**Caso de estudio multiplicación de Matrices: implementación secuencial Implementación con optimización de CPU por Traspuesta**

#### Oscar M. Giraldo Herrera

Universidad Tecnológica De Pereira

Facultad De Ingenierías

Andrés Ramiro Barrios

#### Octubre 2021

# TABLA DE CONTENIDO

Contenido

[TABLA DE CONTENIDO 2](#_Toc90324726)

[RESUMEN 3](#_Toc90324727)

[ABSTRACK 4](#_Toc90324728)

[INTRODUCCIÓN 6](#_Toc90324729)

[MARCO CONCEPTUAL 8](#_Toc90324730)

[MATRICES 8](#_Toc90324731)

[MULTIPLICACION DE MATRICES CUADRADAS 9](#_Toc90324732)

[¿COMO SE REALIZA LA MULTIPLICACION DE DOS MATRICES? 10](#_Toc90324733)

[PROCESOS 10](#_Toc90324734)

[Procesos en la Informática. 10](#_Toc90324735)

[PROCESOS FORK() 11](#_Toc90324736)

[¿QUE ES SOFTWARE? 11](#_Toc90324737)

[¿QUE ES LA SUPERCOMPUTACION? 11](#_Toc90324738)

[¿QUE ES LA COMPUTACION SECUENCIAL? 12](#_Toc90324739)

[COMPUTACION SECUENCIAL 12](#_Toc90324740)

[COMPUTACION PARALELA 14](#_Toc90324741)

[¿QUE ES PARALELISMO? 14](#_Toc90324742)

[¿QUÉ ES LA COMPUTACION PARALELA? 16](#_Toc90324743)

[MEMORIA COMPARTIDA 17](#_Toc90324744)

[¿QUÉ ES HIGH PERFORMANCE COMPUTING? 18](#_Toc90324745)

[¿QUÉ SON LOS HILOS Y LA CONCURRENCIA DE MULTIPLES HILOS? 18](#_Toc90324746)

[MARCO CONTEXTUAL 20](#_Toc90324747)

[DESARROLLO 20](#_Toc90324748)

[PRUEBAS 26](#_Toc90324749)

[CONCLUSIONES 27](#_Toc90324750)

# 

# RESUMEN

El presente documento tiene como objetivo mostrar el proceso de desarrollo con la computación de alto desempeño ha demostrado mejorar exponencialmente tiempos de ejecución de ejecución. La optimización de algoritmos no solo se da en su ejecución sino en el proceso antes del mismo, lo que quiere decir que durante la escritura o digitación del algoritmo; no solo la complejidad computacional, sino como se opera cada uno de los procesos u operaciones dentro de dicho sistema.

. En el desarrollo de esta actividad o este caso de estudio se incursiona dentro de esta nueva perspectiva y se plantea una nueva posibilidad, la cual es la multiplicación de matrices desde la eliminación de las columnas por medio de transposición de valores, de esta manera, se reducen los costos de lectura de manera ligera, en este proceso el cual es masivo.

PALABRAS CLAVE: high performance computing, desarrollo, programación en C, Python, computación secuencial, computación paralela, Forks, Procesos, sistema computacional, CPU, lenguaje de programación, matrices

# ABSTRACK

*The objective of this document is to show the development process with high-performance computing that has shown to exponentially improve execution times. The optimization of algorithms not only occurs in its execution but in the process before it, which means that during the writing or typing of the algorithm; not only the computational complexity, but how each of the processes or operations within said system operates.*

*. In the development of this activity or this case study, we enter into this new perspective and a new possibility arises, which is the multiplication of matrices from the elimination of the columns by means of transposing values, in this way, they reduce reading costs slightly, in this process which is massive.*

KEY WORDS: high performance computing, development, C programming, Python, sequential computing, parallel computing, Forks, Processes, computer system, CPU, programming language, arrays

# INTRODUCCIÓN

En la computación se abordan maneras de solucionar problemas relacionados a aspectos  
varios en contexto a un área de conocimiento, una tecnología o un entorno específico, donde  
ciertos problemas son de alta reiteración, como cálculos matemáticos, análisis de textos o  
simulaciones de procesos. Para dichos problemas se generan códigos que iterativamente son  
mejorados por diferentes programadores y se globalizan en formatos de módulos, interfaces,  
librerías y demás; el uso de dichos códigos agiliza el tiempo de desarrollo de cualquier  
problema y permite la estandarización de procesos.

Dichos códigos pueden ser trasladados al área de la computación de alto desempeño, donde el uso de herramientas como hilos y forks se da mediante librerías que facilitan el contacto  
con el sistema operativo, sin embargo, se ofrecen mayores niveles de abstracción mediante  
frameworks y API, como es el caso actual.

Gracias a la existencia de estos sistemas computacionales, podemos realizar simulaciones de sistemas más complejos, que de forma empírica tardaríamos años en conocer resultados, evaluarlos, compararlos, y lograr entender el funcionamiento de aquello sobre lo que queremos saber un poco más.

Para los procesos de paralelismo mediante hilos se propone el uso de OpenMP, un API de alto nivel que facilita el paralelismo de procesos mediante directivas y declaraciones  
específicas, mejorando la experiencia de desarrollo.

En matemática, una matriz es un arreglo bidimensional de números. Dado que puede definirse tanto la suma como el producto de matrices, en mayor generalidad se dice que son elementos de un anillo, Es larga la historia del uso de las matrices para resolver ecuaciones lineales. Un importante texto matemático chino que proviene del año 300 a. C. a 200 a. C., *Nueve capítulos sobre el Arte de las matemáticas* (*Jiu Zhang Suan Shu*), es el primer ejemplo conocido de uso del método de matrices para resolver un [sistema de ecuaciones simultáneas](https://es.wikipedia.org/wiki/Sistema_lineal_de_ecuaciones). Ref <https://es.wikipedia.org/wiki/Matriz_(matem%C3%A1ticas>). En promedio un ser humano que haya estudiado matrices y su aplicación, puede tardar entre 1 y 2.5 minutos en resolver la multiplicación de dos matrices pequeñas de un tamaño 2x2 o 3x3 con datos enteros de no más de dos dígitos, sin utilizar calculadora. Ref <https://www.youtube.com/watch?v=4HgEfOukm78> . Si en nuestro ámbito científico quisiéramos investigar algún concepto que requiera la multiplicación de matrices de gran tamaño, por ejemplo, por encima de los mil datos, y además con números de hasta 4 y 5 dígitos, tardaríamos muchísimo tiempo en recopilar la información necesaria para nuestra investigación, es aquí donde los sistemas computacionales entran a resolver por nosotros los cálculos matemáticos, de forma precisa y rápida.

Para esta práctica, se analiza un nuevo paradigma de optimización. En le cual yo no se implementarán las técnicas de concurrencia o paralelismo, para mejorar el desempeño de un algoritmo, sino que se interioriza en el consumo interno de recursos y como mejorarlo, en este caso para la memoria cache, la cual por su naturaleza es más rápida que la RAM dado que se encuentra en un nivel inmediato debajo del procesador

El acceso a un dato es más rápido o lento según lo cerca o lejos que esté su apuntador original de éste, siendo qué, en el caso de una estructura de datos x[M] como un array, entre más cerca a M esté dicho dato, más lejos se debe apuntar a la caché y más costosa será la operación. En caso contrario, entre más cerca esté al índex 0 dicho apuntador, más rápido se ejecuta la operación, siendo menor su costo

En el presente caso se lleva a otro nivel dicho concepto, centrándose en la multiplicación de  
matrices que, por su naturaleza, debe trabajar con columnas, siendo que se deben hacer  
recorridos redundantes obligatorios para obtener cada dato de una columna para hacer  
producto punto, con esto en cuenta, se podría transponer la matriz para que, en vez de  
recorrer columnas, se recorran filas, entonces, ¿Cómo mejorará el rendimiento de la  
multiplicación de matrices cuadradas si se transponen las columnas?.

# MARCO CONCEPTUAL

A continuación, se definen los términos y conceptos fundamentales para esclarecer y precisar las expresiones, ocupaciones y alcances de las herramientas usadas para el desarrollo de los programas, su ejecución y la recolección de datos de tiempo de ejecución de CPU t

## MATRICES

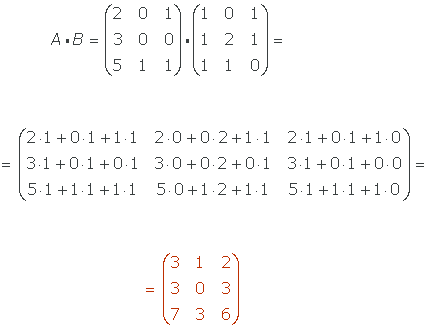
Se define una matriz como una colección de números organizados por una cantidad finita de filas y columnas [3], puede ser una representación de un sistema lineal de ecuaciones, datos ordenados o demás estructuras abstractas. La representación de una matriz puede coincidir con la representación de un vector, en matemáticas o algebra lineal la matriz representa un vector.

Calendario

Descripción generada automáticamente

## MULTIPLICACION DE MATRICES CUADRADAS

Las matrices cuadradas son un subconjunto de las matrices donde se consideran sus filas de la siguiente manera: *N Filas Y M Columnas,* Estas deben ser iguales como se muestra en la siguiente ilustración



En la ilustración anterior se puede evidenciar la multiplicación de dos matrices cuadradas, es decir con igual número de filas que de columnas en este caso 3X3, para este caso se da la multiplicación de matrices ya que poseen la misma cantidad de columnas y filas como se menciona anterior mente, por ende, A y B comparten el mismo valor cuadrado M.

### ¿COMO SE REALIZA LA MULTIPLICACION DE DOS MATRICES?

Dos matrices A y B son multiplicables si el número de columnas de A coincide con el número de filas de B.



El elemento C***ij*** de la matriz producto se obtiene multiplicando cada elemento de la fila i de la matriz A por cada elemento de la columna j de la matriz B y sumándolos.

## ¿QUE ES SOFTWARE?

Es un conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora (Real Academia Española). El software, El Alma del Computador. Así como la esencia del ser humano no está en su cuerpo sino en un algo abstracto llamado alma, es en un elemento intangible llamado software donde radica la mayor parte de la magia que ha convertido al computador en la herramienta más poderosa de nuestro tiempo. (EL TIEMPO, 1996).

## ¿QUE ES LA SUPERCOMPUTACION?

El término supercomputador se refiere al ordenador perteneciente a la gama más alta en cuanto a rendimiento, capacidad de almacenamiento y eficiencia posibles, de todos aquellos disponibles en un momento determinado del tiempo. Por lo tanto, el atributo super debe considerarse como una característica definitoria de un computador que dispone de la más alta tecnología existente en el mercado.

La arquitectura de los ordenadores secuenciales ha mejorado mucho desde las primitivas máquinas de los años 40 y 50. Existen, sin embargo, limitaciones físicas, tales como la velocidad de la luz, que constituyen una barrera natural para hacer del procesamiento secuencial algo ilimitado. El paralelismo ofrece hoy la forma de incrementar la potencia computacional más allá de esas barreras. Procesamiento paralelo es la solución de un único problema mediante el uso de más de un elemento de proceso y de técnicas de segmentación o de empleo repetido de procesadores. Conceptualmente el procesamiento paralelo es algo muy sencillo: subdivisión de un problema en un cierto número de subproblemas. Lo que resulta más complicado es la implementación de ese concepto en un sistema multiprocesador. Para resolver el problema es necesaria la cooperación entre los elementos de proceso o los procesadores. Con el paralelismo y la supercomputación es posible resolver, no solo los problemas que se nos presentan hoy día en fracciones del tiempo empleado hace algunos años, sino también otros más complejos e insolubles en el pasado.

Ref https://www.acta.es/medios/articulos/informatica\_y\_computacion/030009.pdf

REF https://www.youtube.com/watch?v=b5bQdTL0wAg

## API (Aplication Program Interface)

Una Application Program Interface (API) son un conjunto de rutinas, funciones y demás que se pueden usar desde diferentes capas de abstracción. Son una serie de atajos y funcionalidades creadas externamente para llevar a cabo un proceso o consumir un servicio.  
Las API reciben conexiones externas y retornan salidas haciendo un procedimiento interno  
que no es necesariamente explícito. Las API permiten la comunicación entre varios  
programas sin importar su estructura dado que su importancia se da en el proceso de entrada  
y salida, siendo fundamental la gestión de dichos datos.

## ¿QUÉ ES HIGH PERFORMANCE COMPUTING?

La computación de alto rendimiento (HPC) representa la capacidad de procesar datos y realizar cálculos complejos a velocidades muy altas. Para ponerlo en perspectiva, un equipo portátil o de sobremesa con un procesador de 3 GHz puede realizar unos 3.000 millones de cálculos por segundo. Aunque esto es mucho más rápido de lo que puede lograr cualquier humano, palidece en comparación con las soluciones HPC que pueden realizar cuatrillones de cálculos por segundo.

Uno de los tipos de soluciones HPC más conocidos es el superordenador. Un superordenador contiene miles de nodos de computación que trabajan juntos para completar una o varias tareas. Esto se denomina procesamiento paralelo. Es similar a tener miles de equipos conectados en red, combinando la potencia informática para completar tareas más rápidamente.

## OPENMP

Openmp es una API de programación compartida multiproceso en lenguajes tipo Fortran,  
permite el uso de procesos avanzados de computación de alto desempeño mediante el uso  
de directivas, normas internas o instrucciones no nativas que llevan a cabo procesos  
originalmente complejos de manera ágil y sencilla. Su uso puede implicar un leve incremento  
en el costo de ejecución dado que la implementación manual puede ser más efectiva, pero  
ofrece una alta reducción de tiempo de desarrollo.

# MARCO CONTEXTUAL

Para la realización del presente caso de estudio se utilizaron las instalaciones personales para este caso, en el cual se implementó y ejecuto las pruebas necesarias para la recolección de datos, con un tiempo estimado de 4 horas, que suman el proceso total, desde la creación de los programas necesarios para multiplicar las matrices de forma secuencial y paralela, la ejecución y recolección de datos. Este tiempo parcial de 4 horas fue utilizado para la parte final de la fase investigativa, pues en total se consideran unas 4 horas adicionales de aprendizaje, descritas a continuación.

Las pruebas del programa fueron realizadas en un computador con las siguientes especificaciones:

Procesador: Intel® Core™ i5-4210H CPU @ 1,70 GHz a 2,40GHz.

Memoria secundaria: 8192 MB RAM DDR4

Disco Duro: Toshiba HDW 500GB

# DESARROLLO

Para el desarrollo de la presente práctica, se toman como bases conceptos reforzados anterior mente en las sesiones y practicas pasadas, en las cuales se hizo el caso de estudio con el desarrollo de matrices las cuales se implementan en c++. práctica se interioriza en el uso de OpenMP, el cual es una API para lenguajes tipo Fortran, incluido C, el cual permite el uso de memoria compartida y computación de alto desempeño mediante el uso de directivas y descripciones generales. Al indagar teórica y prácticamente sobre dicha API, se empieza el desarrollo del caso de estudio de la presente, el cual es, volver a la primera implementación del semestre de multiplicación de matrices secuencial, y sobreponerlo al contexto de OpenMP, es decir, paralelizar mediante hilos, pero usando esta herramienta en vez del proceso analógico.

# PRUEBAS

Para comparar el tiempo que le toma al usuario implementar cada estudio de caso, necesita ejecutar dos tipos de pruebas: eficiencia y tiempo de ejecución.

Las pruebas de eficiencia se realizan solo con el propósito de validar la funcionalidad del programa y verificar que el programa está funcionando para realizar la multiplicación de matrices correctamente. La implementación de cada programa se ejecuta con un valor muy pequeño de n (0 - 4) y la salida se compara con el resultado de un programa en línea que se dice que funciona correctamente.

Las pruebas de eficacia fueron útiles para corregir bugs y asegurarse de tener los programas funcionales antes de realizar las pruebas de tiempo de ejecución.

Para las pruebas de tiempo de usuario, se ejecutaron los programas de la siguiente manera: Primero las ejecuciones de la implementación secuencial, una ejecución con n=100, la siguiente con n=200, la siguiente n=400, la siguiente n=800, la siguiente n=1600 y finalmente n=3200, después el mismo procedimiento con las implementaciones de diferentes procesos (dos, tres y cuatro). Se hizo lo anterior 10 veces y se anotaron los resultados para sacar un promedio por cada n.

Cuando hay grandes estructuras de datos conjuntas, obtener metadatos de ellas es más  
accesible desde una posición inicial a una posición final, basándose en esta lógica, se  
analiza la multiplicación de matrices en el contexto de estructuras de datos, donde una  
matriz es un arreglo mxm en este caso, y se visualiza como una serie de filas y columnas.  
Entre más cerca esté el dato de su puntero inicial, más accesible es, mientras que, entre  
más lejano esté, más costará su acceso.  
En el caso de un 2 arreglos (suponga un array X[10] y Y [4][3]), el acceso a X[8] es más  
costoso que el acceso a Y[4][1], porque, en el primer caso, solo se transitó por 8  
apuntadores a partir del array X, mientras qué, para el caso Y, se debió transitar por todo  
Y[1-3][1-3] para llegar a dicho dato. Es de resaltar que, aunque no se haya leído cada uno  
de los datos anteriores, se debió recorrer dicho apuntador entero para llegar a él, y esto  
costó (muy levemente) más tiempo que el primer proceso.  
La multiplicación de matrices requiere obligatoriamente leer las columnas de una de las  
matrices como parámetro para la operación punto, por lo que, si se cambia la manera de  
leer los datos de columnas a filas, se cambiará el consumo de caché, y por ende el  
rendimiento.  
Con todo lo anterior en cuenta, se plantea la multiplicación de matrices cuadradas por los  
métodos anteriores (hilos y fork) incluyendo, transposición de una de las matrices para  
eliminar las columnas de la fórmula.  
El código trabajado es muy cercano a los presentados anteriormente, por lo que se centrará  
en la transposición de la matriz en vez de los contenidos anteriores, pero se añadirán para  
mantener la estructura del informe.  
Inicialmente, se delimitan las librerías a usar para el desarrollo de la práctica, donde se  
denota el uso de la librería <pthread> para el uso de hilos, aunque se prueba tanto con hilos  
como con Fork, para efectos prácticos, solo se demostrará el funcionamiento hilos en el  
documento  
.  
Se define el rango de valores aleatorios a llenar en la matriz (entre 0 y 5, denotado en  
MAX\_RANDOM) y 3 variables referentes a la matriz original, una matriz transpuesta y el  
resultado de la operación.

|  |  |
| --- | --- |
| Implementación Secuencial | |
| N | Tiempo de usuario (S) |
| 100 | 0,00254 |
| 200 | 0.01983 |
| 400 | 0.17439 |
| 800 | 1.1620159 |
| 1600 | 21.663026 |
| 3200 | 194.518195 |
|  |  |

# CONCLUSIONES

* La creación de procesos Fork() es costosa y difícil de controlar, pero es útil en el  contexto de crear clones o procesos muy similares, como consola, servidores y  demás estructuras que requieran duplicarse pero su estructura sea idéntica.
* El speedup de los procesos Fork() frente por lo que se intuye que los Fork() son más  costosos y no deberían ser usados en este contexto ya que su desempeño no es  natural y deben ser forzados a usar memoria compartida cuando su propósito es  literalmente inverso.
* Se nota una leve mejora en el uso de procesos Fork() después de multiplicar  matrices cuadradas 400x400, antes de eso, es incluso contraproducente paralelizar  por dicha técnica.
* A diferencia de los hilos, que consumen altos recursos de cómputo (CPU), los procesos Fork() consumen altos niveles de memoria, gracias a que hacen una copia  literal de un conjunto de instrucciones.