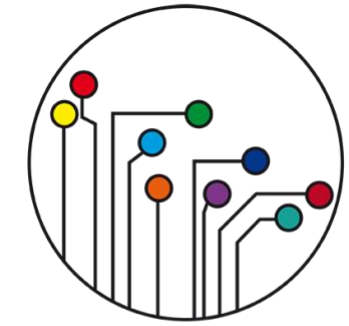
# Computación de Altas Prestaciones

Informe de Prácticas





Máster Universitario en Ingeniería Informática

Curso 2020/2021

Luis Blázquez Miñambres
Miguel Cabezas Puerto
Francisco Pinto Santos

# Contenido

# Tabla de ilustraciones

# Índice de tablas

# 1. Introducción

En este informe se detallarán los análisis referentes a las distintas implementaciones de las prácticas propuestas en la asignatura “Computación de Altas Prestaciones”.

La práctica que se ha pretendido implementar se hará haciendo uso de distintos paradigmas y bibliotecas de funcionamiento en paralelo mediante el lenguaje de programación C, en concreto: Pthreads, MPI y OpenMP.

El objetivo de la práctica es un algoritmo de simulación de propagación de calor en una plancha metálica en el que , haciendo uso de las bibliotecas descritas anteriormente, se calculará la temperatura de un punto de la plancha en un determinado instante en función de la temperatura en un instante anterior.

Para la realización de la práctica y , con el fin de discernir un análisis más detallado y comparativo entre las distintas técnicas utilizadas de paralelización, se ha ejecutado el programa con cada una de las bibliotecas en un ordenador personal en local, por un lado, y por otro lado en la máquina en remoto habilitada por los profesores de la asignatura para su uso en la red dentro de la universidad. Esto con el fin de , como se ha mencionado, desglosar una comparativa que refleje de forma mucho más detallada la variación de distintas métricas y parámetros en función de la máquina sobre la que se lanza el programa.

# 2. Implementación con hilos (Pthreads)

## 2.1. Análisis en máquina local

Dentro del análisis realizado del programa mediante el uso de la biblioteca Pthreads se pueden observar un resumen de los datos extraídos en la Tabla 1 y en la Tabla 2. Se ha partido del análisis de cuatro casos bases con 1, 5, 10 y 20 hilos variando el tamaño de la matriz debido a la gran extensión de los datos en el archivo encolumnado extraído como resultado en la ejecución del programa. Esto se ha llevado a cabo haciendo uso de un ordenador personal con las siguientes especificaciones:

- 16 GB RAM
- Intel® Core ™ i7-7800X CPU 3.50 GHz, 6 procesadores principales

Tabla 1: Resultados del análisis con Pthreads en máquina local usando división por columnas

| ***Nº hilos*** | **Tamaño Matriz** | **Nº iteraciones** | **Tiempo de ejecución (ms)** | **Speedup** | **Eficiencia (%)** | **Coste** |
|---|---|---|---|---|---|---|
| 1 | 100x100 | 1000 | 798998 | 1 | 100 | 0 |
| 5 | 100x100 | 1000 | 711404 | 1,12312835 | 22,462567 | 3557020 |
| 10 | 100x100 | 1000 | 1030296 | 0,77550335 | 7,7550335 | 10302960 |
| 20 | 100x100 | 1000 | 2065194 | 0,38688762 | 1,93443812 | 41303880 |
| 1 | 500x500 | 1000 | 14420794 | 1 | 100 | 0 |
| 5 | 500x500 | 1000 | 4970013 | 2,90156062 | 58,0312124 | 24850065 |
| 10 | 500x500 | 1000 | 5249200 | 2,74723653 | 27,4723653 | 52492000 |
| 20 | 500x500 | 1000 | 4522744 | 3,18850547 | 15,9425274 | 90454880 |
| 1 | 1000x1000 | 1000 | 54387137 | 1 | 100 | 0 |
| 5 | 1000x1000 | 1000 | 18314904 | 2,96955621 | 59,3911243 | 91574520 |
| 10 | 1000x1000 | 1000 | 17887320 | 3,0405414 | 30,405414 | 178873200 |
| 20 | 1000x1000 | 1000 | 15437787 | 3,52298791 | 17,6149396 | 308755740 |

Tabla 2: Resultados del análisis con Pthreads en máquina local usando división por filas

| **Nº hilos** | **Tamaño Matriz** | **Nº iteraciones** | **Tiempo de ejecución (ms)** | **Speedup** | **Eficiencia (%)** | **Coste** |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 811909 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 583264 | 1,39200945 | 27,840189 | 2916320 |
| **10** | 100x100 | 1000 | 1024981 | 0,79212102 | 7,92121025 | 10249810 |
| **20** | 100x100 | 1000 | 1967152 | 0,41273323 | 2,06366615 | 39343040 |
| **1** | 500x500 | 1000 | 14668777 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 4956000 | 2,95980165 | 59,1960331 | 24780000 |
| **10** | 500x500 | 1000 | 4607048 | 3,18398614 | 31,8398614 | 46070480 |
| **20** | 500x500 | 1000 | 4462684 | 3,28698537 | 16,4349268 | 89253680 |

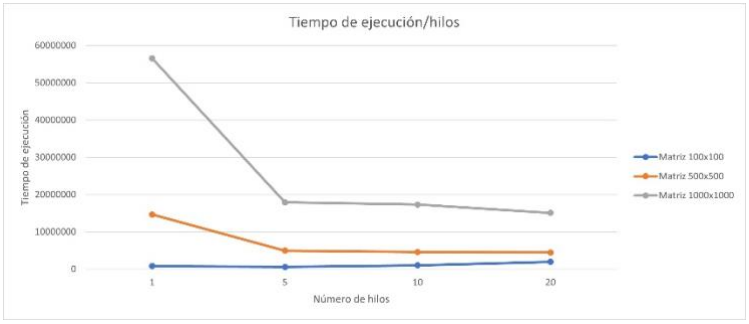
| | | | | | | |
|---|---|---|---|---|---|---|
| **1** | 1000x1000 | 1000 | 56603594 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 17916521 | 3,15929605 | 63,185921 | 89582605 |
| **10** | 1000x1000 | 1000 | 17335269 | 3,26522732 | 32,6522732 | 173352690 |
| **20** | 1000x1000 | 1000 | 15107699 | 3,74667208 | 18,7333604 | 302153980 |

En cuanto a la tendencia del programa, haciendo uso de todo el conjunto obtenidos en varias ejecuciones del programa con distintos parámetros se puede observar las ilustraciones 1-4 para las métricas haciendo uso de la división por columnas y las ilustraciones 5-8 para las métricas haciendo uso de la división por filas. En esta ilustraciones proporcionadas se puede observar que en todas existe una tendencia, esto se ha comprobado manteniendo constante el resto de los parámetros y variando para cada caso base el tamaño de la matriz manteniendo el resto de los parámetros constantes.

Así se pueden observar varios detalles en cuanto a la división por columnas y al aumento del tamaño de la matriz que representa la plancha:

- El tiempo de cálculo decrece a medida que aumenta el número de hilos ya que, a más hilos trabajando sobre la plancha, tardarán menos en realizar las tareas ya que estas estarán distribuidas entre los distintos hilos.
- En cuanto al speedup, se puede ver que, a mayor número de hilos , mayor es la tendencia del speedup porque aumentamos el número de recursos. Y ante un aumento de prestaciones, mayor productividad (ancho de banda) y menor tiempo de ejecución para la realización de cada tarea.
- En cuanto a la eficiencia, como se puede ver, con un solo proceso la eficiencia es del 100 % puesto que , al haber un solo hilo, aprovecha todo el tiempo que tiene realizando las tareas útiles ya que no tiene que comunicarse con otros hilos. Esto ocurre, a medida que aumenta el número de hilos, donde a más hilos, menor es la eficiencia dado que la comunicación en la paralelización impide que aprovechen todo el tiempo de ejecución del programa en realizar los cálculos del algoritmo sino también en comunicarse entre ellos.
- Por último, el coste computacional supone un valor trivial puesto que, ante un aumento del número de hilos , el coste computacional en llevar a cabo las tareas es mayor ya que cada hilo requiere un tiempo de CPU para sí mismo que se tiene que repartir entre los hilos disponibles.

En cuanto a la división por filas, que se puede ver a continuación junto con las gráficas de división por columnas simplemente destacar que el tiempo de cálculo , a medida que aumenta el número de hilos, decrece más progresivamente y tarda más que la división por columnas. Lo que se traduce en menores valores de eficiencia y por tanto un mayor coste , computacionalmente hablando.



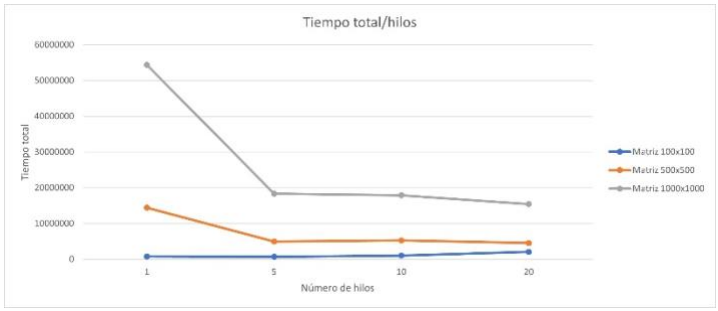




Ilustración 1: Gráfica de comparación del tiempo de ejecución del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)

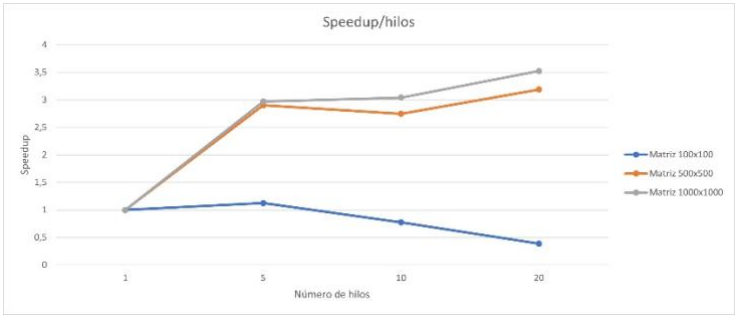
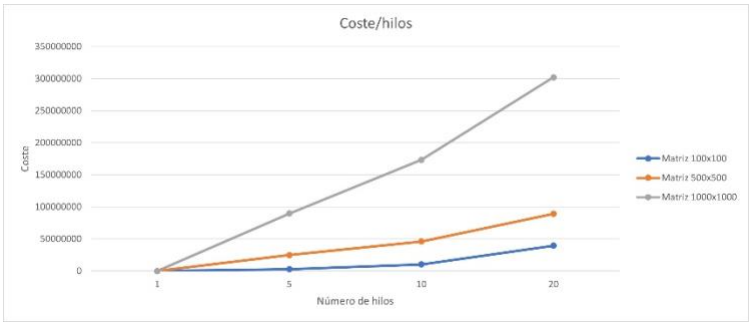




Ilustración 2: Gráfica de comparación del speedup frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)



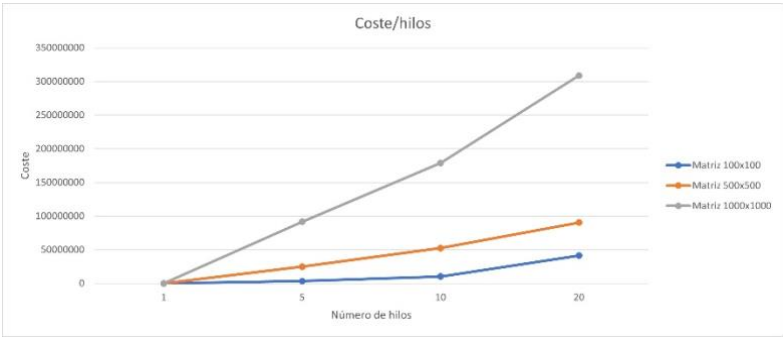
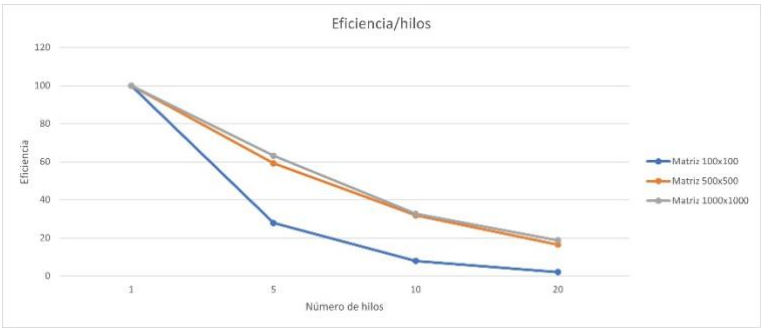




Ilustración 3: Gráfica de comparación del coste frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)



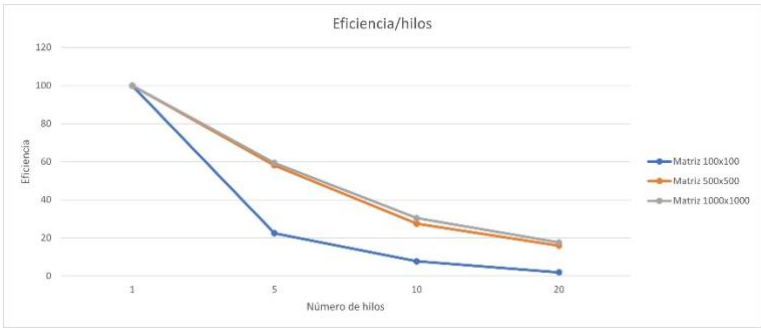




Ilustración 4: Gráfica de comparación de la eficiencia frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha))

## 2.2. Análisis en máquina remota vdia3.fis.usal.es

Dentro del análisis realizado del programa mediante el uso de la biblioteca Pthreads se puede observar un resumen de los datos extraídos en la Tabla 3 para columnas y en la Tabla 4 para filas . En este caso se ha llevado a cabo haciendo uso del servidor vdia3.fis.usal.es del departamento de Informática y Automática de la Universidad de Salamanca.

Tabla 3: Resultados del análisis con Pthreads en máquina en remoto usando división por columnas

| Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (ms) | Speedup | Eficiencia (%) | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 1846609 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 590163 | 3,14163823 | 62,8327645 | 2930 |
| **10** | 100x100 | 1000 | 458311 | 4,0372807 | 40,372807 | 4560 |
| **20** | 100x100 | 1000 | 616057 | 2,99837134 | 14,9918567 | 12280 |
| **1** | 500x500 | 1000 | 16529495 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 9828764 | 1,68217449 | 33,6434898 | 49115 |
| **10** | 500x500 | 1000 | 5120697 | 3,23049853 | 32,3049853 | 51150 |
| **20** | 500x500 | 1000 | 2277269 | 7,26649077 | 36,3324538 | 45480 |
| **1** | 1000x1000 | 1000 | 60559329 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 23824674 | 2,54225618 | 50,8451236 | 119095 |
| **10** | 1000x1000 | 1000 | 15990255 | 3,78817642 | 37,8817642 | 159850 |
| **20** | 1000x1000 | 1000 | 8083928 | 7,49337953 | 37,4668977 | 161620 |

Tabla 4: Resultados del análisis con Pthreads en máquina en remoto usando división por filas

| Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (ms) | Speedup | Eficiencia (%) | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 1772789 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 507928 | 3,5009901 | 70,019802 | 2525 |
| **10** | 100x100 | 1000 | 426165 | 4,16981132 | 41,6981132 | 4240 |
| **20** | 100x100 | 1000 | 597571 | 2,96644295 | 14,8322148 | 11920 |
| **1** | 500x500 | 1000 | 16346293 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 9506097 | 1,71992422 | 34,3984844 | 47505 |
| **10** | 500x500 | 1000 | 4797254 | 3,41005843 | 34,1005843 | 47920 |
| **20** | 500x500 | 1000 | 1987709 | 8,23224181 | 41,1612091 | 39700 |
| **1** | 1000x1000 | 1000 | 60014528 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 22212445 | 2,70225605 | 54,0451209 | 111035 |
| **10** | 1000x1000 | 1000 | 15776537 | 3,80502188 | 38,0502188 | 157710 |
| **20** | 1000x1000 | 1000 | 6972590 | 8,61084804 | 43,0542402 | 139380 |

El análisis resulta análogo al realizado por columnas en la máquina local, sin embargo, podemos observar que el tiempo de cálculo, sobre todo a medida que aumenta el tamaño de

la matriz, resulta mayor para esta técnica de división por filas que aquella que dividía la matriz por columnas explicada previamente.

Existe una tendencia descendente en cada uno de los tamaños de las matrices a medida que aumenta el número de hilos. En matrices pequeñas (100x100 y 500x500) la diferencia entre 5 , 10 y 20 hilos no es significativa ya que cada uno no realiza grandes tareas. No obstante, a medida que aumenta el tamaño de la matriz podemos observar que la diferencia al aumentar los hilos comienza a crecer, reduciéndose en gran medida los tiempo de cálculo empleados. Sin embargo, la diferencia entre 10 y 20 hilos (en una matriz grande como la mostrada de 1000x1000) es poco significativa siendo incluso mayor el tiempo que emplean más hilos (20) que disponiendo de menos hilos (10), en definitiva, se estabilizan los tiempos. Además, como es normal, los tiempos aumentan a medida que aumenta el tamaño de la matriz ya que se requieren más cálculos al existir más puntos. Es por ello que, como se ve en las siguientes gráficas, una disminución del tiempo de cálculo hace que aumente el speedup, disminuya la eficiencia y aumente el coste.

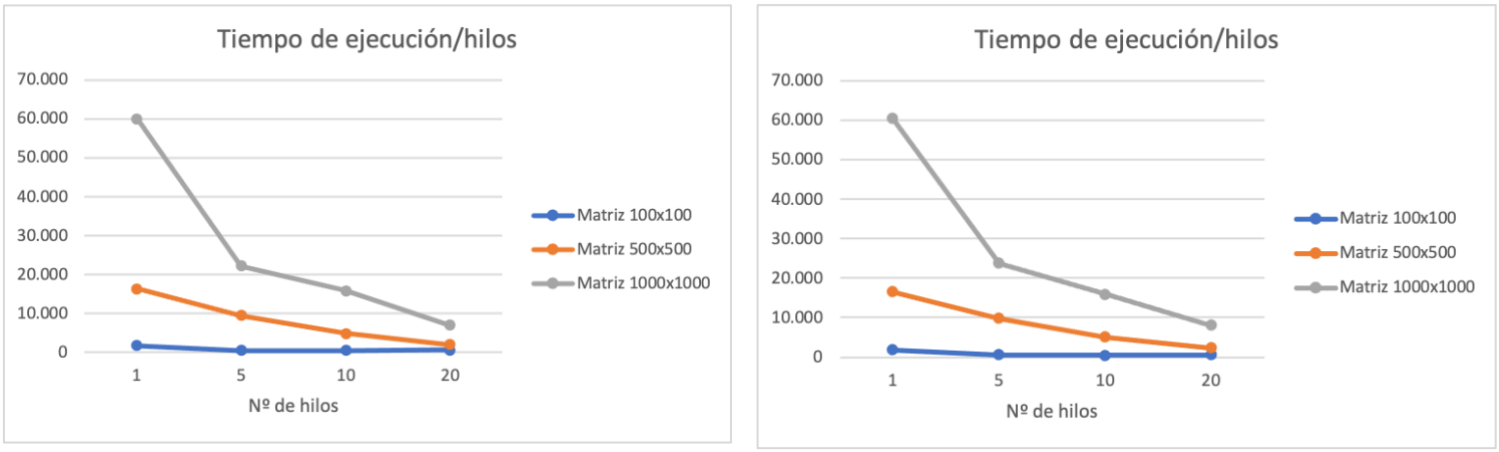



Ilustración 6: Gráfica de comparación del tiempo de ejecución del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)

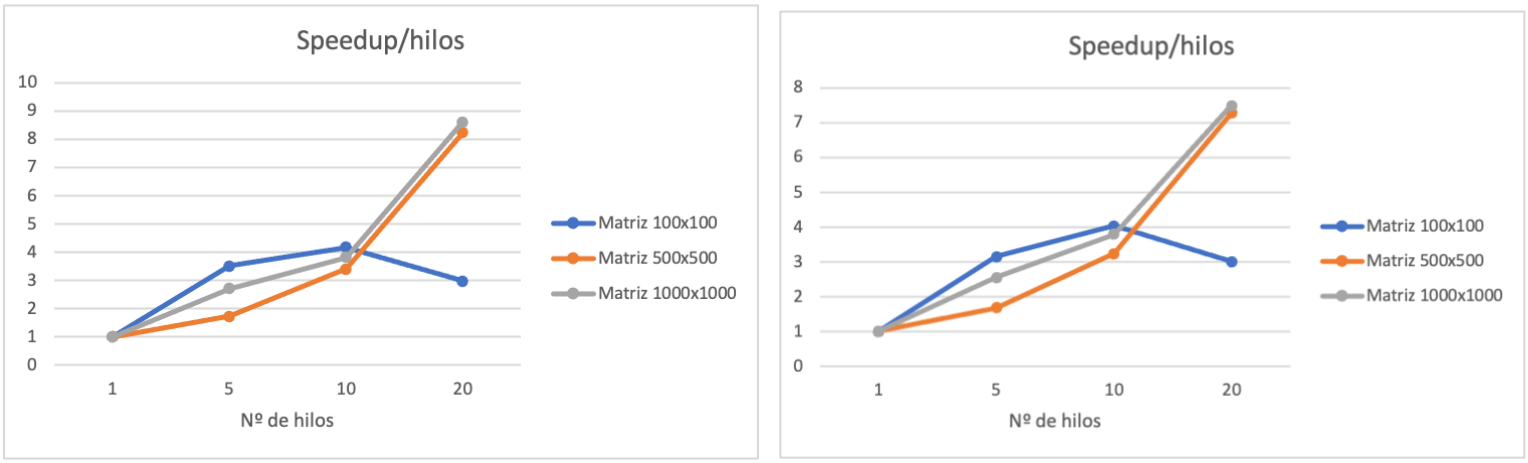
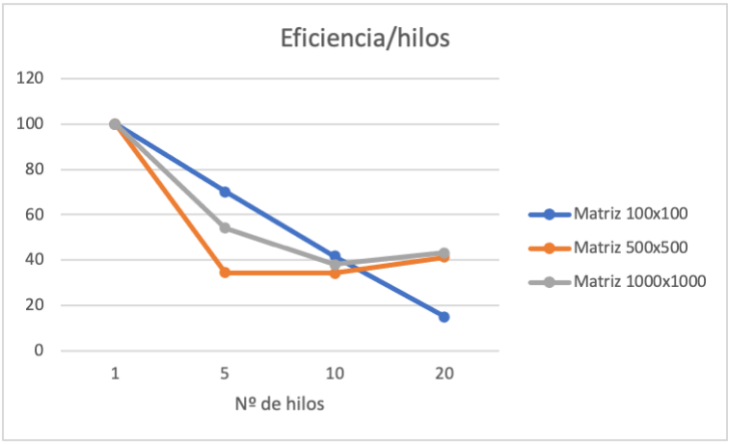



Ilustración 5: Gráfica de comparación del speedup frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)



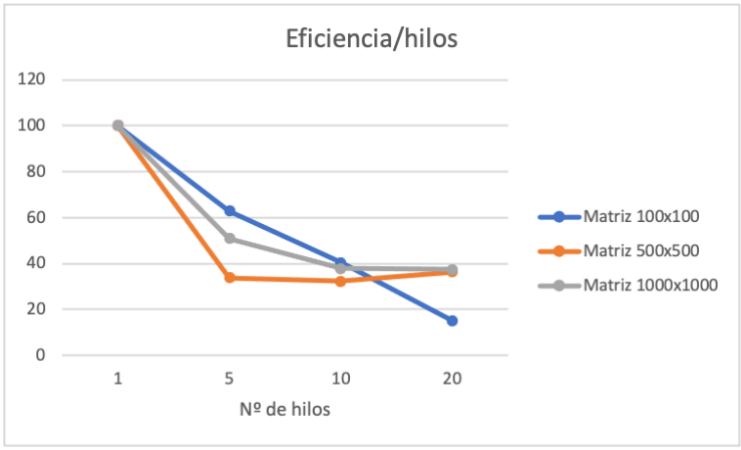
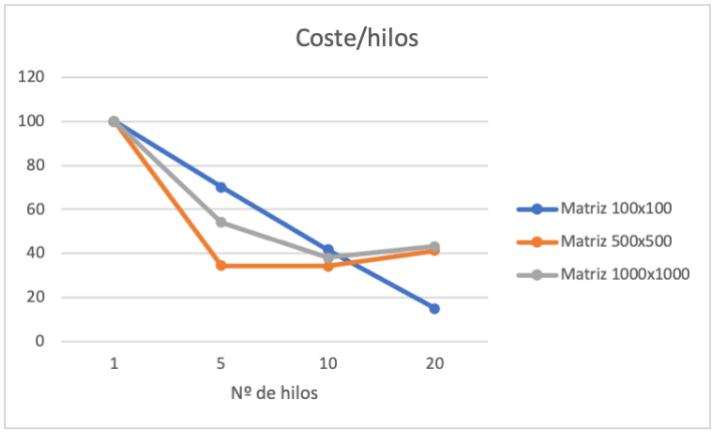




Ilustración 7: Gráfica de comparación del coste frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)



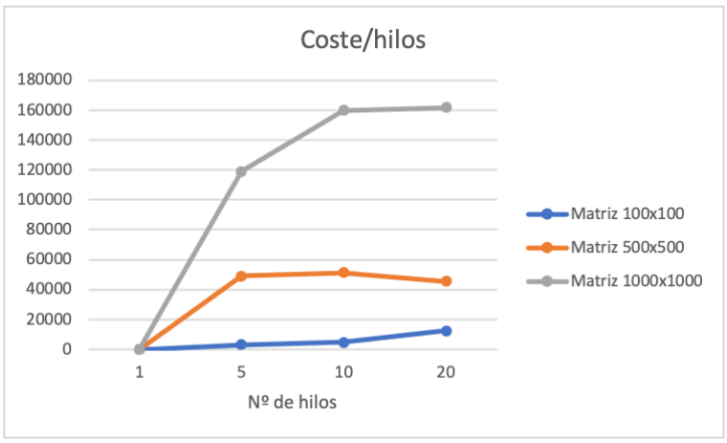




Ilustración 8: Gráfica de comparación de la eficiencia frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha))

## 2.3. Conclusiones

En conclusión, se puede observar que , a medida que aumenta el número de procesos, disminuye el tiempo total (manteniendo el número de parámetros constantes) en ambos casos. Ya que se realiza el mismo trabajo sobre la plancha, pero de forma paralela, por tanto, cuantos más procesos trabajen sobre los puntos de cálculo de temperatura , menos carga de trabajo obtendrán sobre la matriz y antes terminarán la ejecución.

En cuanto al caso base planteado con 1, 5, 10 y 20 hilos para matrices de 100x100, 500x500 y 1000x1000, se ha analizado que, tanto para la implementación en la máquina local como en la máquina en remoto, ambas presentan valores similares comparando ambos tipos de división.

Sin embargo, se pueden denotar dos diferencias en la comparativa. Por un lado, se puede ver que, tanto en la máquina local como en la máquina en remoto, la división por filas supone valores peores en cuanto a tiempo de cálculo y de ejecución, eficiencia y coste frente a la división por columnas. Esto se debe a que el cálculo de división de la matriz en subtareas para cada uno de los hilos es mucho más rápido a la hora de compactar cuando se realiza por columnas ya que el recorrido de la matriz es menor, pues en el bucle anidado que recorre la matriz para asignar subtareas no itera sobre las filas, sólo sobre las columnas, mientras que en el caso contrario itera en ambos.

Y, por otro lado, que la máquina en tiene valores más bajos en cuanto a tiempo de cálculo y ejecución que la máquina en debido a la diferencia notable en prestaciones (CPU, número de procesadores, etc.) con los que cuenta. También cabe mencionar que el análisis del programa con Pthreads en la máquina en remoto se realizó sin ninguna otra tarea o usuario dentro del sistema que pudiera hacer que el rendimiento disminuyese, mientras que la máquina en local se realizó sobre un sistema operativo Windows con varias tareas ejecutándose en background.

# 3. Implementación con procesos (MPI)

## 3.1. Análisis en máquina local

Dentro del análisis realizado del programa mediante el uso de la biblioteca MPI se ha implementado una estrategia similar , aunque no idéntica, a la realizada en el caso del uso de la biblioteca de Pthreads. Para los datos extraídos en esta implementación no se ha hecho una propuesta de división por filas y columnas para repartir las tareas entre los procesos. En este caso se ha optado, debido a un replanteamiento del programa debido al uso de la biblioteca, por una división en submatrices, de tal manera que quedaría representado como se ve en la Ilustración 9.

Aquí se puede ver que , dado que la ventaja que dan los procesos en que no dependen del número de procesadores de la máquina en la que se ejecutan como los hilos, ya que se pueden ejecutar en varias máquinas. En este caso, la comunicación se realiza por paso de mensajes mediante llamadas a MPI_Send y MPI_Recv y, con el fin de reducirla, el reparto de las subtareas se realizará mediante un reparto cartesiano, siendo el número de filas y de columnas dos números que , multiplicados entre sí, resulte el número total de procesos ejecutándose y cuya diferencia sea la menor posible.

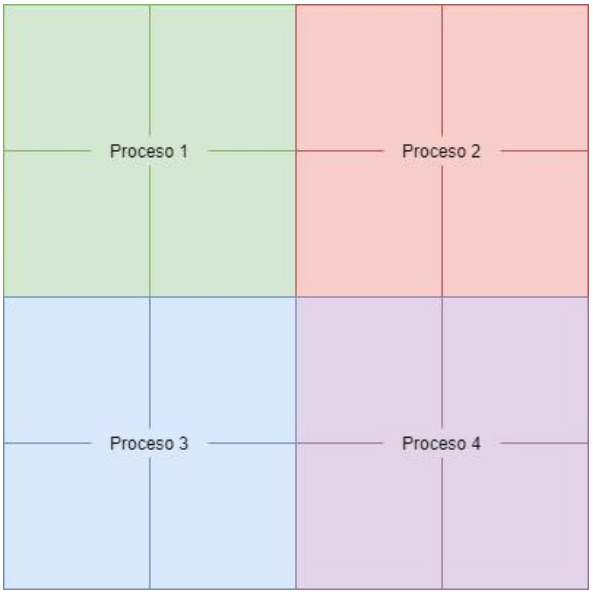



Ilustración 9: Esquema de división por submatrices en MPI con un ejemplo de una matriz de 4x4 en división para procesos en submatrices de 2x2 para 4 procesos

Debido a la limitación de procesos por parte de la versión de OpenMPI utilizada para esta práctica se ha partido del análisis de tres casos bases con 1, 2, y 3 procesos variando el tamaño de la matriz se ha llevado a cabo haciendo uso de un ordenador personal con las mismas especificaciones que el anterior. Los resultados en la máquina local se pueden observar en la Tabla 5.

Tabla 5: Resultados del análisis con MPI en máquina en local

| Nº procesos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 0,521078 | 1 | 100 % | 0 |
| **2** | 100x100 | 1000 | 0,266421 | 1,95584 | 97,7 % | 0,532842 |
| **3** | 100x100 | 1000 | 0,303439 | 1,717224 | 57,2 % | 0,910317 |
| **4** | 100x100 | 1000 | 0,176422 | 2,95358855 | 73,8 % | 0,705688 |
| **1** | 500x500 | 1000 | 13,562 | 1 | 100 % | 0 |
| **2** | 500x500 | 1000 | 8,396 | 1,615293 | 80,7 % | 16,792 |
| **3** | 500x500 | 1000 | 6,574 | 2,06297536 | 68,7 % | 19,722 |
| **4** | 500x500 | 1000 | 4,05641 | 3,3433504 | 83,5 % | 16,22564 |
| **1** | 1000x1000 | 1000 | 52,964 | 1 | 100 % | 0 |
| **2** | 1000x1000 | 1000 | 32,802 | 1,61465764 | 80,7 % | 65,604 |
| **3** | 1000x1000 | 1000 | 30,643 | 1,72842085 | 57,6 % | 91,929 |
| **4** | 1000x1000 | 1000 | 16,2186 | 3,26563329 | 81,6 % | 64,8744 |

En cuanto a la tendencia del programa, se puede denotar un análisis similar al descrito en la ejecución del programa con la biblioteca de Pthreads en la máquina local. En la que se puede observar que , tal y como ocurría en otros casos, cuanto mayor es el tamaño de la matriz mayores son los tiempos de cómputo. Además , a medida que aumentamos el número de procesos en ejecución del programa vemos como la tendencia de tiempo de ejecución es decreciente, al igual que ocurría en la implementación anterior.

Sin embargo, a diferencia de los Pthreads, a medida que aumenta el número de procesos , al hacer uso de los mecanismos de comunicación por paso de mensajes, aumentan las prestaciones ya que se requieren un número de recursos mayor, por lo que el speedup aumenta. Esto provoca que , a mayor número de procesos la eficiencia disminuya y aumente el coste computacional de cada proceso a la hora de realizar las tareas. Como se puede observar en la Ilustración 13, el coste para una matriz de 1000x1000 es mucho mayor en comparación con las matrices de 100x100 y 500x500. Esto se debe a lo explicado anteriormente.

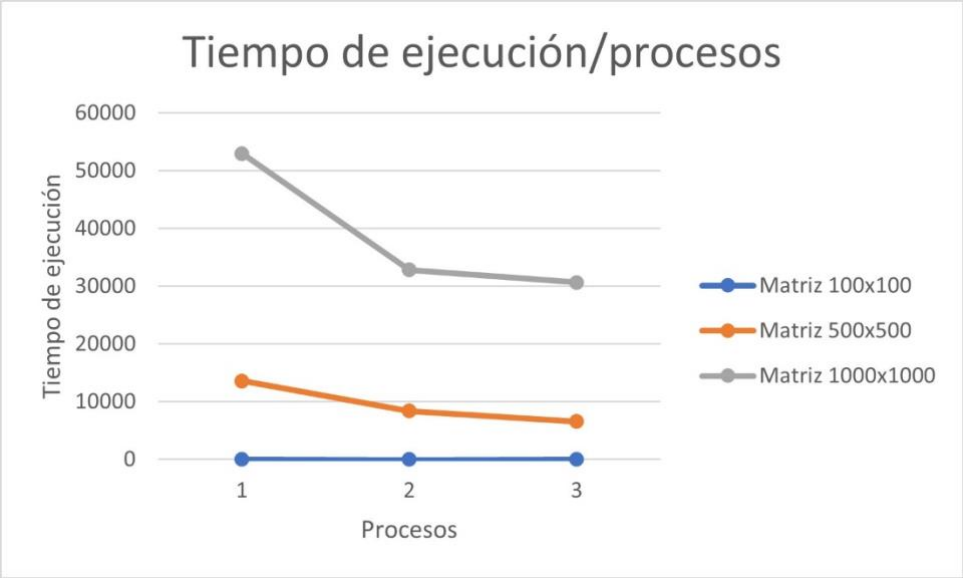



Ilustración 10: Gráfica de representación del tiempo de ejecución frente al número de procesos en la máquina local

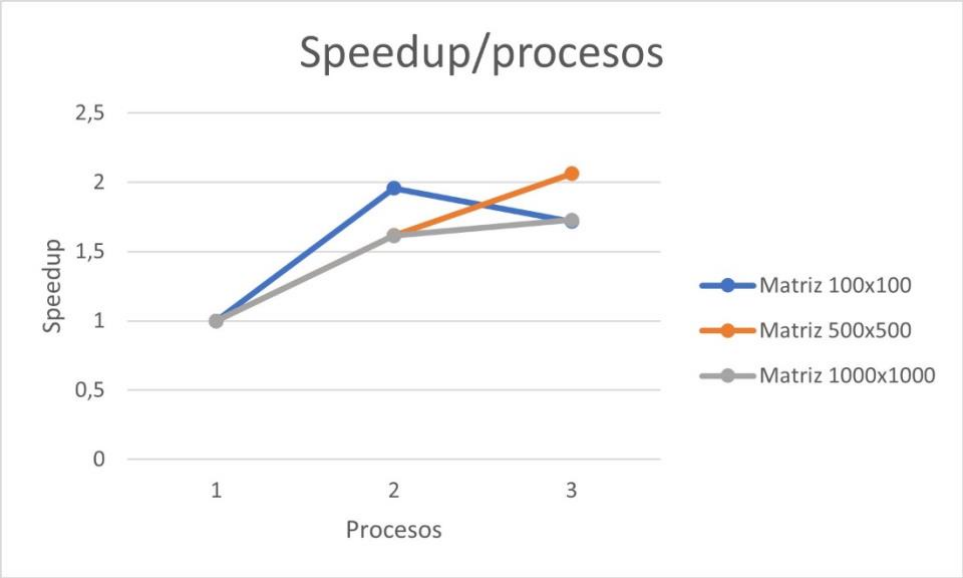



Ilustración 11: Gráfica de representación de speedup frente al número de procesos en la máquina local

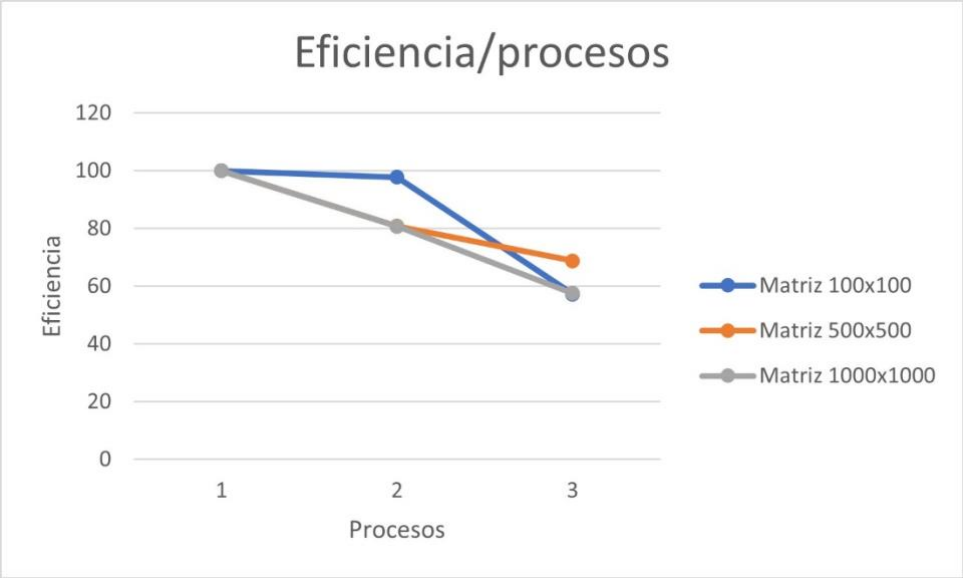



Ilustración 12: Gráfica de representación de eficiencia frente al número de procesos en la máquina local

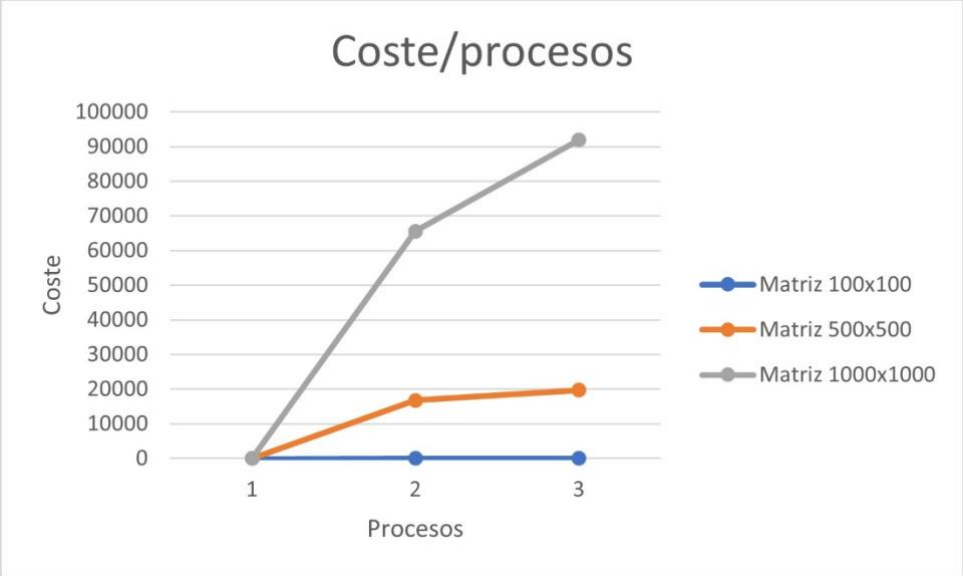



Ilustración 13: Gráfica de representación de coste frente al número de procesos en la máquina local

## 3.2. Análisis en máquina remota vdia3.fis.usal.es

Tabla 6: Resultados del análisis con MPI en máquina en remoto

| Nº procesos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 1,733790 | 1 | 100 % | 0 |
| **2** | 100x100 | 1000 | 0,875482 | 1,980383377 | 99,01916887 | 1,750964 |
| **3** | 100x100 | 1000 | 0,597344 | 2,902498393 | 96,74994643 | 1,792032 |
| **4** | 100x100 | 1000 | | | | |
| **1** | 500x500 | 1000 | 42,038900 | 1 | 100 % | 0 |
| **2** | 500x500 | 1000 | 21,093300 | 1,992997777 | 99,64988883 | 42,1866 |
| **3** | 500x500 | 1000 | 14,380700 | 2,923286071 | 97,44286903 | 43,1421 |
| **4** | 500x500 | 1000 | | | | |
| **1** | 1000x1000 | 1000 | 167,357000 | 1 | 100 % | 0 |
| **2** | 1000x1000 | 1000 | 83,935800 | 1,993869124 | 99,69345619 | 167,8716 |
| **3** | 1000x1000 | 1000 | 55,976100 | 2,989793858 | 99,65979528 | 167,9283 |
| **4** | 1000x1000 | 1000 | 33,872300 | 4,940821851 | 98,81643703 | 169,3615 |

La tendencia del programa sigue una evolución similar a la analizada en el caso de uso de la biblioteca de Pthreads en el servidor como se puede observar en la Tabla 6. Tal y como ocurría en dichos casos, cuanto mayor es el tamaño de la matriz mayores son los tiempos de cómputo. Además, a medida que aumentamos el número de procesos en ejecución del programa vemos como la tendencia de tiempo de ejecución es decreciente, así, de igual modo a como ocurría en hilos, el speedup crece, decreciendo la eficiencia y aumentando el coste.

En segundo lugar, para un solo proceso se obtienen peores prestaciones que en la máquina local, pero a medida que aumentamos el nº de procesos, al disponer la máquina servidor de más núcleos, se van obteniendo mejores prestaciones que en local, más acentuadas a medida que aumenta el tamaño de la matriz sobre la que se actúa.

Finalmente, observando los datos, podemos darnos cuenta de que mediante el uso de la biblioteca de MPI se obtienen unas mejores prestaciones relativas a la obtención de un mayor speedup, una mayor eficiencia y un menor coste de computación que aquellas obtenidas mediante el uso de la biblioteca Pthreads.

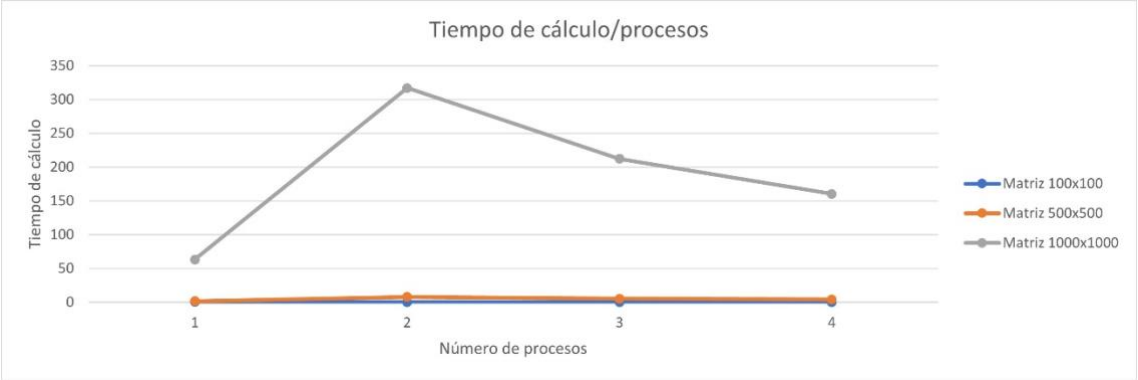



Ilustración 14: Gráfica de representación del tiempo de ejecución frente al número de procesos en la máquina remota

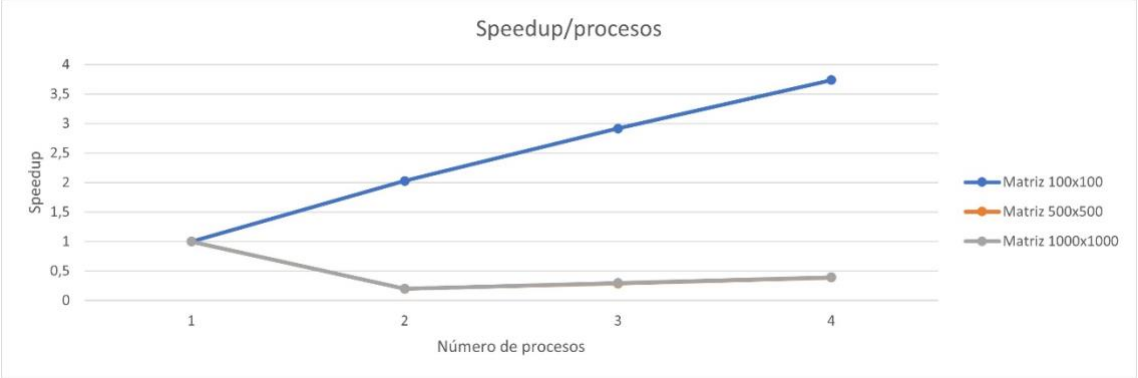



Ilustración 15: Gráfica de representación del speedup frente al número de procesos en la máquina remota

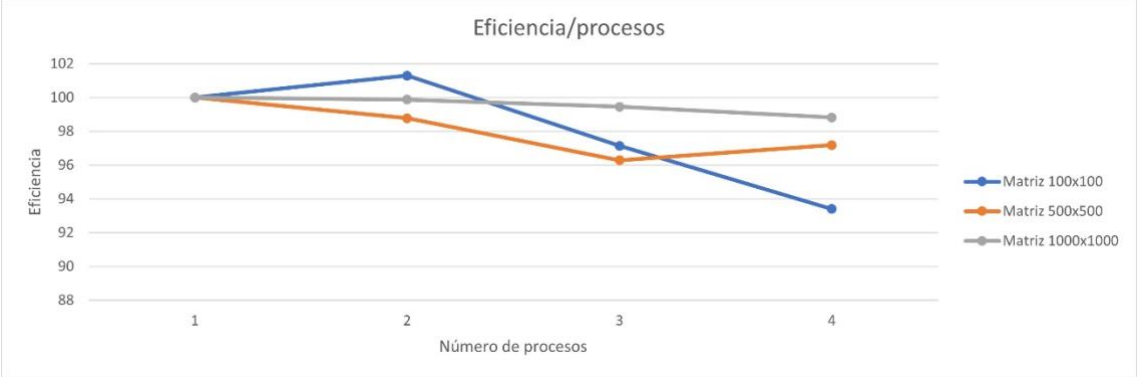



Ilustración 16: Gráfica de representación de la eficiencia  frente al número de procesos en la máquina remota

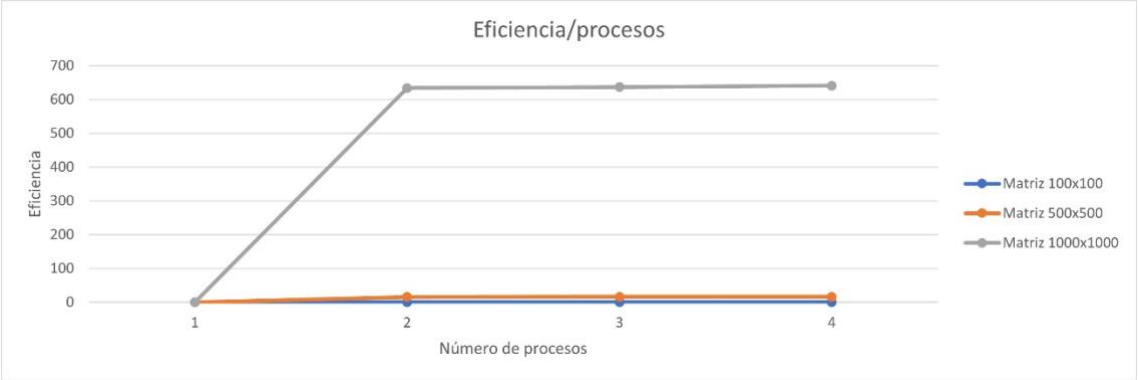



Ilustración 17: Gráfica de representación del coste frente al número de procesos en la máquina remota

## 3.3. Conclusiones

Tal y como se puede observar en los análisis previos, el uso de la paralelización para la resolución del problema del cálculo de la temperatura haciendo uso de la biblioteca MPI supone un consumo de tiempo menor para los procesos en la realización de las tareas, suponiendo en este caso una mejora frente al tiempo que consumían los hilos haciendo uso de la biblioteca Pthreads.

Sin embargo, y aunque los rangos de eficiencia son mayores que los de Pthredas y los de coste son menores, el uso de esta biblioteca se ve afectado en el uso de la comunicación del paso por mensajes, que supone un impacto en rendimiento debido a un aumento del speedup y a una ligera disminución en eficiencia, debido a que los procesos consumen parte del tiempo en realizar tareas de comunicación como en realizar las tareas propias del algoritmo.

En conclusión, y haciendo alusión a lo mencionado en la introducción del análisis, el uso de MPI supone una cierta ventaja frente a Pthreads y es la que dan los procesos al no depender del número de procesadores de la máquina en la que se ejecutan , como sí ocurre con los hilos, ya que se pueden ejecutar en varias máquinas, reduciendo el tiempo de ejecución y aumentando la eficiencia y reduciendo coste computacional para cada proceso.

# 4. Implementación con hilos (OpenMP)

## 4.1. Análisis en máquina local

Dentro del análisis realizado del programa mediante el uso de la biblioteca MPI se ha implementado la misma estrategia planteada con Pthreads en la que los hilos harán un barrido por filas y columnas de la matriz que conforma la plancha de temperaturas. En este caso, un resumen de los datos sacados para 1,5,10 y 20 hilos para matrices de 100x100, 500x500 y 1000x1000 se puede observar en las tablas Tabla 7 para columnas y Tabla 8 para filas.

Tabla 7: Resultados del análisis con OpenMP en máquina en local usando división por columnas

| Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia (%) | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 682965 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 350035 | 1,23596794 | 100 | 681393 |
| **10** | 100x100 | 1000 | 384497 | 1,95113346 | 39,0226692 | 1750175 |
| **20** | 100x100 | 1000 | 402849 | 2,83883451 | 36,7766902 | 1852785 |
| **1** | 500x500 | 1000 | 15201401 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 4662672 | 3,22456035 | 78,3058503 | 15474994 |
| **10** | 500x500 | 1000 | 4398991 | 3,26023383 | 65,2046766 | 23313360 |
| **20** | 500x500 | 1000 | 4780752 | 3,27852957 | 65,5705915 | 23600510 |
| **1** | 1000x1000 | 1000 | 52067910 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 16375253 | 3,15938069 | 75,3058503 | 51638595 |
| **10** | 1000x1000 | 1000 | 14748542 | 3,17967057 | 63,5934114 | 81876265 |
| **20** | 1000x1000 | 1000 | 13008091 | 3,19367895 | 63,8735791 | 80845000 |

Tabla 8: Resultados del análisis con OpenMP en máquina en local usando división por filas

| Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 681393 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 370557 | 1,83883451 | 36,7766902 | 1852785 |
| **10** | 100x100 | 1000 | 387842 | 1,75688296 | 17,5688296 | 3878420 |
| **20** | 100x100 | 1000 | 369100 | 1,8460932 | 9,230466 | 7382000 |
| **1** | 500x500 | 1000 | 15474994 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 4720102 | 3,65874651 | 73,1749302 | 23600510 |
| **10** | 500x500 | 1000 | 4229589 | 3,17970918 | 31,7970918 | 42295890 |
| **20** | 500x500 | 1000 | 3724288 | 4,15515503 | 20,7757751 | 74485760 |
| **1** | 1000x1000 | 1000 | 51638595 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 16169000 | 3,19367895 | 63,8735791 | 80845000 |
| **10** | 1000x1000 | 1000 | 14353579 | 3,59761109 | 35,9761109 | 143535790 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **20** | 1000x1000 | 1000 | 12537212 | 4,11882602 | 20,5941301 | 250744240 |

Finalmente, observando los datos de ambas tablas, a continuación, se mostrarán una comparativa de las estrategias de filas y columnas de cara los análisis anteriormente planteados. En estas podemos observar que, al igual que ocurría en Pthreads, la estrategia de resolución por filas es ligeramente mejor en cuanto a tiempo de ejecución, speedup, eficiencia y coste frente al recorrido por columnas.

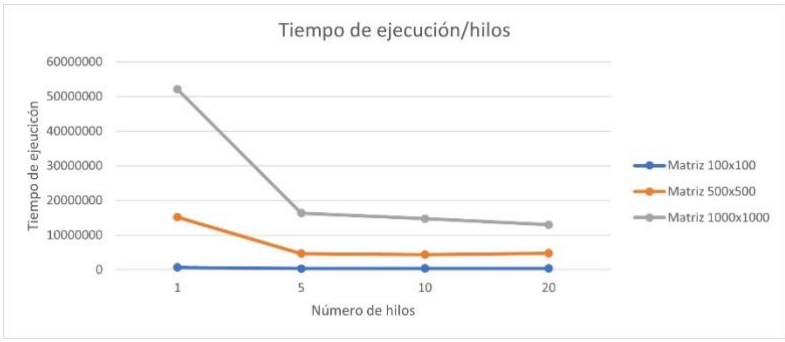




Ilustración 19: Gráfica de comparación del tiempo de ejecución del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)



Ilustración 18: Gráfica de comparación del speedup del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)



Ilustración 20: Gráfica de comparación de la eficiencia del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)

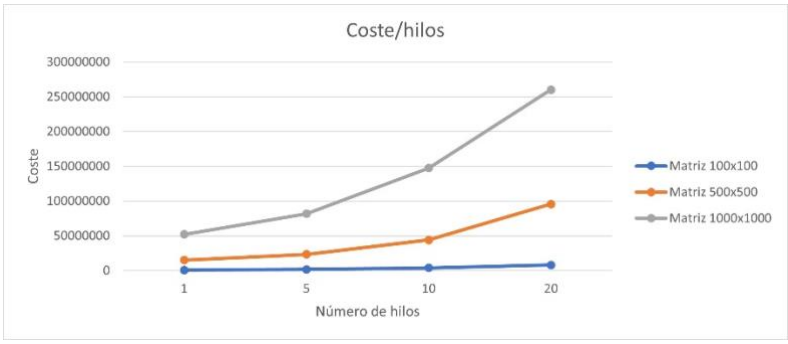



Ilustración 21: Gráfica de comparación del tiempo del coste del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha)

## 4.2. Análisis en máquina remota vdia3.fis.usal.es

Tabla 10: Resultados del análisis con OpenMP en máquina en remoto usando división por columnas

| Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 619143 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 233446 | 2,6637101 | 53,274202 | 1159565 |
| **10** | 100x100 | 1000 | 190058 | 3,27770467 | 32,7770467 | 1884700 |
| **20** | 100x100 | 1000 | 167480 | 3,73804308 | 18,6902154 | 3305200 |
| **1** | 500x500 | 1000 | 15187100 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 3594479 | 4,22668704 | 84,5337408 | 17963775 |
| **10** | 500x500 | 1000 | 2138091 | 7,10829518 | 71,0829518 | 21363000 |
| **20** | 500x500 | 1000 | 1354927 | 11,2229438 | 56,114719 | 27061440 |
| **1** | 1000x1000 | 1000 | 60719271 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 13914417 | 4,36420567 | 87,2841134 | 69563195 |
| **10** | 1000x1000 | 1000 | 7593652 | 7,99771361 | 79,9771361 | 75918720 |
| **20** | 1000x1000 | 1000 | 4220624 | 14,3921059 | 71,9605297 | 84376280 |

Tabla 11: Resultados del análisis con OpenMP en máquina en remoto usando división por filas

| Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|
| **1** | 100x100 | 1000 | 614015 | 1 | 100 | 0 |
| **5** | 100x100 | 1000 | 149238 | 4,14792623 | 82,9585246 | 738510 |
| **10** | 100x100 | 1000 | 96897 | 6,42258704 | 64,2258704 | 953910 |
| **20** | 100x100 | 1000 | 75788 | 8,31341339 | 41,567067 | 1473900 |
| **1** | 500x500 | 1000 | 15190494 | 1 | 100 | 0 |
| **5** | 500x500 | 1000 | 3280862 | 4,63198013 | 92,6396026 | 16395650 |
| **10** | 500x500 | 1000 | 1740519 | 8,73580972 | 87,3580972 | 17386900 |
| **20** | 500x500 | 1000 | 990494 | 15,3632844 | 76,8164218 | 19772940 |
| **1** | 1000x1000 | 1000 | 62957171 | 1 | 100 | 0 |
| **5** | 1000x1000 | 1000 | 13021669 | 4,83531651 | 96,7063302 | 65099695 |
| **10** | 1000x1000 | 1000 | 6744998 | 9,3361319 | 93,361319 | 67432130 |
| **20** | 1000x1000 | 1000 | 3469460 | 18,1548259 | 90,7741294 | 69354040 |

La tendencia del programa, a partir de lo que se puede comprobar en las tablas Tabla 10 y en la Tabla 11, sigue una evolución similar a la analizada en el caso de uso de la biblioteca de Pthreads en el servidor. Tal y como ocurría en dichos casos, cuanto mayor es el tamaño de la matriz mayores son los tiempos de cómputo. Además, a medida que aumentamos el número de procesos en ejecución del programa vemos como la tendencia de tiempo de ejecución es

decreciente, así, de igual modo a como ocurría en hilos, el speedup crece, decreciendo la eficiencia y aumentando el coste.

En segundo lugar, para un solo proceso se obtienen peores prestaciones que en la máquina local, pero a medida que aumentamos el nº de procesos, al disponer la máquina servidor de más núcleos, se van obteniendo mejores prestaciones que en local, más acentuadas a medida que aumenta el tamaño de la matriz sobre la que se actúa.

Finalmente, observando los datos representados en las Ilustraciones a continuación, podemos darnos cuenta de que mediante el uso de la biblioteca de OpenMP se obtienen unas mejores prestaciones relativas a la obtención de un mayor speedup, una mayor eficiencia y un menor coste de computación que aquellas obtenidas mediante el uso de la biblioteca Pthreads.

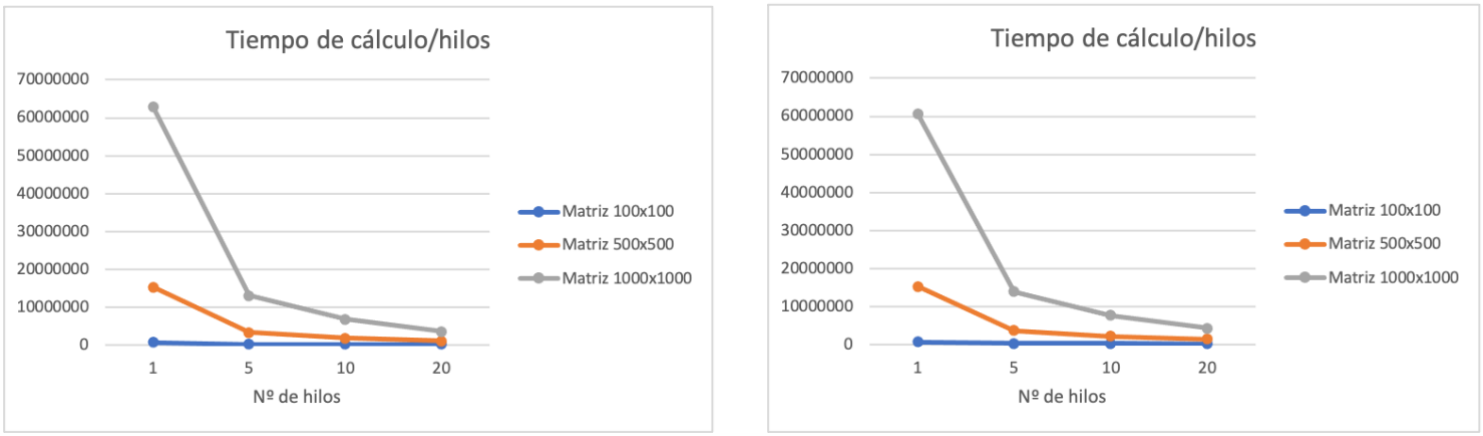



Ilustración 22: Gráfica de comparación del tiempo de ejecución del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha) de la máquina en remoto para OpenMP

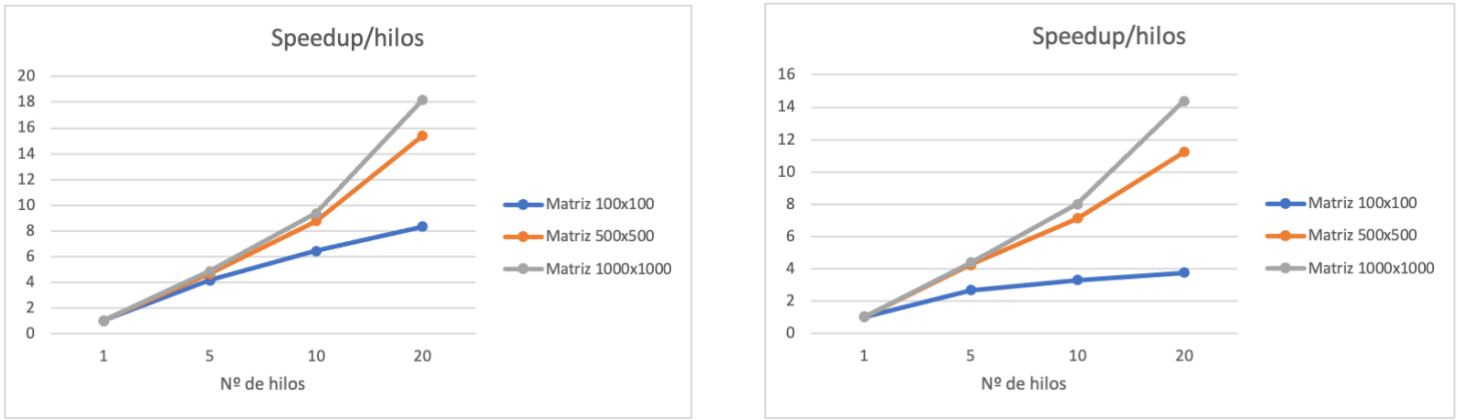
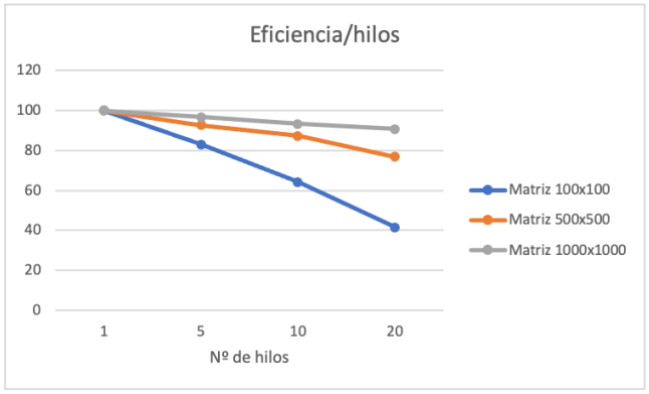



Ilustración 23: Gráfica de comparación del speedup del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha) de la máquina en remoto para OpenMP



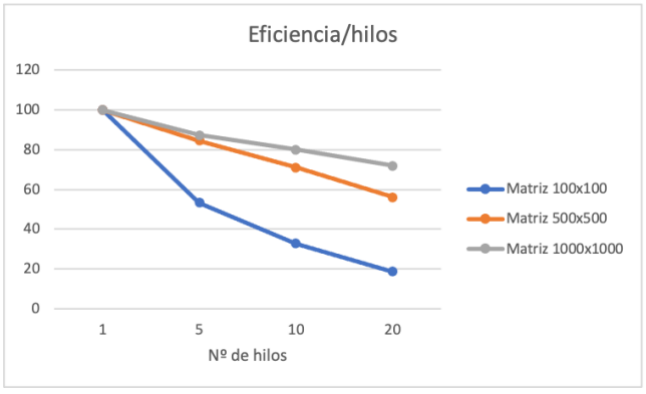
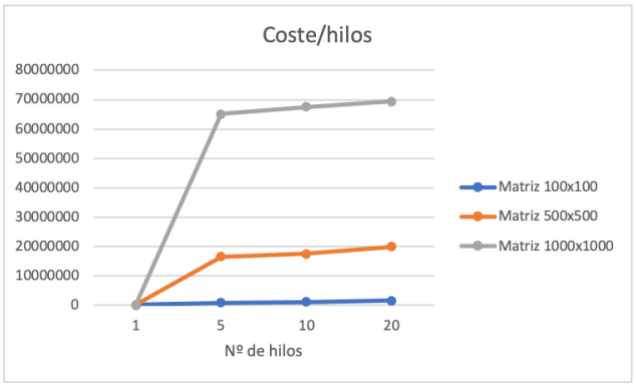




Ilustración 24: Gráfica de comparación de la eficiencia del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha) de la máquina en remoto para OpenMP



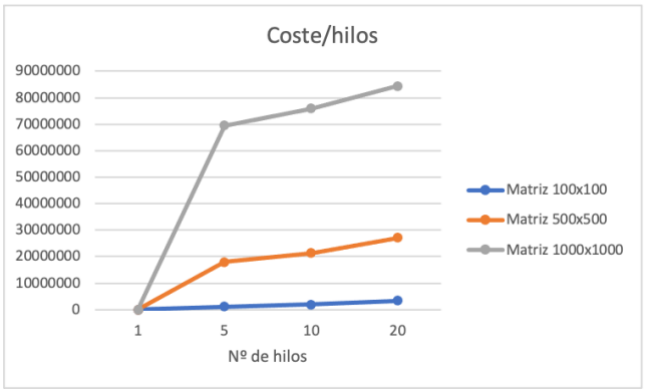




Ilustración 25: Gráfica de comparación del coste del programa frente al número de hilos utilizados usando la división por filas (izquierda) y de columnas (derecha) de la máquina en remoto para OpenMP

## 4.3. Conclusiones

Tal y como se ha ido observando a lo largo de la explicación y desarrollo de la solución mediante OpenMP, el planteamiento del ejercicio con esta herramienta supone una pequeña ventaja frente al uso de la biblioteca Pthreads, aunque siempre dependiendo de las prestaciones de la máquina en la que nos encontremos.

Esta pequeña mejoría se puede asimilar en el hecho de que, aunque los rangos de tiempo son similares, el speedup es mayor con el mismo número de procesos, la eficiencia es mayor (lo que indica que los hilos se encuentran trabajando en los cálculos y en realizar tareas útiles más tiempo) y el coste de computación es similar, por lo que se justifica con los valores de eficiencia y speedup que el uso de OpenMP es una ventaja , aunque leve, del uso de la biblioteca Pthreads.

# 5. Implementación híbrida con hilos y procesos (OpenMP + MPI)

## 5.1. Análisis en máquina local

Tabla 12: Resultados del análisis con OpenMP + MPI en máquina local usando 1 proceso

| Nº procesos | Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 100x100 | 1000 | 0,65 | 1 | 100 | 0 |
| **1** | 10 | 100x100 | 1000 | 0,26 | 2,50602335 | 83,5341117 | 0,776229 |
| **1** | 20 | 100x100 | 1000 | 0,24 | 2,67178705 | 66,7946763 | 0,97076 |
| **1** | 1 | 500x500 | 1000 | 157,12 | 1 | 100 | 0 |
| **1** | 10 | 500x500 | 1000 | 684,82 | 0,22942659 | 7,64755302 | 2054,448 |
| **1** | 20 | 500x500 | 1000 | 541,36 | 0,29022438 | 7,25560951 | 2165,428 |
| **1** | 1 | 1000x1000 | 1000 | 649,12 | 1 | 100 | 0 |
| **1** | 10 | 1000x1000 | 1000 | 274,71 | 2,36290601 | 78,7635338 | 824,139 |
| **1** | 20 | 1000x1000 | 1000 | 217,76 | 2,98091468 | 74,522867 | 871,036 |

Tabla 13: Resultados del análisis con OpenMP + MPI en máquina local usando 2 procesos

| Nº procesos | Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|---|
| **2** | 1 | 100x100 | 1000 | 0,33 | 1 | 100 | 0 |
| **2** | 10 | 100x100 | 1000 | 0,21 | 1,55181949 | 51,7273164 | 0,631413 |
| **2** | 20 | 100x100 | 1000 | 0,23 | 1,42217123 | 35,5542807 | 0,918632 |
| **2** | 1 | 500x500 | 1000 | 988,49 | 1 | 100 | 0 |
| **2** | 10 | 500x500 | 1000 | 595,08 | 1,66110829 | 55,3702764 | 1785,234 |
| **2** | 20 | 500x500 | 1000 | 514,95 | 1,91958621 | 47,9896553 | 2059,796 |
| **2** | 1 | 1000x1000 | 1000 | 385,18 | 1 | 100 | 0 |
| **2** | 10 | 1000x1000 | 1000 | 219,80 | 1,75239421 | 58,4131404 | 659,403 |
| **1** | 20 | 1000x1000 | 1000 | 196,58 | 1,95940563 | 48,9851408 | 786,316 |

Tabla 14: Resultados del análisis con OpenMP + MPI en máquina local usando 3 procesos

| Nº procesos | Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|---|
| **3** | 1 | 100x100 | 1000 | 0,30 | 1 | 100 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **3** | 10 | 100x100 | 1000 | 151,16 | 0,00196229 | 0,06540975 | 453,492 |
| **3** | 20 | 100x100 | 1000 | 244,58 | 0,00121282 | 0,03032039 | 978,312 |
| **3** | 1 | 500x500 | 1000 | 706,26 | 1 | 100 | 0 |
| **3** | 10 | 500x500 | 1000 | 309,95 | 2,27862003 | 75,954001 | 929,847 |
| **3** | 20 | 500x500 | 1000 | 283,90 | 2,48771037 | 62,1927594 | 1135,592 |
| **3** | 1 | 1000x1000 | 1000 | 275,04 | 1 | 100 | 0 |
| **3** | 10 | 1000x1000 | 1000 | 40,24 | 6,8353754 | 227,845847 | 120,711 |
| **3** | 20 | 1000x1000 | 1000 | 420,15 | 0,65460833 | 16,3652083 | 1680,608 |

Tabla 15: Resultados del análisis con OpenMP + MPI en máquina local usando 4 procesos

| **Nº procesos** | **Nº hilos** | **Tamaño Matriz** | **Nº iteraciones** | **Tiempo de ejecución (s)** | **Speedup** | **Eficiencia** | **Coste** |
|---|---|---|---|---|---|---|---|
| **4** | 1 | 100x100 | 1000 | 0,27 | 1 | 100 | 0 |
| **4** | 10 | 100x100 | 1000 | 132,16 | 0,00201665 | 0,06722153 | 396,492 |
| **4** | 20 | 100x100 | 1000 | 2445,78 | 0,00010897 | 0,00272437 | 9783,12 |
| **4** | 1 | 500x500 | 1000 | 706,26 | 1 | 100 | 0 |
| **4** | 10 | 500x500 | 1000 | 289,95 | 2,43579388 | 81,1931294 | 869,847 |
| **4** | 20 | 500x500 | 1000 | 263,90 | 2,67624613 | 66,9061531 | 1055,592 |
| **4** | 1 | 1000x1000 | 1000 | 255,04 | 1 | 100 | 0 |
| **4** | 10 | 1000x1000 | 1000 | 40,24 | 6,33832045 | 211,277348 | 120,711 |
| **4** | 20 | 1000x1000 | 1000 | 399,02 | 0,63916111 | 15,9790279 | 1596,0608 |

En este apartado, tal y como se puede deducir a partir de los datos de las Tablas 12-15 y de las Ilustraciones 26-29 el tiempo de ejecución es significativamente inferior a los obtenidos con OpenMP y con las anteriores bibliotecas utilizadas.

Aunque se puede observar una disminución de los valores de speedup en casi la mitad y unos valores de eficiencia similares a los que utilizan OpenMP y MPI por separado. El motivo por el que se ha producido unos valores menores de speedup con respecto a OpenMP es debido, principalmente, al uso combinado con MPI que requiere de gran cantidad de tiempo de ejecución para el envío de mensajes entre procesos. Y, aunque el uso de varios hilos por proceso logra que el tiempo dedicado al envío de mensajes se reduzca sigue existiendo un cuello de botella en este aspecto, el cual perjudica al rendimiento del programa.

En cuanto al coste, este aumenta con el tamaño del problema y el número de hilos utilizados por proceso, al igual que ocurría en apartados anteriores.


Ilustración 26: Gráfica de comparación del tiempo de ejecución del programa frente al número de hilos utilizados en función del número de procesos de la máquina en local para el híbrido entre OpenMP y MPI

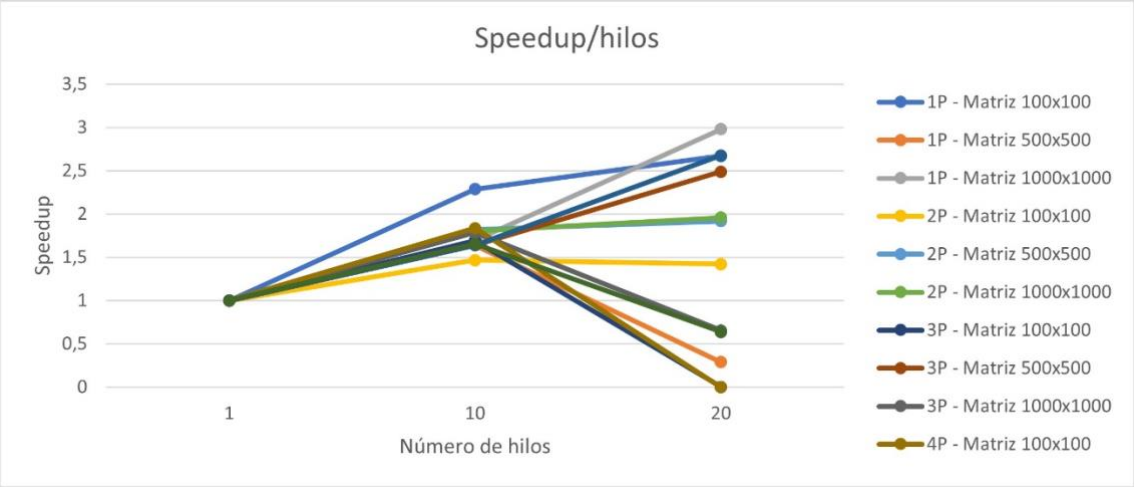



Ilustración 27: Gráfica de comparación del speedup del programa frente al número de hilos utilizados en función del número de procesos de la máquina en local para el híbrido entre OpenMP y MPI

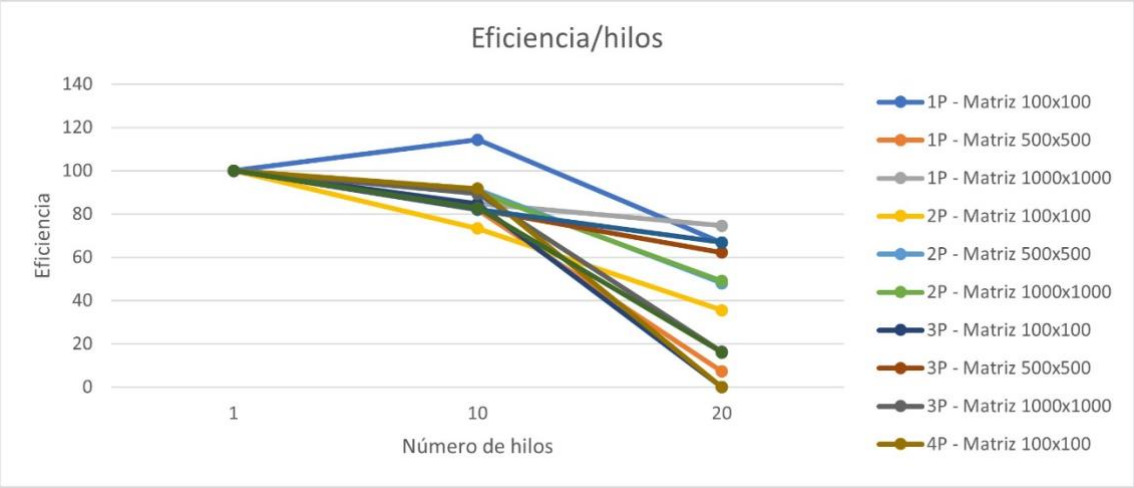



Ilustración 28: Gráfica de comparación de la eficiencia del programa frente al número de hilos utilizados en función del número de procesos de la máquina en local para el híbrido entre OpenMP y MPI

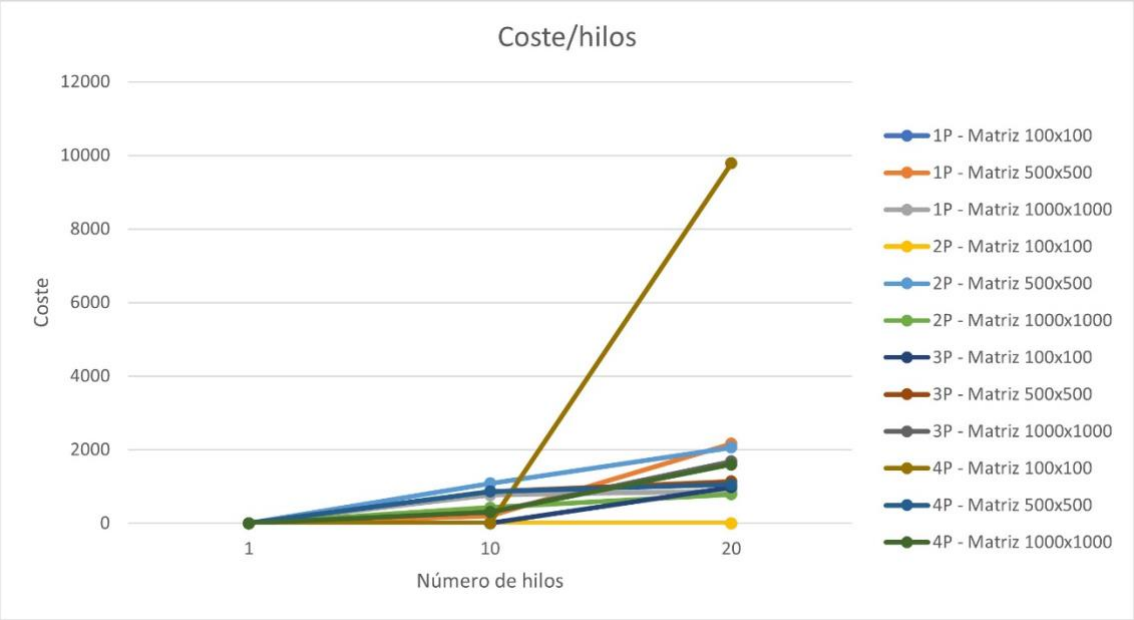



Ilustración 29: Gráfica de comparación del coste del programa frente al número de hilos utilizados en función del número de procesos de la máquina en local para el híbrido entre OpenMP y MPI

## 5.2. Análisis en máquina remota vdia3.fis.usal.es

Tabla 16: Resultados del análisis con OpenMP + MPI en máquina en remoto usando 1 proceso

| Nº procesos | Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 100x100 | 1000 | 0,667421 | 1 | 100 | 0 |
| **1** | 10 | 100x100 | 1000 | 0,567421 | 2,91457137 | 29,1457137 | 5,67421 |
| **1** | 20 | 100x100 | 1000 | 0,613419 | 2,69601855 | 13,4800927 | 12,26838 |
| **1** | 1 | 500x500 | 1000 | 15,7692 | 1 | 100 | 0 |
| **1** | 10 | 500x500 | 1000 | 13,4205 | 1,17500838 | 11,7500838 | 134,205 |
| **1** | 20 | 500x500 | 1000 | 13,8118 | 1,1417194 | 5,708597 | 276,236 |
| **1** | 1 | 1000x1000 | 1000 | 63,0894 | 1 | 100 | 0 |
| **1** | 10 | 1000x1000 | 1000 | 53,8155 | 1,17232768 | 11,7232768 | 538,155 |
| **1** | 20 | 1000x1000 | 1000 | 54,6913 | 1,15355459 | 5,76777294 | 1093,826 |

Tabla 17: Resultados del análisis con OpenMP + MPI en máquina en remoto usando 2 procesos

| Nº procesos | Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|---|
| **2** | 1 | 100x100 | 1000 | 0,324879 | 1 | 100 | 0 |
| **2** | 10 | 100x100 | 1000 | 0,311984 | 1,04133225 | 5,20666124 | 6,23968 |
| **2** | 20 | 100x100 | 1000 | 0,355724 | 0,91328952 | 2,28322379 | 14,22896 |
| **2** | 1 | 500x500 | 1000 | 7,94457 | 1 | 100 | 0 |
| **2** | 10 | 500x500 | 1000 | 6,87553 | 1,15548474 | 5,7774237 | 137,5106 |
| **2** | 20 | 500x500 | 1000 | 7,0395 | 1,12857021 | 2,82142553 | 281,58 |
| **2** | 1 | 1000x1000 | 1000 | 31,7513 | 1 | 100 | 0 |
| **2** | 10 | 1000x1000 | 1000 | 27,4119 | 1,15830351 | 5,79151755 | 548,238 |
| **2** | 20 | 1000x1000 | 1000 | 27,8381 | 1,14056994 | 2,85142485 | 1113,524 |

Tabla 18: Resultados del análisis con OpenMP + MPI en máquina en remoto usando 3 procesos

| Nº procesos | Nº hilos | Tamaño Matriz | Nº iteraciones | Tiempo de ejecución (s) | Speedup | Eficiencia | Coste |
|---|---|---|---|---|---|---|---|
| **3** | 1 | 100x100 | 1000 | 0,22702 | 1 | 100 | 0 |
| **3** | 10 | 100x100 | 1000 | 0,082166 | 2,76294331 | 9,20981103 | 2,46498 |
| **3** | 20 | 100x100 | 1000 | 14,7017 | 0,01544175 | 0,02573625 | 882,102 |
| **3** | 1 | 500x500 | 1000 | 5,33443 | 1 | 100 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **3** | 10 | 500x500 | 1000 | 1,15068 | 4,63589356 | 15,4529785 | 34,5204 |
| **3** | 20 | 500x500 | 1000 | 13,6735 | 0,39012908 | 0,65021514 | 820,41 |
| **3** | 1 | 1000x1000 | 1000 | 21,2102 | 1 | 100 | 0 |
| **3** | 10 | 1000x1000 | 1000 | 4,23996 | 5,00245285 | 16,6748428 | 127,1988 |
| **3** | 20 | 1000x1000 | 1000 | 17,9342 | 1,18266775 | 1,97111292 | 1076,052 |

Tabla 19: Resultados del análisis con OpenMP + MPI en máquina en remoto usando 4 procesos

| **Nº procesos** | **Nº hilos** | **Tamaño Matriz** | **Nº iteraciones** | **Tiempo de ejecución (s)** | **Speedup** | **Eficiencia** | **Coste** |
|---|---|---|---|---|---|---|---|
| **4** | 1 | 100x100 | 1000 | 0,12507 | 1 | 100 | 0 |
| **4** | 10 | 100x100 | 1000 | 0,057106 | 2,19013764 | 21,9013764 | 0,57106 |
| **4** | 20 | 100x100 | 1000 | 10,7517 | 0,01163258 | 0,0581629 | 215,034 |
| **4** | 1 | 500x500 | 1000 | 4,4443 | 1 | 100 | 0 |
| **4** | 10 | 500x500 | 1000 | 0,556988 | 7,97916652 | 79,7916652 | 5,56988 |
| **4** | 20 | 500x500 | 1000 | 11,2335 | 0,39562914 | 1,97814572 | 224,67 |
| **4** | 1 | 1000x1000 | 1000 | 19,2302 | 1 | 100 | 0 |
| **4** | 10 | 1000x1000 | 1000 | 2,05802 | 9,3440297 | 93,440297 | 20,5802 |
| **4** | 20 | 1000x1000 | 1000 | 15,9992 | 1,2019476 | 6,00973799 | 319,984 |

En cuanto al uso del servidor remoto habilitado para realizar las pruebas, el programa utilizando la solución híbrida y observando los datos proporcionados en las Tablas 16-19 y de las Ilustraciones 30-33, el tiempo de ejecución es menor que el proporcionado en la máquina en local.

Esto es debido a que, para la máquina en local se ha utilizado una máquina virtual del sistema operativo Debian para las pruebas simulando una máquina nativa con este sistema operativo y esto puede haber afectado, en gran medida, a la ejecución del programa y la representación de los valores.

Y es por ello por lo que, el servidor, al tratarse de una máquina con el sistema operativo nativo representa valores que se adecúan más a la realidad.

Se puede observar, además , que, ante un aumento del número de procesos por máquina, a medida que aumentan el número de hilos por proceso (y al igual que ocurría en el análisis de la máquina local) se observa una reducción del tiempo de ejecución hasta que con 3-4 procesos el tiempo de ejecución se estabiliza y se muestran valores muy similares.

De forma análoga ocurre con los valores de speedup, eficiencia y coste, en los que los valores son similares con 1-2 procesos. Pero a partir de los 3-4 procesos los valores se estabilizan dentro de la misma franja, independientemente del número de hilos que se crean por proceso.

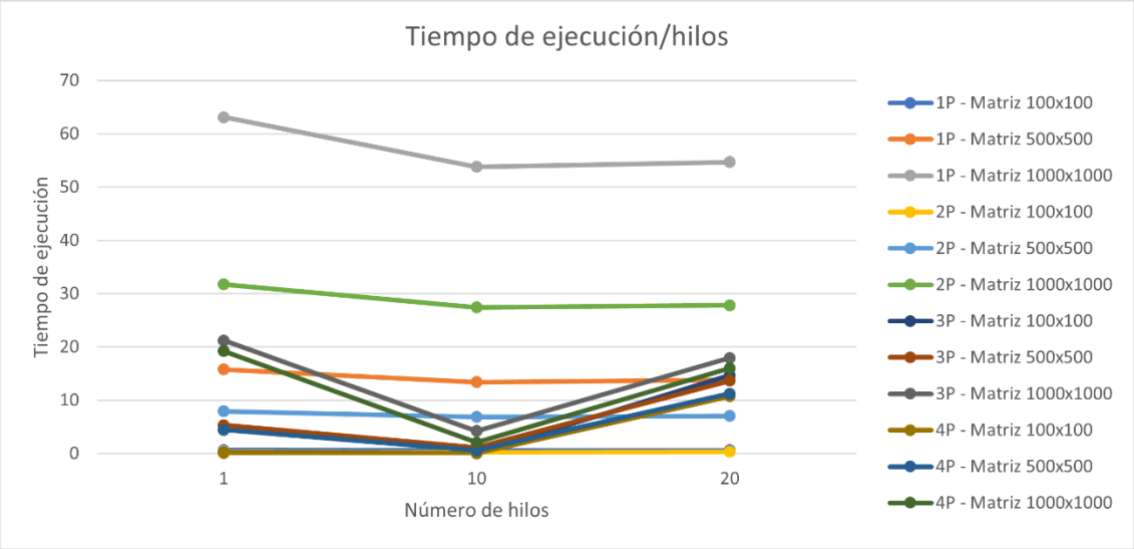



Ilustración 30: Gráfica de comparación del tiempo de ejecución del programa frente al número de hilos utilizados en función del número de procesos de la máquina en remoto para el híbrido entre OpenMP y MPI

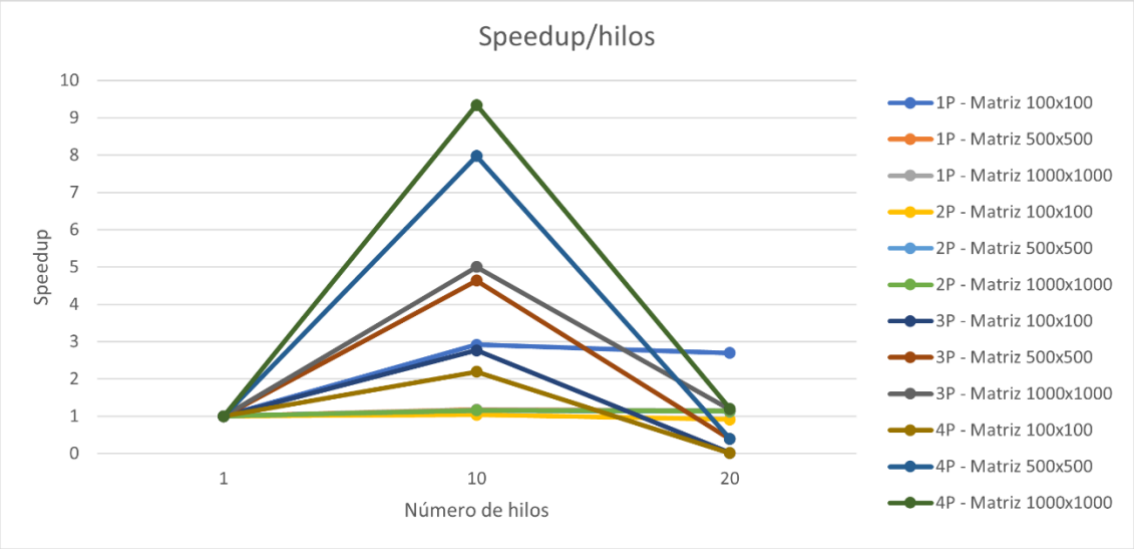



Ilustración 31: Gráfica de comparación del speedup del programa frente al número de hilos utilizados en función del número de procesos de la máquina en remoto para el híbrido entre OpenMP y MPI

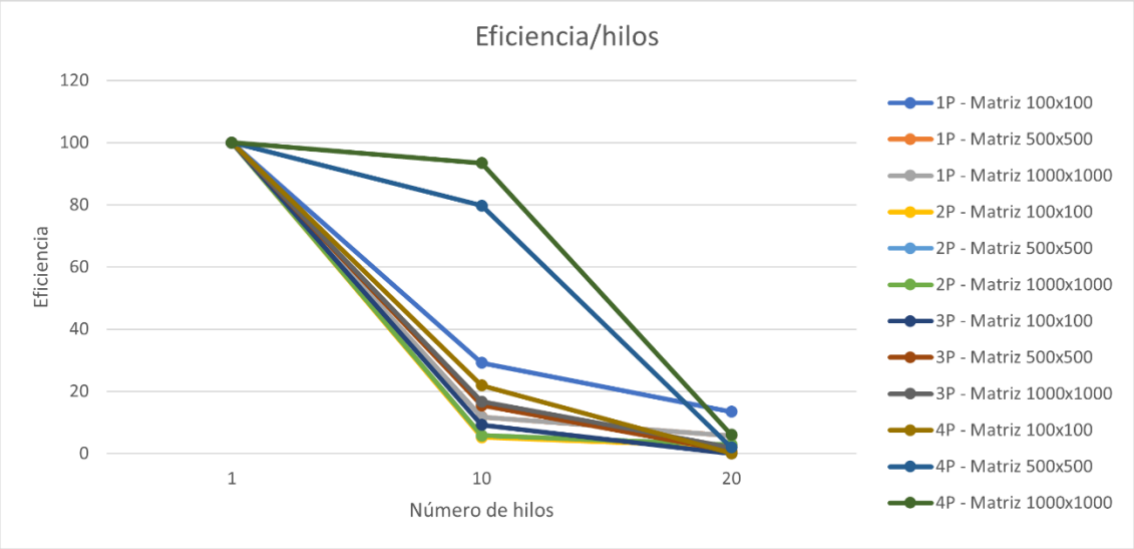



Ilustración 32: Gráfica de comparación de la eficiencia del programa frente al número de hilos utilizados en función del número de procesos de la máquina en remoto para el híbrido entre OpenMP y MPI

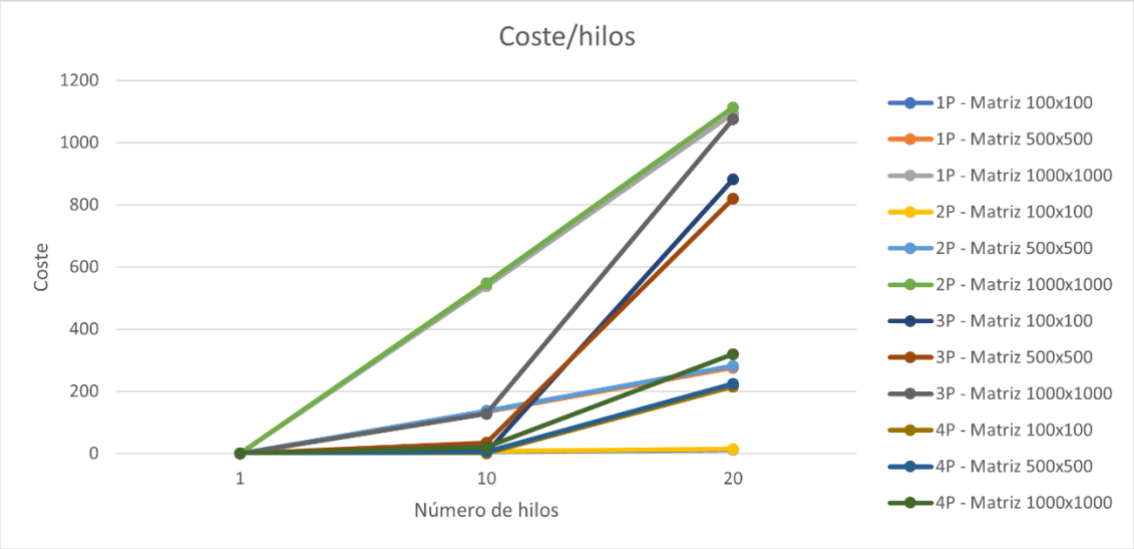



Ilustración 33: Gráfica de comparación del coste del programa frente al número de hilos utilizados en función del número de procesos de la máquina en remoto para el híbrido entre OpenMP y MPI

## 5.3. Conclusiones

Con la combinación de procesos con MPI junto con la paralelización de OpenMP se ha conseguido aprovechar de una forma más eficiente los recursos de las máquinas (tanto local como remota) logrando un tiempo de ejecución menor (en algunos casos perjudicados por los tiempo de comunicación entre procesos cuando el número de procesos es mayor que 1), un speedup y eficiencia mayor y un menor coste.

Además, al aumentar el número de procesos por máquina (aunque aumenten el número de hilos) disminuye el tiempo de ejecución y sobre todo, cuando se realiza con procesos que son múltiplos de 4 (debido a la estrategia de la división cartesiana implementada y explicada en el apartado de Implementación con procesos (MPI))

En definitiva, con esta solución implementada usando la combinación híbrida de las bibliotecas OpenMP con MPI se consigue utilizar las ventajas de ambos con el fin de lograr unos valores de tiempo, speedup, eficiencia y coste mucho mejores que el resto de las bibliotecas analizadas en este informe. Sin embargo, cabe destacar que, aunque con múltiplos de 4 procesos se consigue aprovechar la estrategia planteada para la división cartesiana , la comunicación entre ellos debido a la biblioteca MPI supone un impacto en las métricas analizadas, siendo esta la principal razón para utilizar este planteamiento en sustitución de la estrategia de división por filas y columnas, como se ha explicado anteriormente.

# 6. Conclusiones finales

En conclusión, del análisis realizado en todos los apartados, se analizará brevemente los puntos destacables del uso de cada biblioteca, ventajas y desventajas y resumen de las conclusiones extraídas.

Pthreads , por un lado, es la biblioteca que se ha utilizado como base de comparación de las otras bibliotecas, aprovechando la paralelización de los hilos para el reparto de tareas de la matriz entre ellos.

Por otro lado, MPI, supone un cambio sustancial en cuanto a speedup, coste y eficiencia, aunque no tanto en tiempo de ejecución frente a Pthreads. Esto es debido a que la comunicación entre procesos provoca una desventaja frente a las otras bibliotecas, desaprovechando tiempo en el envío de mensajes entre procesos.

En cuanto a OpenMP, el uso de directivas reduce el tiempo frente a su homóloga, Pthreads, reduciendo el tiempo de ejecución y coste y aumentando speedup, eficiencia. Sin embargo, según aumentan el número de hilos los valores extraídos de estas métricas no distan demasiado de los extraídos con Pthreads.

En definitiva, se puede afirmar que, cuanto mayor es el tamaño de la matriz, mayor es la eficiencia del algoritmo de paralalelización debido a que un aumento del problema hace que la parte paralelizable crezca de forma cuadrática mientras que la parte secuencial (1 hilo – 1 proceso) crezca de forma lineal.

Como se ha mencionado, el uso de la implementación híbrida OpenMP con MPI no dista demasiado de los valores obtenidos con OpenMP aunque si supone una mejora y es la implementación que mejores resultados ha obtenido. Esto es debido a que ejecutar un proceso por máquina que use tanto hilos como procesadores tenga dicha máquina supone la estrategia más eficiente para aprovechar recursos y reducir costes.

# Bibliografía

**No hay ninguna fuente en el documento actual.**