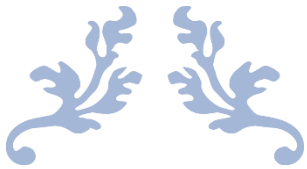

PRÁCTICA 1. BASES DE DATOS PARALELAS

Jose Lluch Palop
Javier Argente Micó

Grado en Ingeniería Informática
Sistemas de Información de Nueva Generación

Índice

1. Introducción. 2

2. Conclusiones tras ejecución de pruebas. 3

2.1 Conclusiones de No Transactional read-only. 4

2.2 Conclusiones de Transactional read-only. 9

2.3 Conclusiones de Transactional read-Write. 12

3. Conclusiones y comparativa de las tres tesituras. 15

1. Introducción.

Primero de todo decir que las pruebas que hemos llevado a cabo las hemos realizado utilizando el sistema de gestión de bases de datos MariaDB, y a través de una máquina basado en el procesador Intel Xeon E5-2620 que nos han proporcionado, dicho esto comenzaremos introduciendo los conceptos sobre los que se basan las pruebas que hemos hecho, y mediciones que hemos sacado como resultado de estas.

Llamamos **transacción**, a la unidad de trabajo compuesta por diversas tareas cuyo resultado final debe ser ejecutar todas estas tareas o ninguna de ellas.

Con esto queremos decir que dentro de un sistema de base de datos todas las operaciones relacionadas entre sí que se ejecuten dentro de un mismo flujo lógico de trabajo deben ejecutarse en bloque. De esta manera si todas estas operaciones conjuntas tienen éxito, el bloque de trabajo tiene éxito, pero si cualquiera de ellas falla, deberán retroceder todas las anteriores que se hayan realizado correctamente, con el objetivo de evitar que la base de datos quede en un estado de inconsistencia.

Con **speedup**, nos referimos al cálculo realizado con el objetivo de dar una representación a la mejora en la velocidad de ejecución de una tarea, ejecutado en dos arquitecturas similares, pero con diferentes recursos, en nuestro caso, la arquitectura es la misma, pero variando la cantidad de hilos que utilizamos para llevar a cabo las pruebas.

Este concepto fue introducido por la ley de Amdahl, que está enfocada principalmente a la computación paralela, pero también se puede usar de forma más general para mostrar el efecto en el rendimiento de cualquier mejora en los recursos.

Cuando hablemos de **coste**, nos referimos a la medida del trabajo total que han realizado todos los procesadores involucrados dentro de las transacciones que estemos llevando a cabo.

Finalmente hablaremos de la **eficiencia**, con ello nos referimos a una medida alternativa en cuanto al rendimiento de un programa paralelo, en nuestro caso una serie de operaciones sobre una base de datos paralela.

Dicho dato representa la fracción de tiempo que un procesador es aprovechado para llevar a cabo el cálculo o las operaciones requeridas, además, nos permite conocer si un sistema es **escalable** para un determinado número de procesadores, siempre y cuando se cumpla que la eficiencia se mantenga constante y en todo momento en un factor 0.5 o superior.

2. Conclusiones tras ejecución de pruebas.

Con el fin de obtener la mejorar la percepción del comportamiento de la ejecución de la consulta preparada, tal y como se informa en el guion de esta práctica, se ha decidido montar un plan de pruebas con un total de 23 casos. Estos casos, vienen definidos en la Tabla 1.

Número de Caso	Número de hilos
1	1
2	2
3	4
4	8
5	10
6	16
7	20
8	30
9	32
10	40
11	50
12	60
13	64
14	70
15	80
16	90
17	100
18	120
19	128
20	130
21	140
22	144
23	255

Tabla 1. Plan de pruebas.

Se han escogido estos números debido a diferentes motivos. Primero de todo que, debido a la posibilidad de automatizar las ejecuciones, cuantas más mejor y así poder obtener una visión más perimétrica del comportamiento de estas conforme aumenta el número de hilos en los que se paraleliza una misma consulta ya preparada. Dicho lo cual, este plan de pruebas nace de unificar:

- Potencias de dos desde exponente uno hasta el exponente más cercano al límite de la propia máquina y, como es 144, potencia de dos con exponente 7.
- Ejecuciones de diez en diez hasta llegar al límite múltiple de 10 más cercano a 144.

- Forzar el doble de lo que, en teoría puede soportar la máquina. Como a la hora de la verdad una consulta con 288 provocaba una expulsión de la máquina, se decide lanzar la ejecución para 255, valor que aparentemente es el límite para el momento de las ejecuciones.

Cabe comentar que este plan de pruebas se ha ejecutado para los tres casos indicado, para las dos tesituras transaccionales y la única no transaccional.

2.1 Conclusiones de No Transactional read-only.

Antes que nada, hacer hincapié en la razón de existencia de un apartado de ejecuciones no transaccionales en este caso de estudio.

Como se ha comentado previamente en esta memoria en lo referido en la definición de transacción, su única finalidad es la de realizar una operación de atómica siempre y cuando se mantenga la consistencia de los datos a cambio de consumir tiempo de ejecución. Por tanto, una ejecución no transaccional se desmarca de la definición vista en el documento y en la teoría ya que se leerán datos sin importar si están actualizados a su última versión o no.

La razón de ser de este caso es para contrastar con las ejecuciones transacciones y ver todo lo que supone mantener la consistencia del conjunto de datos, especialmente cuando crece la cantidad de estos junto al deseo de paralelizar consultas.

Dicho esto, se procede a exponer los resultados obtenidos tras las ejecuciones. De los ficheros de *log* generados por las ejecuciones se han extraído dos parámetros: las transacciones por segundo y el tiempo total de ejecución. Después, se han calculado las transacciones totales, *Speed-up*, Coste y eficiencia. Los resultados se pueden apreciar en las gráficas de las siguientes páginas.

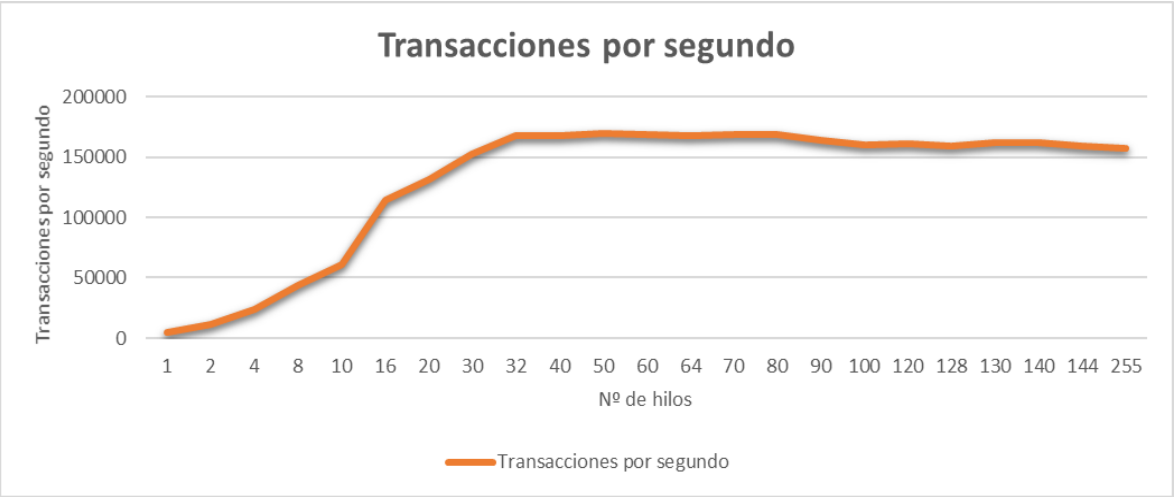

Tabla 2. Transacciones por segundo frente al número de hilos.

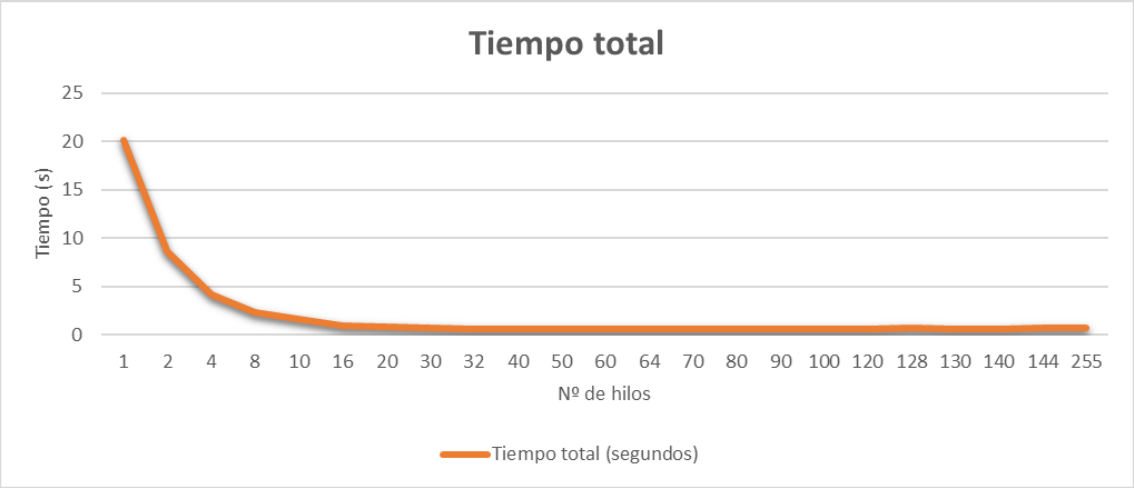

Tabla 3. Tiempo total requerido frente al número de hilos.

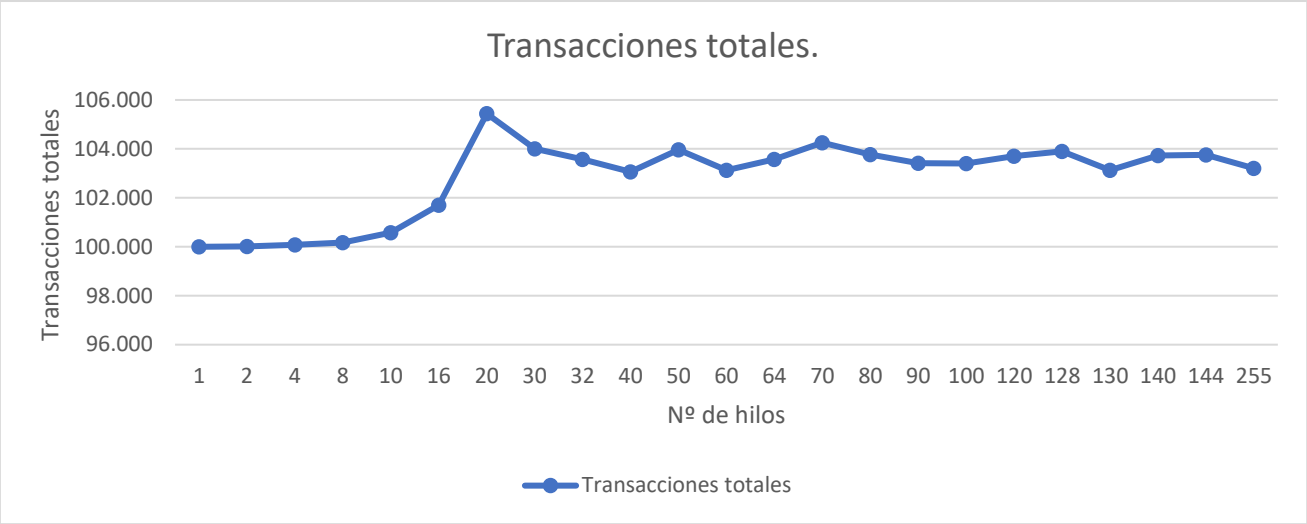

Tabla 4. Transacciones totales frente al número de hilos.

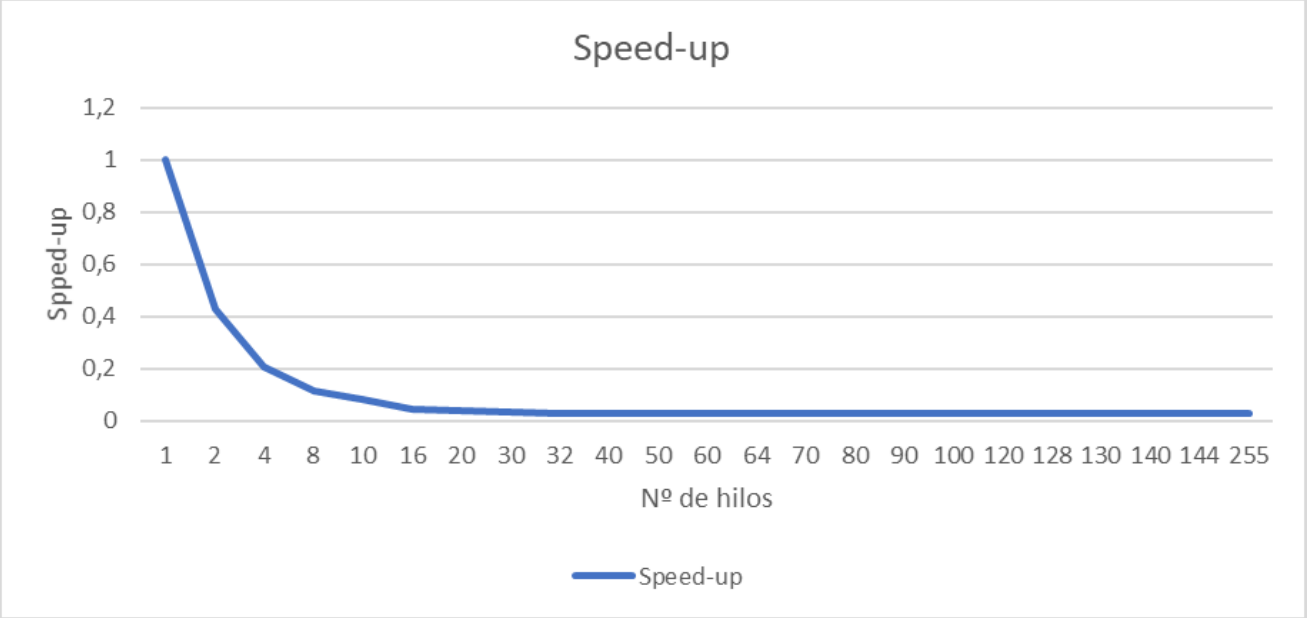

Tabla 5. Variación del incremento de la velocidad frente al número de hilos.

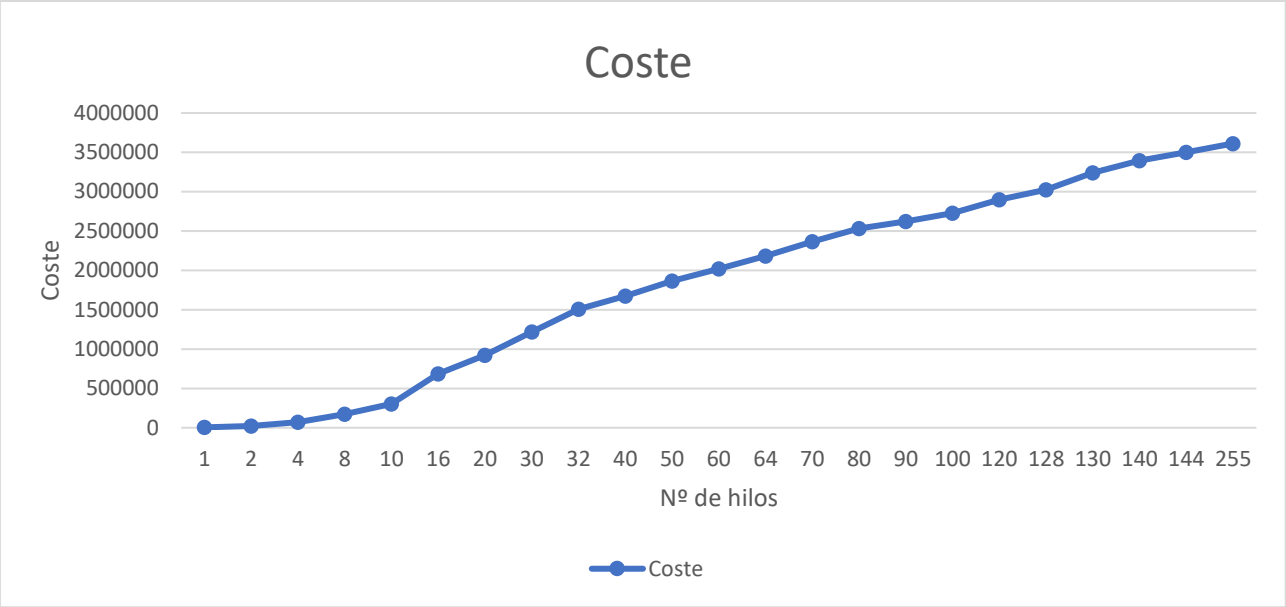

Tabla 6. Variación del Coste frente al número de hilos.

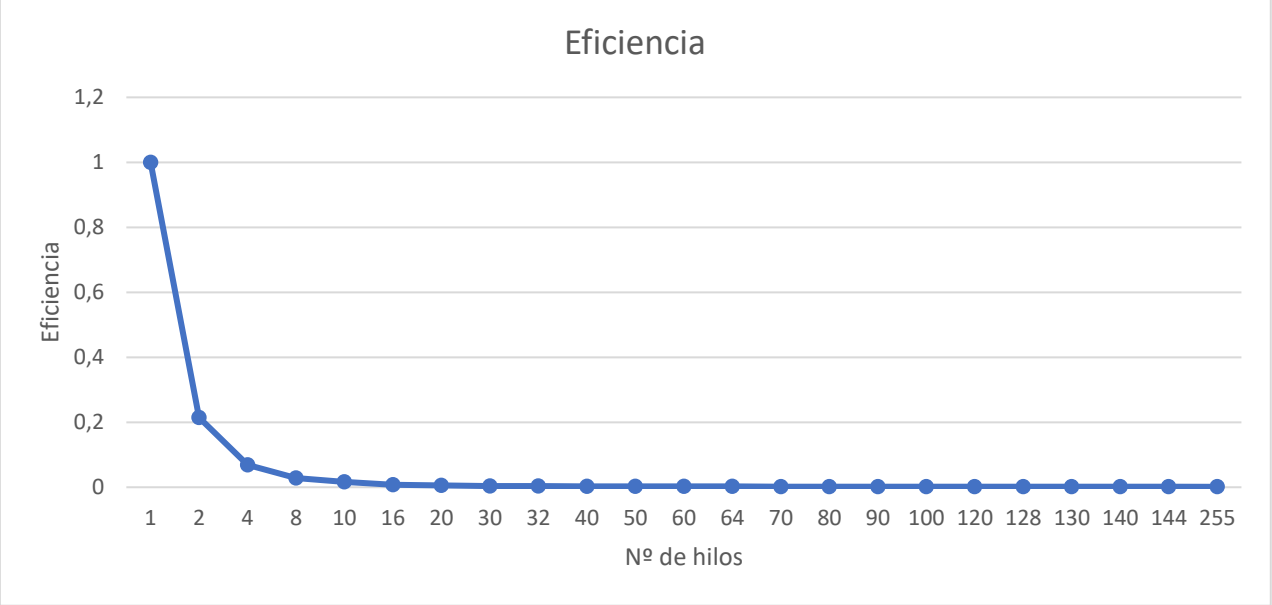

Tabla 7. Variación de la eficiencia respecto al número de hilos.

Tras haber realizado un análisis sobre el comportamiento reflejado en las gráficas se extraen las siguientes conclusiones.

Basándose en la gráfica de la Tabla 4, se puede apreciar como la cantidad de transacciones crece hasta llegar a un máximo absoluto y decae hasta, aparentemente, estabilizarse y sufrir grandes cambios. A esta conclusión se le suma la gráfica de la Tabla 6, la del coste, en la que puede apreciarse un comportamiento casi lineal del número de transacciones respecto al número de hilos. Con la mirada puesta en estas dos gráficas, se plantea la solución que se ha llegado a un punto de saturación ya que conforme se incrementa el número de hilos y el trabajo computacional de cada uno, no se aprecia un crecimiento notable del número de transacciones.

Análogamente se puede apreciar, centrando la mirada en las gráficas de las Tablas 5 y 7, en la que el incremento y eficiencia cae bruscamente y se mantiene prácticamente constante hasta el final. Esto, hace pensar que se alcanza el punto máximo con pocos hilos y conforme se van incrementando no se aprecia una mejora en el resultado de las ejecuciones.

Se puede concluir que el sistema no transaccional es asombrosamente rápido ya que, al no respetar la definición de transacción, la recuperación de datos es inmediata sin tiempos de esperas para respetar la consistencia de datos. Por tanto, el mismo sistema, pero transaccional, da pie a pensar que en el mismo periodo de tiempo se realizará mucho menos trabajo computacional.

2.2 Conclusiones de Transactional read-only.

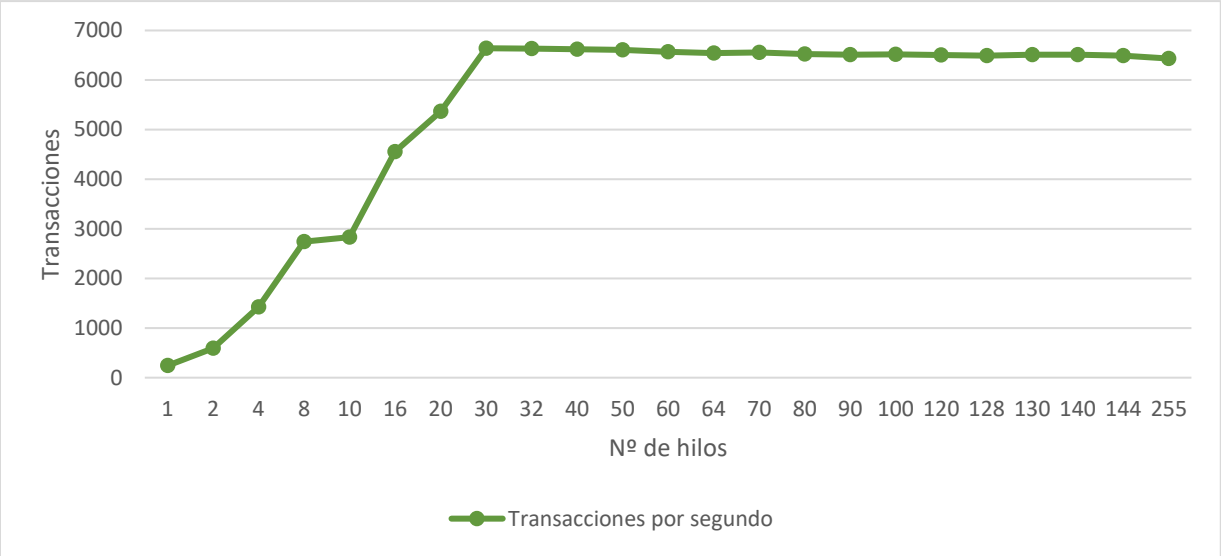

Tabla 8. Transacciones por segundo para ejecuciones con distintos hilos.

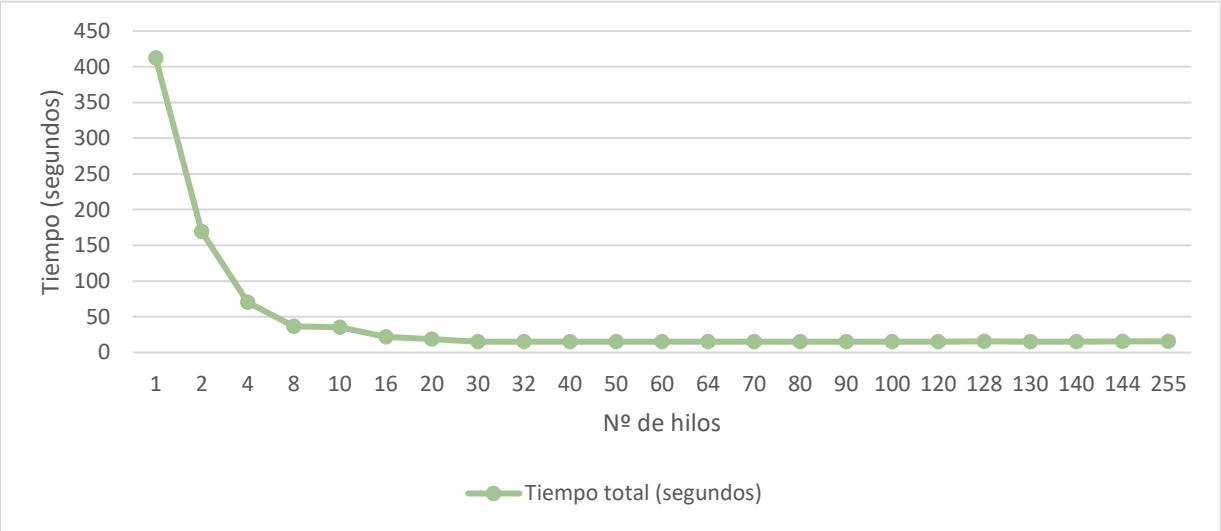

Tabla 9. Tiempo total requerido para ejecuciones con distintos hilos.

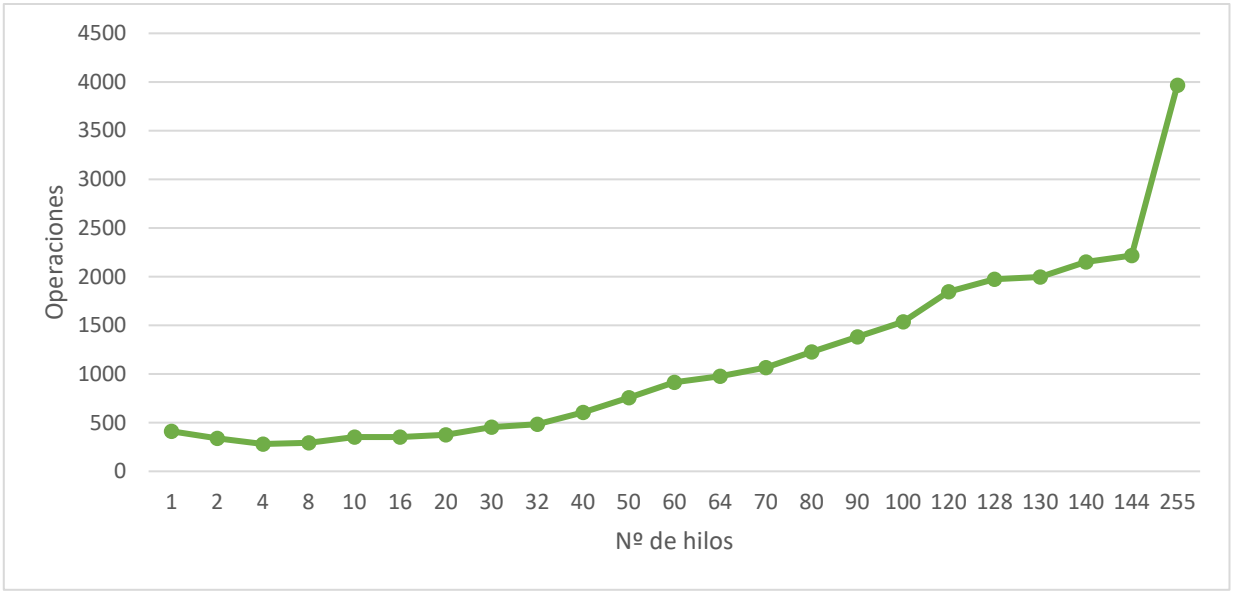

Tabla 121. Coste de ejecución por nº de hilos.

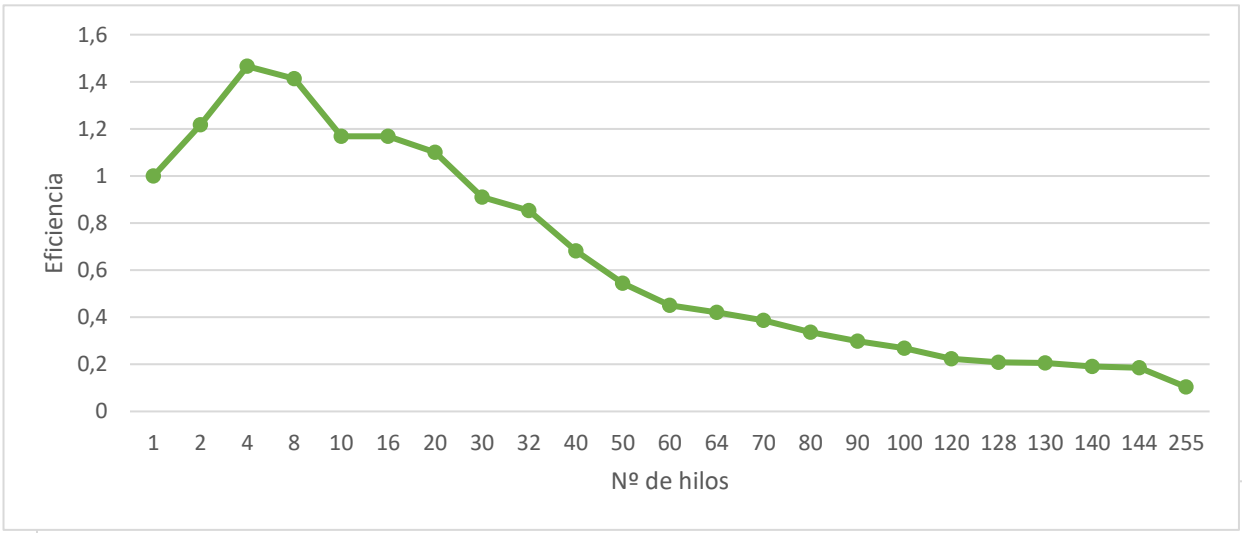

Tabla 112. Eficiencia de las ejecuciones en función del nº de hilos.

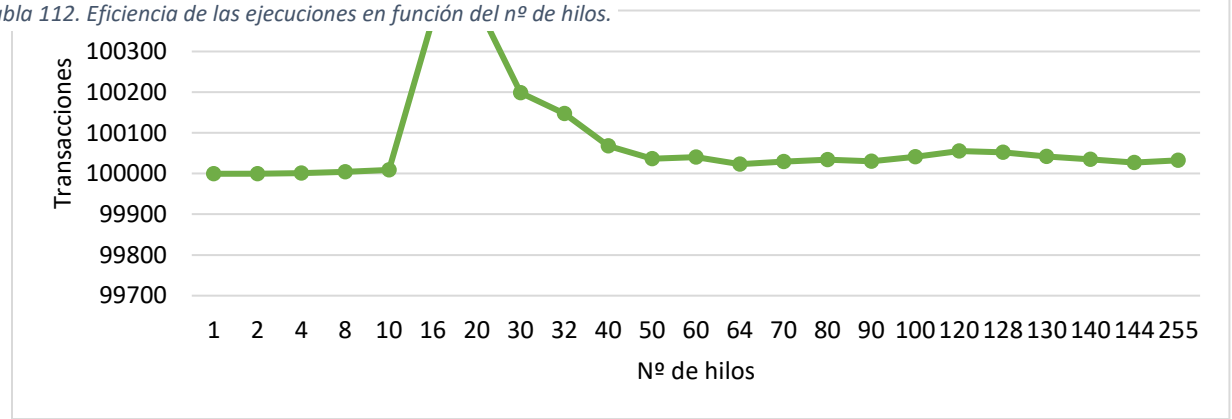

Tabla 10. Transacciones totales realizadas para las ejecuciones con diferentes hilos.

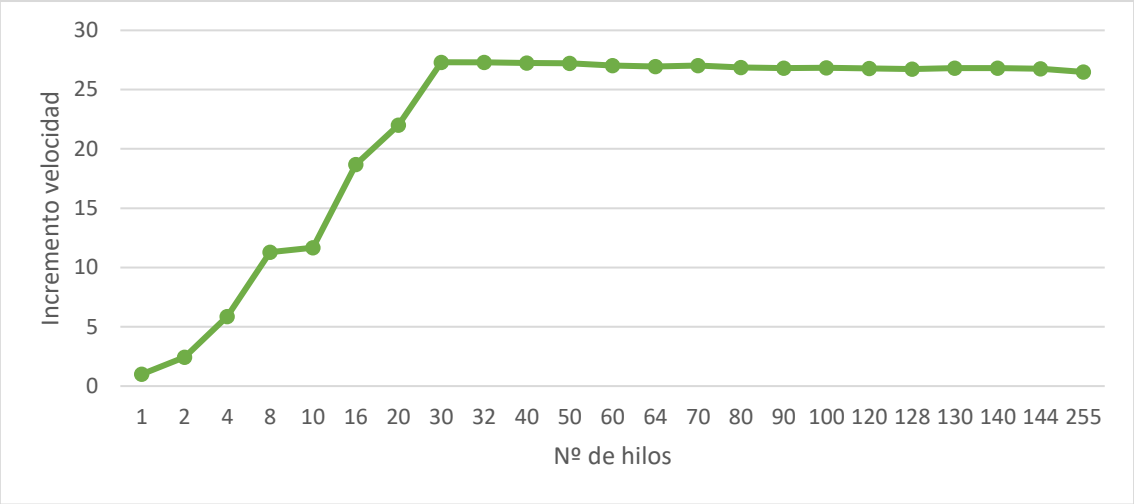

Tabla 13. Speed-up. Incremento de velocidad respecto de la ejecución monohilo.

Tras haber plasmado los resultados de las ejecuciones que se muestra al inicio de este capítulo segundo, se pueden extraer diferentes conclusiones y razonamientos.

En primer lugar, se observar una clara diferencia en lo que refiere a tiempo y transacciones totales de las Tablas 9 y 10, respectivamente. Tal y como se estimaba al final del apartado de las ejecuciones no transaccionales, ha sido enorme la diferencia tanto en tiempo como en operaciones realizadas. Esto se debe a que se respeta la definición de transacción que, si se le suma la cantidad de datos que existen en la base de datos, la consulta requiere de mucho más tiempo.

En segundo lugar, cabe destacar o marcar un punto de inflexión el cual divide de una manera conceptual en dos la comprensión que se puedan extraer de las gráficas. Para ello, se utilizará la Tabla 11 que trata acerca de cuán eficiente ha sido la ejecución para una cantidad de hilos determinada. Con el fin de refrescar la memoria, la eficiencia mide cuánto ha trabajado un procesador a lo largo de la ejecución e indica si tal y como está montado el sistema, es escalable para poder realizar ejecuciones con más recursos -hilos en este caso de estudio-. Se toma como valor de diferencia una eficiencia superior a 0,5. Por tanto si se aprecia Tabla 11, ejecuciones con eficiencia superior a 0,5 serán desde un hilo hasta 50. El resto indica que el sistema cada vez es menos escalable.

Referido a los once primeros casos de prueba, asombran los resultados que se obtienen. Centrándose en el *Speed-up*, rápidamente con aumentar cuatro hilos la velocidad de ejecución ya se sitúa en cinco veces mejor que la ejecución con un solo hilo. Conforme se van incrementando el número de hilos, más se nota este incremento de velocidad hasta que se llega a los 30 hilos donde se alcanza el incremento máximo de velocidad o como se ha llamado también en esta memoria, punto de saturación. De la ejecución con 30 hilos hasta los 255 existe una disminución paulatina del incremento de velocidad, casi imperceptible.

Esta última conclusión se ve reforzada con las gráficas de las Tablas 10 y 11, las transacciones totales y el coste de cada una de las ejecuciones, respectivamente. Es aquí donde se aprecia hasta las ejecuciones con 20 hilos que las transacciones realizadas son

más y el coste asociado no incrementa de manera considerable. Es partir de 30 donde las cosas cambian, las transacciones totales decrementan de una manera considerable y el coste empieza a subir a la vez que los hilos. Cabe comentar que la gráfica de la eficiencia da soporte a estos resultados ya que, es en el momento de la ejecución con cuatro hilos donde el coste es el mínimo absoluto y el número de transacciones totales no se desmarca de las obtenidas por dos y ocho hilos, el valor de la eficiencia se dispara hasta alcanzar el máximo absoluto de la tabla.

Para finalizar el análisis, a partir de los 30 hilos, se nota claramente cómo la eficiencia va en descenso y rebasando la frontera de la escalabilidad (0,5), el *speed-up* y las transacciones totales permanecen casi sin cambios bruscos. Análogamente, el coste se dispara para poder hacer frente a la cantidad de hilos que se requieren para la ejecución.

2.3 Conclusiones de Transactional read-Write.

Para este apartado, indicar que los resultados son prácticamente iguales que el apartado anterior sólo que los números diferentes ya que aquí se realizan ejecuciones de lectura y escritura y es conocido el tiempo invertido para realizar una transacción con escritura incluida. Se pueden extrapolar las mismas conclusiones ya que, como se observa a continuación, las gráficas presentan un comportamiento casi idéntico.

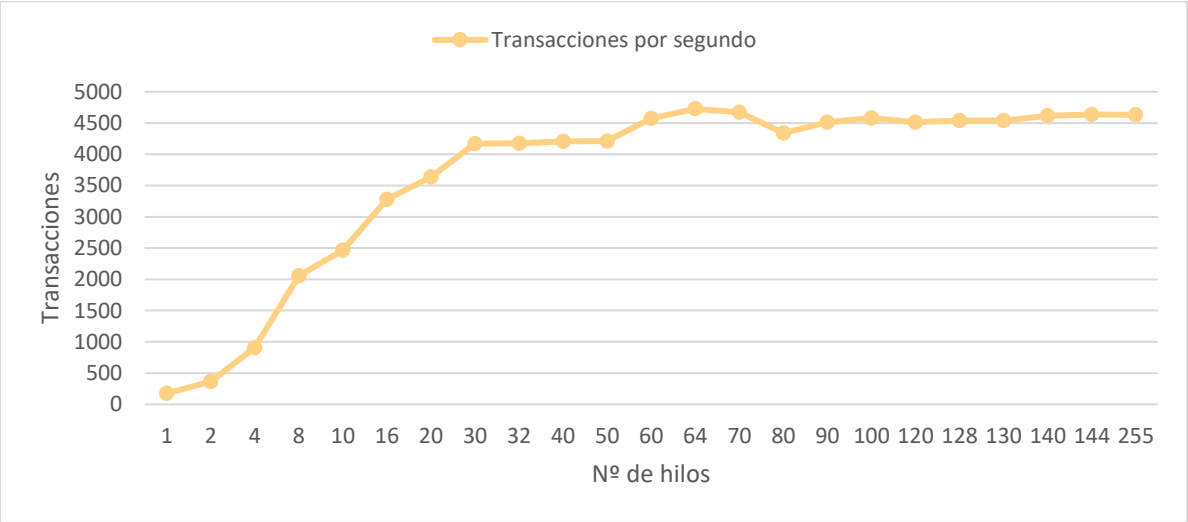

Tabla 14. Transacciones por segundo en función de la cantidad de hilos.

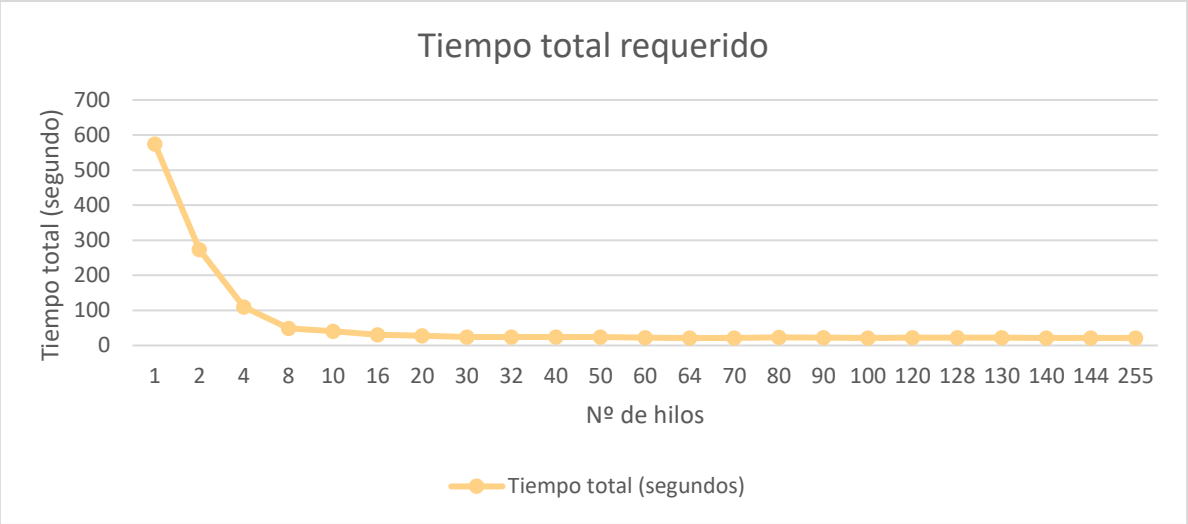

Tabla 15. Tiempo requerido para completar las ejecuciones en función del número de hilos.

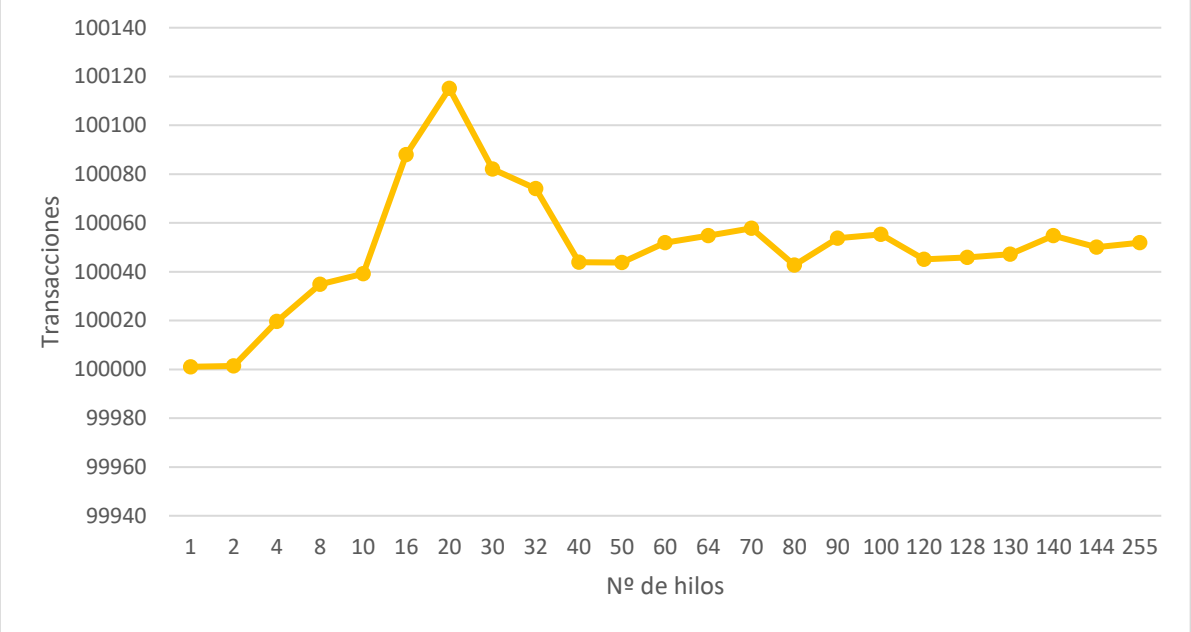

Tabla 16. Transacciones totales en función del número de hilos.

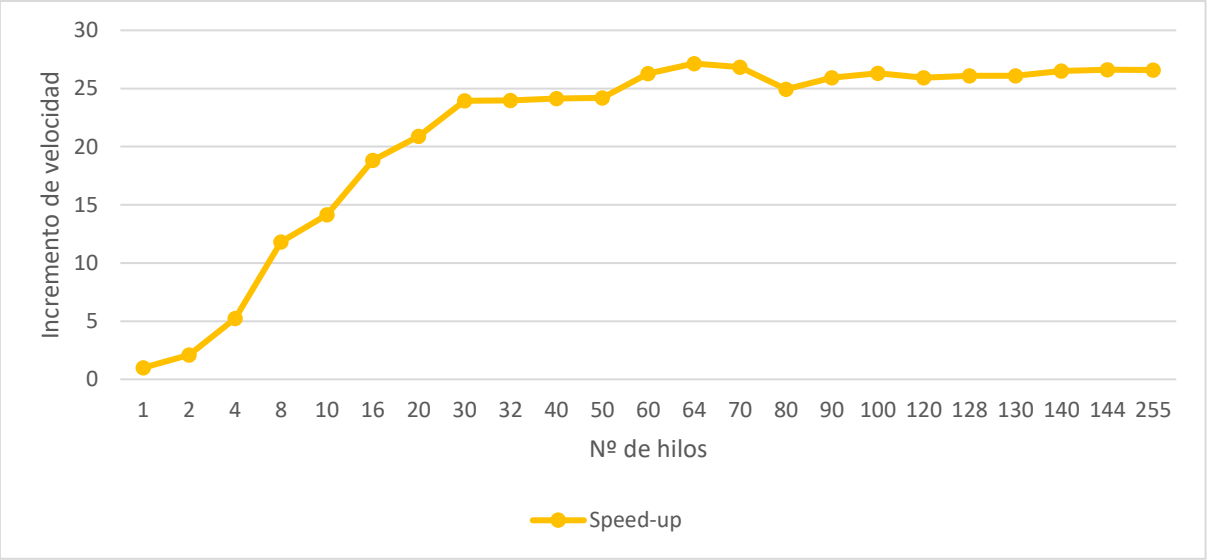

Tabla 17. Speed-up. Incremento de la velocidad de ejecución con una cantidad de hilos frente a la ejecución monohilo.

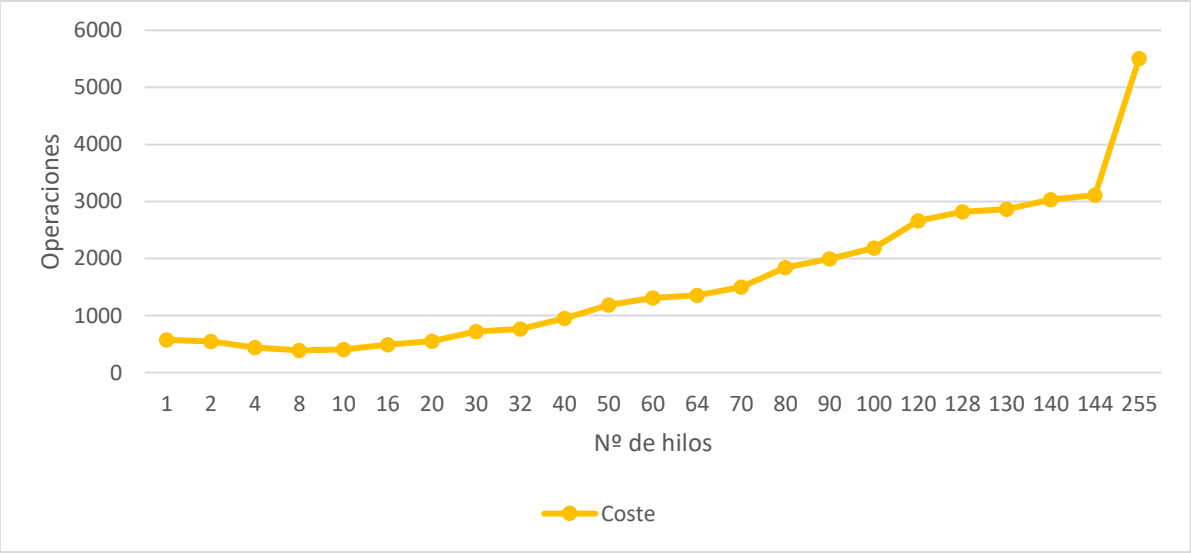

Tabla 18. Coste de ejecuciones en función de cantidad de hilos.

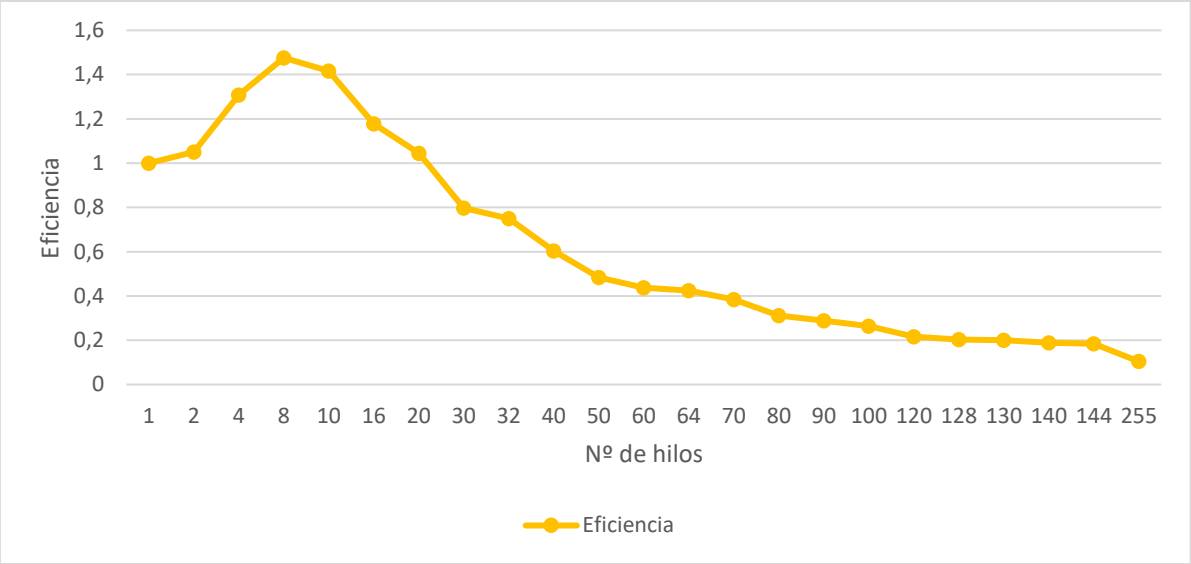

Tabla 19. Eficiencia de tiempo de ejecución respecto al número de hilos y el tiempo secuencial.

3. Conclusiones y comparativa de las tres tesituras.

En el último de los capítulos de esta memoria, se realiza una comparativa de los tres tipos de ejecuciones, exactamente acerca de las transacciones totales, *Speed-up*, Coste y Eficiencia. Con esto se pretende unificar las conclusiones tratadas por separado y así poder tener una visión perimétrica del caso de estudio.

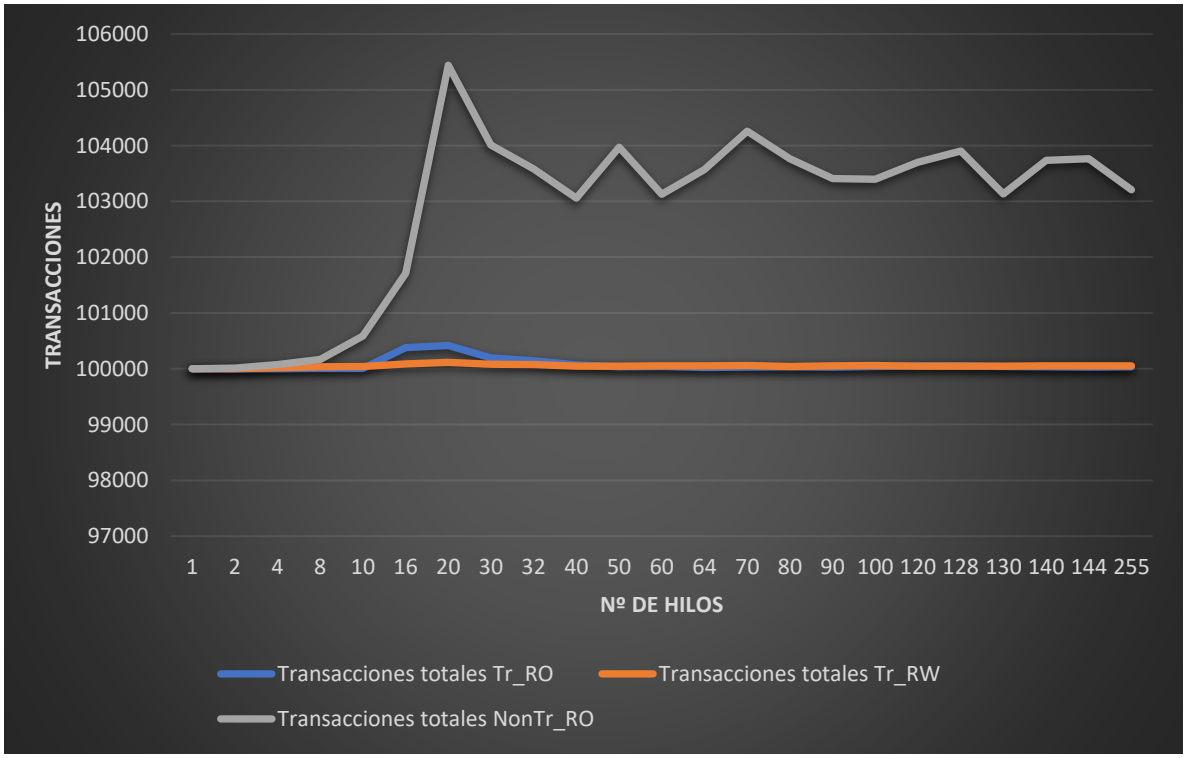

Tabla 20. Comparativa de las Transacciones totales para los tres casos de estudio.

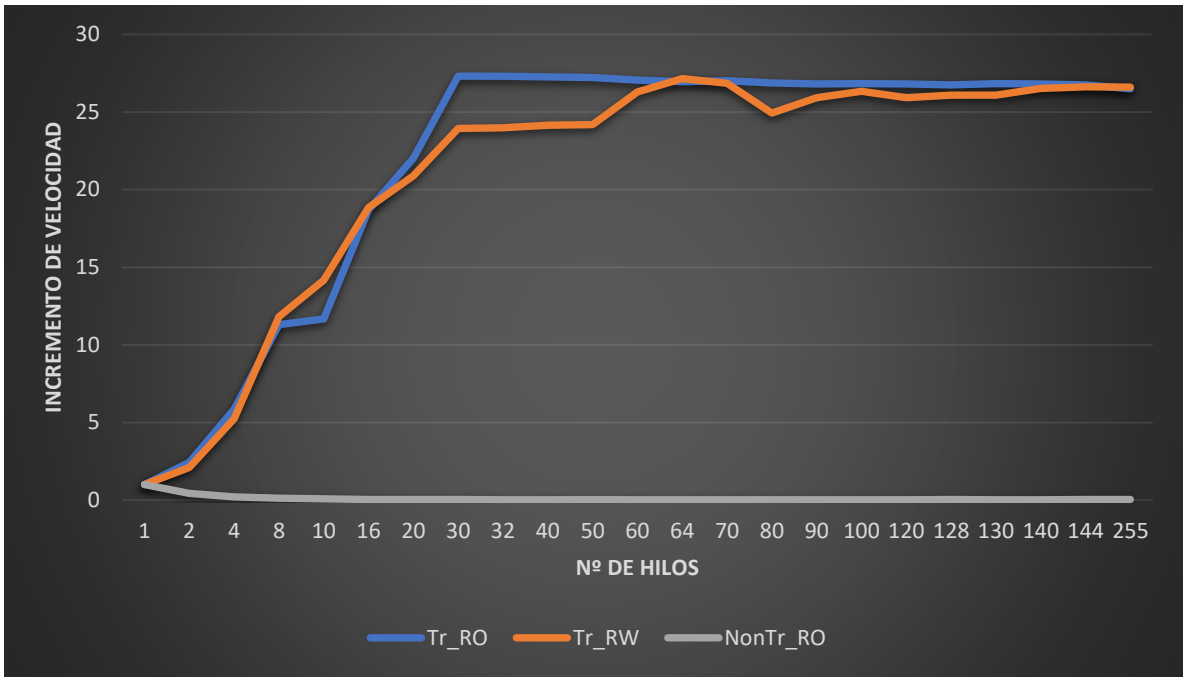

Tabla 21. Comparativa del Speed-up para los tres casos de estudio.

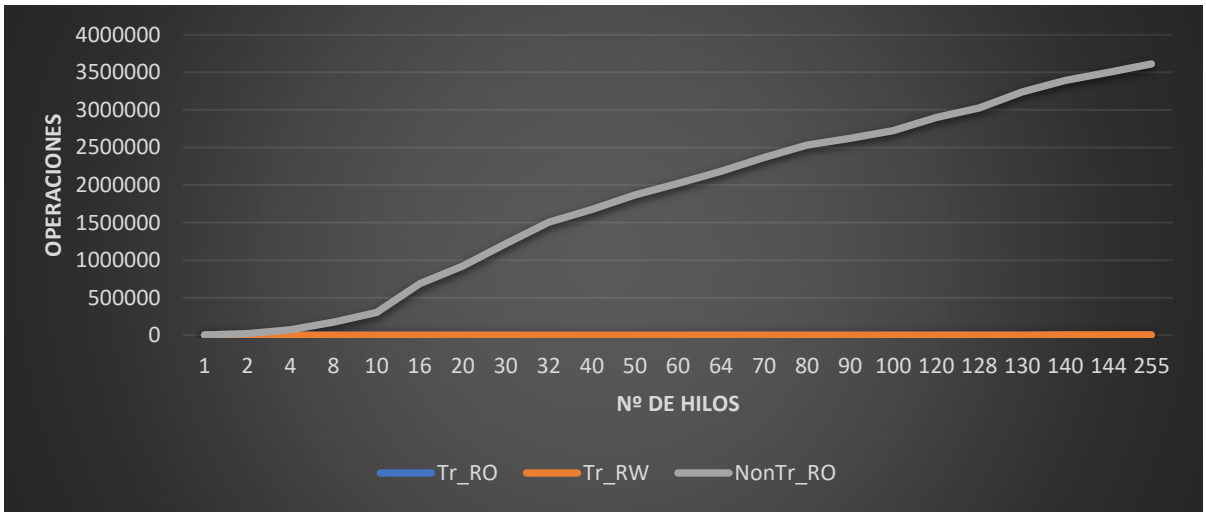

Tabla 22. Comparativa del Coste para los tres casos de estudio.

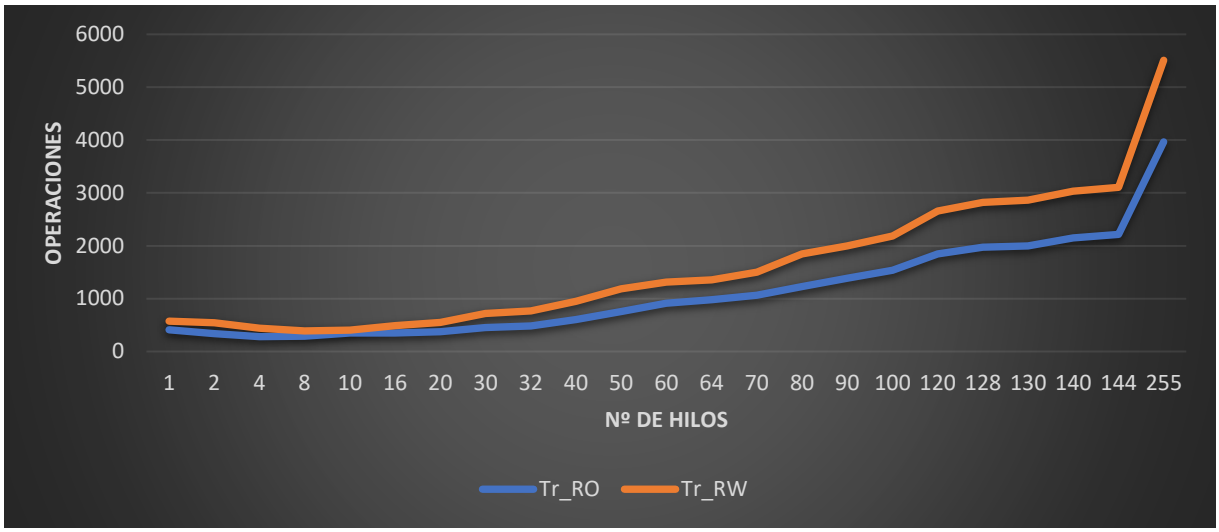

Tabla 23. Comparativa del Coste de las ejecuciones transaccionales.

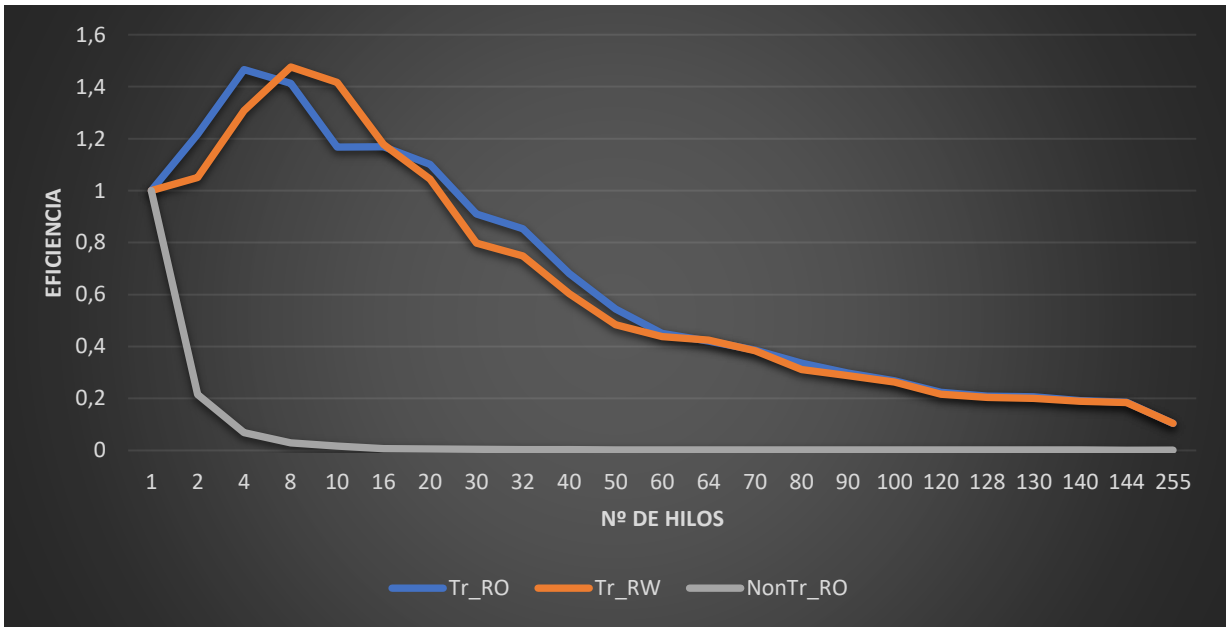

Tabla 24. Comparativa de la Eficiencia para los tres casos de estudio.