,9</td></tr>
<tr><td>8</td><td>1659,76667</td><td>57390,1333</td><td>468064,5</td></tr>
<tr><td>16</td><td>1765,66667</td><td>54316,5667</td><td>461307,667</td></tr>
</table>
</td></tr>
<tr><td>La matriz se multiplica utilizando comandos de POSIX con la variación de hilos y tamaños.</td><td>
<table>
<tr><th colspan="4">MMCLASICAPOSIX</th></tr>
<tr><td>Tamaño de Matriz/Hilos</td><td>100</td><td>500</td><td>1000</td></tr>
<tr><td>1</td><td>5785,266667</td><td>467763,4333</td><td>3638952,967</td></tr>
<tr><td>2</td><td>3649,866667</td><td>246297,4</td><td>1841196,567</td></tr>
<tr><td>4</td><td>2840,93</td><td>147591,1667</td><td>953186,93</td></tr>
<tr><td>8</td><td>3162,166667</td><td>137849,4</td><td>924587,8667</td></tr>
<tr><td>16</td><td>3911,7</td><td>132822,1333</td><td>857654,53</td></tr>
</table>
</td></tr>
</table>

*Resultados para las Máquina Nativa*

<table>
<tr><th>Descripción del Caso</th><th>Valores de Entrada</th><th>Valor Esperado</th><th>Valor Obtenido</th></tr>
<tr><td>La matriz se multiplica utilizando fork() con la variación de hilos y tamaños.</td><td rowspan="3">- Repeticiones: 30<br>- Hilos: 1 (en serie) 2,4,8,16 (en paralelo)<br>- Tamaños de Matriz: 100, 500, 1000<br><br>Los valores del tiempo en que se demora cada hilo en operar cada tamaño de matriz</td><td rowspan="3">Se espera que en cierto punto de uso de hilos se consiga el menor tiempo posible (en ms), es decir, si se hace uso excesivo de hilos, se espera que se demore más tiempo; al igual que si usamos muy pocos hilos. Requerimos los hilos necesarios para conseguir el mejor tiempo estimado.</td><td>FORK<br><table><tr><th>Numero de Hilos / Tamaño Matriz</th><th>100</th><th>500</th><th>1000</th></tr><tr><td>1</td><td>4883.067</td><td>586774.2</td><td>7188713</td></tr><tr><td>2</td><td>2901.733</td><td>381212.8</td><td>3605284</td></tr><tr><td>4</td><td>2227.3</td><td>264172.3</td><td>2659385</td></tr><tr><td>8</td><td>2403.933</td><td>363162.5</td><td>3592917</td></tr><tr><td>16</td><td>2762.967</td><td>368588.667</td><td>348760</td></tr></table></td></tr>
<tr><td>La matriz se multiplica utilizando comandos de OpenMP con la variación de hilos y tamaños.</td><td>OpenMP<br><table><tr><th>Numero de Hilos / Tamaño Matriz</th><th>100</th><th>500</th><th>1000</th></tr><tr><td>1</td><td>1081</td><td>157679.6</td><td>3688459</td></tr><tr><td>2</td><td>611.4</td><td>80412.8</td><td>2083616</td></tr><tr><td>4</td><td>456.6</td><td>150373</td><td>2271888</td></tr><tr><td>8</td><td>5877.633</td><td>154289.5</td><td>2160792</td></tr><tr><td>16</td><td>1109.8</td><td>119480.9</td><td>2129080</td></tr></table></td></tr>
<tr><td>La matriz se multiplica utilizando comandos de POSIX con la variación de hilos y tamaños.</td><td>Posix<br><table><tr><th>Numero de Hilos / Tamaño Matriz</th><th>100</th><th>500</th><th>1000</th></tr><tr><td>1</td><td>4480.633</td><td>534772.4</td><td>6350488</td></tr><tr><td>2</td><td>2619.1</td><td>46500.7</td><td>4756349</td></tr><tr><td>4</td><td>1658.1</td><td>292356.4</td><td>2565656</td></tr><tr><td>8</td><td>1904.067</td><td>264610.2</td><td>2257211</td></tr><tr><td>16</td><td>1906.867</td><td>245306.4</td><td>2533589</td></tr></table></td></tr>
</table>

## IV. CONCLUSIONES

Para concluir, se observó que los algoritmos paralelos, al hacer uso de múltiples hilos, mejoran considerablemente los tiempos de ejecución en comparación con los algoritmos en serie, especialmente en matrices de mayor tamaño. Sin embargo, también se evidenció que existe un punto óptimo en la cantidad de hilos utilizados, ya que un número excesivo puede generar sobrecarga en el sistema y disminuir el rendimiento. Por el contrario, el algoritmo en serie presenta un comportamiento más constante pero significativamente más lento a medida que aumenta la carga de trabajo.

La elección de los tamaños de matriz (100, 500 y 1000) se hizo para representar distintos niveles de carga computacional: pequeña, media y alta, respectivamente. Esto permite evaluar cómo escalan los algoritmos en diferentes escenarios. Por su parte, el uso de 1, 2, 4, 8 y 16 hilos permite analizar el impacto del paralelismo progresivo, desde una ejecución secuencial (1 hilo) hasta una altamente concurrente (16 hilos), y así determinar el número de hilos que ofrece el mejor rendimiento sin incurrir en sobrecarga innecesaria.

## V. REFERENCIAS


- Apuntes y clase de Sistemas Operativos - Prof. John Corredor