Análisis de datos reto 3 HPC

Presentado por:

Cristian Camilo Perilla Castaño

Daniel Villada Moncada

Luis Miguel Ortiz Alarcón

Carlos A. Vivas Reyes

Presentado a:

Ramiro Andres Barrios Valencia

Universidad Tecnológica de Pereira

Facultad de Ingenierías

High Performance Computing

Pereira

2020

**Introducción.**

Este documento presenta un análisis de datos de tiempo obtenidos gracias a la cronometración de códigos desarrollados en c++ de multiplicación de matrices, un mismo programa que ha sido programado de tres formas diferentes, la primera de forma tradicional secuencial, la segunda utilizando concurrencia por hilos con la librería pthreads y la tercera utilizando concurrencia por procesos con fork.

En esta nueva entrega veremos si el cache-line mejora el rendimiento de los programas concurrentes (hilos y procesos) de multiplicación de matrices.

**CARACTERÍSTICAS DEL COMPUTADOR**

|  |  |
| --- | --- |
| **Procesador** | AMD Ryzen R5 2600 overclockeado a 4Ghz |
| **Ram** | 2 modulos de 8gb c/u DDR4 3200MHz |
| **Disco duro** | SSD kingston UV400 hasta 550MB/s lectura y 350MB/s escritura |
| **S.O** | Ubuntu 18.04lts |
| **Número de cpu** | 6 núcleos 12 hilos |

**DESCRIPCIÓN DEL PROCESO**

Basados en el primer reto, en este se va a reutilizar los datos obtenidos y a adicionar unos nuevos datos generados por el código de multiplicación de matrices pero esta vez utilizando transpuesta de la matriz para lograr cache-line, se va a analizar la información resultante mediante tablas y gráficos para luego generar conclusiones.

**TABLAS DE TIEMPOS**

**Tiempos en secuencial**

|  |  |  |
| --- | --- | --- |
| Tamaño Matriz | Secuencial | total celdas |
| 10 | 0 | 100 |
| 100 | 0,15625 | 10000 |
| 500 | 1,65625 | 250000 |
| 1000 | 13,4219 | 1000000 |
| 1500 | 46,0781 | 2250000 |
| 2000 | 110,281 | 4000000 |
| 2500 | 267,906 | 6250000 |
| 3000 | 482,281 | 9000000 |
| 3500 | 759,948 | 12250000 |

**Tiempo en hilos**

|  |  |
| --- | --- |
| Hilos |  |
| Tamaño Matriz | 12 nucleos |
| 10 | 0 |
| 100 | 0,0083896 |
| 500 | 0,253582 |
| 1000 | 1,77788 |
| 1500 | 5,86188 |
| 2000 | 13,6723 |
| 2500 | 34,1269 |
| 3000 | 63,9952 |
| 3500 | 105,309 |

**Tiempos speedup hilos**

|  |  |
| --- | --- |
| Speedup rate hilos normal |  |
|  | 12 nucleos |
| 10 | 0 |
| 100 | 18,62424907 |
| 500 | 6,531417845 |
| 1000 | 7,54938466 |
| 1500 | 7,860635155 |
| 2000 | 8,066016691 |
| 2500 | 7,850288189 |
| 3000 | 7,53620584 |
| 3500 | 7,216363274 |

**Tiempos hilos transpuesta**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hilos Transpuesta |  |  |  |  |
| Tamaño Matriz | Tiempo transpuesta | 12 nucleos | Tiempo total | total celdas |
| 10 | 0,00000460 | 0,0009006 | 0,0009055 | 100 |
| 100 | 0,0001633 | 0,0089359 | 0,0090992 | 10000 |
| 500 | 0,0036964 | 0,242428 | 0,246125 | 250000 |
| 1000 | 0,0148165 | 1,72809 | 1,7429 | 1000000 |
| 1500 | 0,0340505 | 5,70017 | 5,73422 | 2250000 |
| 2000 | 0,0641815 | 13,5123 | 13,5765 | 4000000 |
| 2500 | 0,104936 | 26,0736 | 26,1785 | 6250000 |
| 3000 | 0,149436 | 44,9188 | 45,0682 | 9000000 |
| 3500 | 0,23255 | 71,2916 | 71,5241 | 12250000 |

**Tiempos speedup hilos transpuesta**

|  |  |
| --- | --- |
| Speedup rate hilos transpuesta |  |
|  | 12 nucleos |
| 10 | 0,00000000 |
| 100 | 17,17183928 |
| 500 | 6,72930422 |
| 1000 | 7,70090080 |
| 1500 | 8,03563519 |
| 2000 | 8,12293301 |
| 2500 | 10,23381783 |
| 3000 | 10,70113739 |
| 3500 | 10,62506204 |

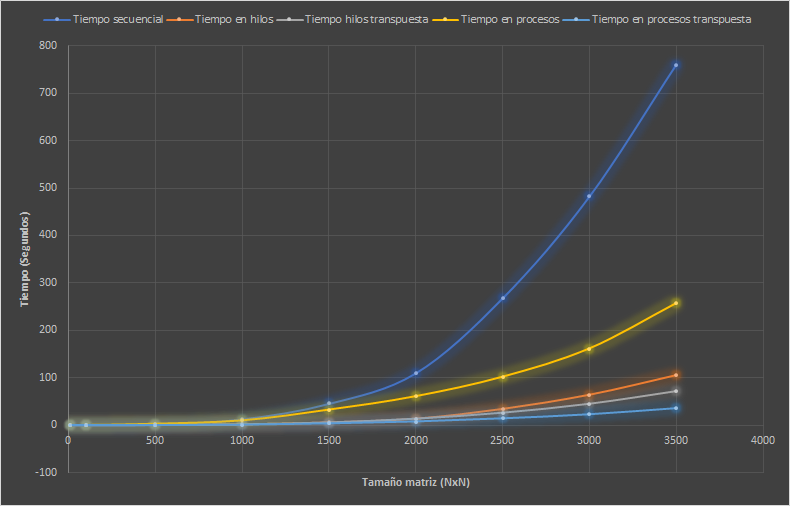
**Tiempo en procesos**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PROCESOS |  |  |  |  |  |  |
| Tamaño Matriz | 12 nucleos | 8 núcleos | 6 núcleos | 4 núcleos | 2 núcleos | total celdas |
| 10 | 0,033 | 0,032 | 0,03 | 0,032 | 0,032 | 100 |
| 100 | 0,375 | 0,368 | 0,36 | 0,349 | 0,411 | 10000 |
| 500 | 3,122 | 3,253 | 2,963 | 2,979 | 3,933 | 250000 |
| 1000 | 10,148 | 10,304 | 9,16 | 9,372 | 16,31 | 1000000 |
| 1500 | 32,817 | 36,281 | 31,889 | 41,354 | 79,859 | 2250000 |
| 2000 | 61,119 | 74,411 | 63,114 | 91,297 | 184,38 | 4000000 |
| 2500 | 102,14 | 133,902 | 112,831 | 162,569 | 328,736 | 6250000 |
| 3000 | 161,511 | 225,045 | 196,825 | 286,565 | 575,236 | 9000000 |
| 3500 | 256,89 | 374,357 | 307,493 | 449,245 | 891,284 | 12250000 |

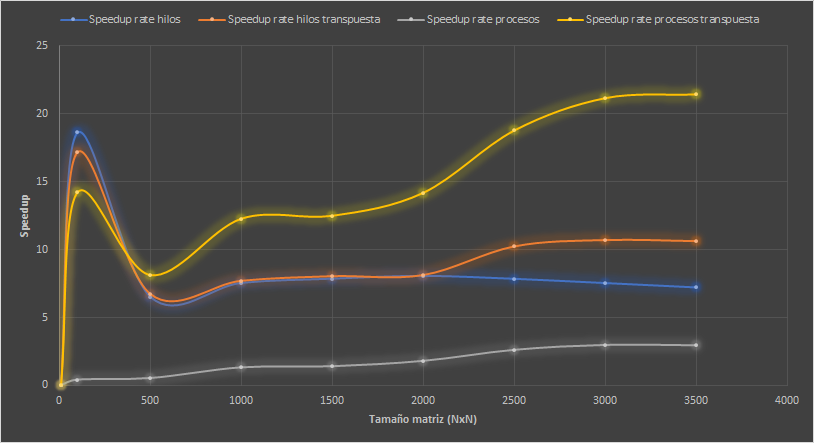
**Tiempo speedup procesos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SPEED UP RATE |  |  |  |  |  |
|  | 12 nucleos | 8 núcleos | 6 núcleos | 4 núcleos | 2 núcleos |
| 10 | 0 | 0 | 0 | 0 | 0 |
| 100 | 18,62424907 | 17,02069717 | 18,65760753 | 12,92561464 | 11,33009927 |
| 500 | 6,531417845 | 4,234287438 | 5,078356161 | 3,702052138 | 1,919837998 |
| 1000 | 7,54938466 | 5,222204065 | 5,77790501 | 4,030394845 | 2,009940459 |
| 1500 | 7,860635155 | 4,929145954 | 6,253508243 | 4,200641791 | 2,051087905 |
| 2000 | 8,066016691 | 5,347502049 | 6,312773691 | 4,247817948 | 2,094196197 |
| 2500 | 7,850288189 | 5,390126572 | 7,192184614 | 4,802981762 | 2,364676288 |
| 3000 | 7,53620584 | 5,150852332 | 6,965986218 | 4,67294854 | 2,295756278 |
| 3500 | 7,216363274 | 4,957745376 | 6,849155063 | 4,541449539 | 1,999237081 |

**GRÁFICA DE TIEMPO SECUENCIAL/HILOS NORMAL/HILOS TRANSPUESTA/PROCESOS NORMAL/PROCESOS TRANSPUESTA**

****

**GRÁFICAS SPEED-UP RATE HILOS NORMAL (12 NÚCLEOS), TRANSPUESTA HILOS (12 NUCLEOS) PROCESOS NORMAL (12 NÚCLEOS) TRANSPUESTA PROCESOS (12 NÚCLEOS)**

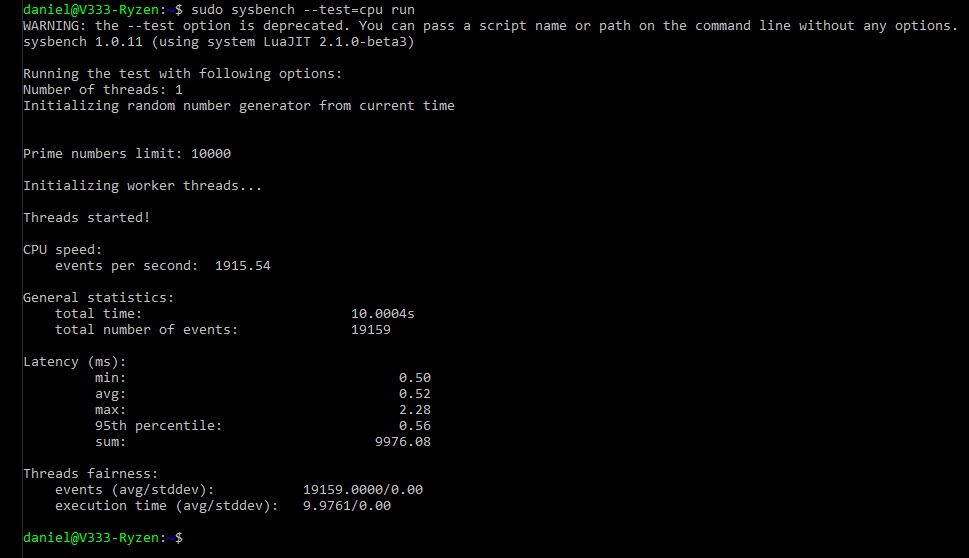
****

**Tabla de tiempos comparativa entre la speedup de hilos transpuesta y procesos transpuesta**

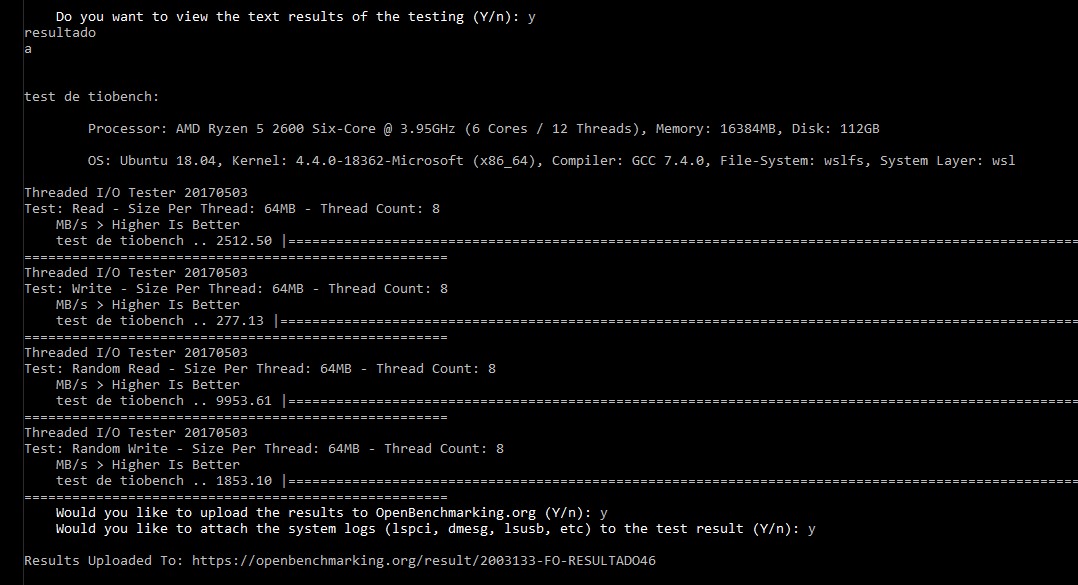
|  |  |  |  |
| --- | --- | --- | --- |
| Speedup Rate Hilos Transpuesta vs Procesos Transpuesta |  |  |  |
| Tamaño Matriz | Tiempo Procesos Transpuesta | Tiempo Hilos Transpuesta | Tiempos totales |
| 10 | 0,0020000 | 0,0009055 | 0,45275 |
| 100 | 0,0110000 | 0,0090992 | 0,8272 |
| 500 | 0,2040000 | 0,246125 | 1,206495098 |
| 1000 | 1,0940000 | 1,7429 | 1,593144424 |
| 1500 | 3,6900000 | 5,73422 | 1,55398916 |
| 2000 | 7,7780000 | 13,5765 | 1,745500129 |
| 2500 | 14,2750000 | 26,1785 | 1,833870403 |
| 3000 | 22,7910000 | 45,0682 | 1,977456013 |
| 3500 | 35,4490000 | 71,5241 | 2,017661993 |

**BENCHMARKS**

**Sysbench:**



**Phoronix test suite:**



<https://openbenchmarking.org/result/2003133-FO-RESULTADO46>

**CONCLUSIONES**

* Los códigos que puedan ser programados concurrentemente son definitivamente mejores ya que aprovechan más los recursos de la máquina y de esta forma optimizando todos los aspectos del programa.
* Se puede evidenciar que dependiendo del algoritmo que se quiera paralelizar, se puede optimizar más por medio de procesos o por hilos, es notorio que existen muchos aspectos que pueden alterar los resultados como la forma de leer y escribir datos, si utilizan o no memoria compartida, si su concurrencia es por hilos o por procesos etc.

* Hasta ahora, la forma mas optima de aplicar concurrencia a la multiplicación de matrices la hemos logrado con la implementación de procesos utilizando memoria compartida y cache-line, vemos que estos tres aspectos juntos reducen mucho el tiempo de cálculo y tiene sentido ya que si bien la implementación de solo concurrencia por procesos no fue tan efectiva como por hilos, compartir memoria es una de las formas más eficientes de comunicar procesos (a diferencia de por ejemplo las tuberías que son procesos más lentos), y en el caso de cache-line hay una ventaja muy importante y es que el bloque completo de datos que se carga en el procesador para ser tratado es más rentable que de la forma tradicional ya que no tiene que estar cargando datos de la memoria caché cada vez que haya un cambio de columna lo que genera una mayor latencia y que en grandes cantidades de datos se ve reflejado en mayor tiempo de ejecución, en nuestro como lo revela la tabla de speedup rate de hilos con matriz transpuesta vs procesos con matriz transpuesta de hasta 2 veces más velocidad de procesamiento.

* en las gráficas de speedup rate se observa que las matrices pequeñas, que no superan las 10.000 celdas se alcanzan a guardar en las caches del procesador, pero cuando superamos ese límite como es el caso 250.000 celdas hay un descenso importante en la velocidad de procesamiento y esto es debido a que se hace uso de la memoria principal (RAM) para poder cargar los datos a ser tratados.