**Caso de estudio multiplicación de Matrices: forks vs implementación secuencial**

#### Oscar M. Giraldo Herrera

Universidad Tecnológica De Pereira

Facultad De Ingenierías

Andrés Ramiro Barrios

#### Octubre 2021

# TABLA DE CONTENIDO

Contenido

[TABLA DE CONTENIDO 2](#_Toc89601507)

[RESUMEN 3](#_Toc89601508)

[ABSTRACK 4](#_Toc89601509)

[*KEY WORDS: high performance computing, development, C programming, Python, sequential computing, parallel computing, Forks, Processes, computer system, CPU, programming language, parallelism, concurrency, hardware, software, speed up rate.* 4](#_Toc89601510)

[INTRODUCCIÓN 5](#_Toc89601511)

[MARCO CONCEPTUAL 8](#_Toc89601512)

[MATRICES 8](#_Toc89601513)

[MULTIPLICACION DE MATRICES CUADRADAS 9](#_Toc89601514)

[¿COMO SE REALIZA LA MULTIPLICACION DE DOS MATRICES? 9](#_Toc89601515)

[PROCESOS 10](#_Toc89601516)

[Procesos en la Informática. 10](#_Toc89601517)

[PROCESOS FORK() 10](#_Toc89601518)

[¿QUE ES SOFTWARE? 10](#_Toc89601519)

[¿QUE ES LA SUPERCOMPUTACION? 11](#_Toc89601520)

[¿QUE ES LA COMPUTACION SECUENCIAL? 12](#_Toc89601521)

[COMPUTACION SECUENCIAL 12](#_Toc89601522)

[COMPUTACION PARALELA 14](#_Toc89601523)

[¿QUE ES PARALELISMO? 14](#_Toc89601524)

[¿QUÉ ES LA COMPUTACION PARALELA? 15](#_Toc89601525)

[MEMORIA COMPARTIDA 17](#_Toc89601526)

[¿QUÉ ES HIGH PERFORMANCE COMPUTING? 18](#_Toc89601527)

[¿QUÉ SON LOS HILOS Y LA CONCURRENCIA DE MULTIPLES HILOS? 18](#_Toc89601528)

[MARCO CONTEXTUAL 19](#_Toc89601529)

[DESARROLLO 20](#_Toc89601530)

[PRUEBAS 21](#_Toc89601531)

# 

# 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 mediante técnicas internas de paralelismo. Estas técnicas las cuales se aplican en diferentes contextos, donde su implementación no necesariamente optimizara el proceso tradicional. Lo que hace necesario probar diferentes técnicas de paralelismo para delimitar los campos de aplicación.

En el caso de estudio actual en el cual se mostrará la implementación y ejecución de un programa que realice la multiplicación de matrices de tamaño N x N, en forma secuencial, un algoritmo el cual se comprará usando paralelismo mediante procesos *Forks().*

PALABRAS CLAVE: high performance computing, desarrollo, programación en C, Python, computación secuencial, computación paralela, Forks, Procesos, sistema computacional, CPU, lenguaje de programación, paralelismo, concurrencia, hardware, software, speed up rate.

# ABSTRACK

*The purpose of this document is to show the development process with high-performance computing that has been shown to exponentially improve execution times through internal parallelism techniques. These techniques which are applied in different contexts, where their implementation will not necessarily optimize the traditional process. What makes it necessary to try different parallelism techniques to delimit the fields of application.*

*In the current case study in which the implementation and execution of a program that performs the multiplication of matrices of size N x N will be shown, sequentially, an algorithm which will be purchased using parallelism through Forks () processes.*

# *KEY WORDS: high performance computing, development, C programming, Python, sequential computing, parallel computing, Forks, Processes, computer system, CPU, programming language, parallelism, concurrency, hardware, software, speed up rate.*

# INTRODUCCIÓN

James Nicholas Graymás conocido como **Jim Gray**, fue un científico de la computación estadounidense Recibió el Premio Turing en 1998 "por contribuciones originales a la investigación en bases de datos y procesamiento de transacciones, y su liderazgo técnico en la implementación de sistemas".

Gray estudió en la Universidad de California, Berkeley, donde recibió su licenciatura en ingeniería matemática en 1966 y su doctorado en ciencias de la computación en 1969. Fue el primer doctorando del departamento de ciencias de la computación en Berkeley. Ref. <https://es.wikipedia.org/wiki/Jim_Gray>

Uno de los aportes a la ciencia de la computación que dejo Gray antes de su desaparición en el año 2007 fueron “los cuatro paradigmas de la ciencia” los

Que cuales son una recapitulación sobre la evolución que ha tenido la ciencia, desde sus comienzos de forma empírica, pasando por la rama de la teoría y llegando hasta la rama computacional, esta última permitiéndonos avanzar hacia lo que Gray llamo la exploración de datos o e- science.

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.

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.

Pero ¿qué tan rápido podría un sistema computacional resolver nuestra multiplicación de matrices? ¿Acaso estamos utilizando toda la capacidad del sistema para resolverlo o solo una parte? ¿Es posible mejorar el tiempo que tarda nuestro sistema en resolver la multiplicación?

Este documento mostrara los resultados obtenidos al desarrollar la multiplicación de matrices para cantidades de datos diferentes, aplicada primero de forma secuencial y enfrentándola contra una versión paralela de la misma multiplicación en la que buscaremos disminuir los tiempos de respuesta del sistema y aplicar los conceptos de speed up rate revisados en la ley de Amdahl, la cual propone que dependiendo del tamaño o porción de código que se logre paralelizar se obtendrá cierta aceleración en tiempos de ejecución y que llegando a un 95% de código paralelizado se podría alcanzar una aceleración de 20x de speed up rate.

En la optimización de procesos, lo que se refiere a la computación de alto desempeño (Hihg Performance Computing) HPC por sus siglas en inglés, se basa en técnicas para mejorar el rendimiento de un código es decir su compilación, mediante técnicas y prácticas de programación. Algunas de las prácticas de programación se refieren al uso del paralelismo y la concurrencia de procesos.

Dentro del área de la computación paralela se manejan métodos como: hilos, los cuales mediante la computación ejecutan varios procesos a la vez en sus procesadores, estos paradigmas también se pueden aplicar a las arquitecturas cliente-servidor. Existe otro método primitivo de los sistemas UNIX llamado Fork(), lo cual se denomina como proceso el cual es capaz de realizar una copia literal de un bloque de datos para que se ejecuten paralelamente; este proceso pose grandes ventajas y también desventajas.

# 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 tanto de la multiplicación secuencial como de la ejecución paralela usando procesos Forks().

## MATRICES

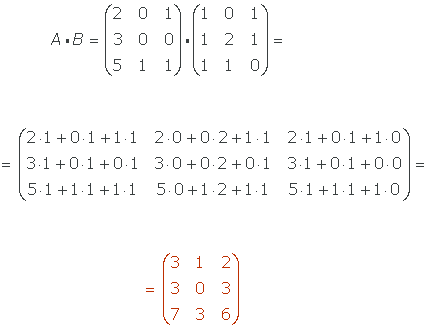
Se define una matriz como una colección de números organizados por una cantidad finita de filas y columnas [3], puede ser una representación de un sistema lineal de ecuaciones, datos ordenados o demás estructuras abstractas. La representación de una matriz puede coincidir con la representación de un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

## PROCESOS

Un proceso es una secuencia de acciones que se llevan a cabo para lograr una tarea o fin determinado, Se trata de un concepto aplicable a muchos ámbitos como por ejemplo al sector industrial, a la química, a la informática, biología entre otros. Y para el desarrollo de este trabajo abarca el ámbito de la informática. Un proceso se trata entonces de una serie de operaciones realizadas en un orden especifico con un objetivo.

### Procesos en la Informática.

Un proceso es la instancia de un programa informático que está siendo ejecutado por uno o varios hilos. Contiene el código del programa y su actividad. Un proceso es masivo dentro de los equipos de cómputo y son administrados por el sistema operativo. En la programación; se considera un proceso cualquier bloque de código a ejecutar.

## PROCESOS FORK()

La funcion *Fork(),* es el método primario de la creación de procesos en sistemas parentados con arquitectura UNIX, permite crear una copia del proceso original llamado hijo que ejecutara las instrucciones inmediatas a su creación. Los fork se ejecutan de manera independiente a sus padres u no comparten memoria, por lo que pueden ser fáciles de controlar.

## ¿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

## ¿QUE ES LA COMPUTACION SECUENCIAL?

### COMPUTACION SECUENCIAL

En las ciencias de la computación, el acceso secuencial significa que un grupo de elementos es accedido en un predeterminado orden secuencial. Indica que el procesador que debe ejecutar de forma consecutiva una lista de acciones (Estas pueden ser a su vez otras estructuras de control). A veces es la única forma de acceder a los datos. Para construir una secuencia de acciones basta con escribir cada acción en una línea diferente.

Por último, es necesario señalar un aspecto importante de la composición secuencial y es  
que no es conmutativa.

En las estructuras de datos, se dice que una estructura tiene acceso secuencial si solo podemos visitar los valores contenidos en un determinado orden. El ejemplo trivial, es la lista enlazada.  
  
A pesar de que la computación secuencial ha alcanzado avances extraordinarios en nuestra historia reciente, no ha podido demostrar su aplicación exitosa en problemas complejos donde se requiera máquinas que funcionen de manera autónoma, robusta y eficiente, dentro de ambientes dinámicos y  
  
amenazantes. Existen otros paradigmas alternativos de cómputo, agrupados en lo que se ha denominado bioinformática y computación celular, los cuales tienen una motivación fundamentalmente biológica, y ciertamente han mostrado resultados prometedores. En este artículo se abordan sus principales características y se mencionan algunos avances.

Forma

Descripción generada automáticamente con confianza media

Ilustración Esquema Secuencial

Gráfico

Descripción generada automáticamente

## COMPUTACION PARALELA

En el sentido más simple, la computación paralela es el uso simultaneo. Operando sobre el principio de problemas grandes, a menudo se pueden dividir en unos mas puequeños que luego son resueltos simultáneamente (en paralelo).

Un problema se divide en partes discretas que se pueden resolver simultáneamente.

* Cada parte se descompone en una serie de instrucciones
* Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores.
* Se emplea un mecanismo global de control/coordinación

### ¿QUE ES PARALELISMO?

El paralelismo es una forma de computación en la cual varios cálculos pueden realizarse simultáneamente,1 basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo. Hay varios tipos diferentes de paralelismo: nivel de bit, nivel de instrucción, de datos y de tarea. El paralelismo ha sido empleado durante muchos años, sobre todo para la Computación de alto rendimiento.

Diagrama

Descripción generada automáticamente

El paralelismo tiene res formas diferentes de aplicarse:

1. **Paralelismo independiente:** paralelismo-And se denomina independiente, debido a que su principal característica es que, una vez establecidas las dependencias entre variables, únicamente se permite ejecutar en un paralelo aquellas metas que sean independientes, de forma en que sus resultados no sean inconsistentes entre sí.
2. **Paralelismo Regular:** Se trata de aplicar paralelismo operaciones con datos donde sus procesos son mutuamente independientes.

Poniendo como ejemplo la condición para que dos rectas sean paralelas si sus vectores son paralelos, es decir si estos son linealmente independientes.

1. **Paralelismo no Estructurado:** Es cuando las computaciones concurrentes difieren, significando que el acceso a los datos no es predecible y necesitan ser coordinados a través de sincronización explícita. Esta forma de paralelismo es la más común en programas escritos usando hilos.

### ¿QUÉ ES LA COMPUTACION PARALELA?

El procesamiento paralelo es un tipo de procesamiento de la información, que permite que se ejecuten varios procesos concurrentemente [5, 10, 17, 30, 35]. El procesamiento paralelo puede ser de diferentes tipos: i) Un tipo es ejecutar procesos independientes simultáneamente, los cuales son controlados por el sistema operativo (usando tiempo compartido, multiprogramación y multiprocesamiento). ii) Otro tipo es descomponer los programas en tareas (controladas por el sistema operativo, los compiladores, los lenguajes de programación, etc.), algunas de las cuales pueden ser ejecutadas en paralelo. iii) Finalmente, el último tipo se basa en usar técnicas de encauzamiento para introducir paralelismo a nivel de instrucciones, lo que implica dividirlas en pasos sucesivos que pueden ser ejecutados en paralelo, cada uno procesando datos diferentes. Es difícil determinar qué resulta más natural, sí el procesamiento paralelo o el secuencial. Los enfoques paralelos resultan una necesidad, dada la constante afirmación de que la velocidad máxima en procesamiento secuencial se alcanzará prontamente, excepto que aparezcan avances en otras áreas que definan nuevos mecanismos de procesamiento, los cuales pueden venir de nuevos descubrimientos en áreas tales como computación cuántica

Gráfico, Gráfico de líneas, Gráfico de cajas y bigotes

Descripción generada automáticamente

## MEMORIA COMPARTIDA

En el contexto de la programación, la memoria se considera como los espacios en los que se guardaran los contenidos de un proceso, en este contexto, funciones, variables e instruciones.

En la ejecución de un programa de C, la memoria se maneja de manera arbitraria, donde el sistema operativo elige direcciones de memoria para asignarlas a variables y demás; este funcionamiento es correcto en la programación secuencial y en entornos como paralelismo por hilos, pero su funcionamiento se ve obstruido en el contexto de procesos Fork()

Los procesos Fork(), al ser copias literales de instrucciones de código, no pueden acceder a las mismas variables que su programa padre, por lo que se deben realizar alternativas de memoria compartida para que, en el contexto de multiplicación de matrices, puedan acceder en conjunto a un mismo espacio de memoria.

## ¿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 cuadrillones 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.

## ¿QUÉ SON LOS HILOS Y LA CONCURRENCIA DE MULTIPLES HILOS?

De la misma manera que un Sistema Operativo puede ejecutar varios procesos al mismo tiempo bien sea por concurrencia o paralelismo, dentro de un proceso puede haber varios hilos de ejecución. Por tanto, un hilo puede definirse como cada secuencia de control dentro de un proceso que ejecuta sus instrucciones de forma independiente. En la Figura puede verse cómo sobre el hardware subyacente (una o varias CPU’s) se sitúa el Sistema Operativo. Sobre éste se sitúan los procesos (Pi) que pueden ejecutarse concurrentemente y dentro de estos se ejecutan los hilos (hj) que también se pueden ejecutar de forma concurrente dentro del proceso. Es decir, tenemos concurrencia a dos niveles, una entre procesos y otra entre hilos de un mismo proceso. Si por ejemplo tenemos dos procesadores, se podrían estar ejecutando al mismo tiempo el hilo 1 del proceso 1 y el hilo 2 del proceso 3. Otra posibilidad podría ser el hilo 1 y el hilo 2 del proceso 1.

Los procesos son entidades pesadas. La estructura del proceso está en la parte del núcleo y, cada vez que el proceso quiere acceder a ella, tiene que hacer algún tipo de llamada al sistema, consumiendo tiempo extra de procesador. Por otra parte, los cambios de contexto entre procesos son costosos en cuanto a tiempo de computación se refiere. Por el contrario, la estructura de los hilos reside en el espacio de usuario, con lo que un hilo es una entidad ligera. Los hilos comparten la información del proceso (código, datos, etc). Si un hilo modifica una variable del proceso, el resto de los hilos verán esa modificación cuando accedan a esa variable. Los cambios de contexto entre hilos consumen poco tiempo de procesador, de ahí su éxito.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

# 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, lo que conlleva a realizar una investigación técnica sobre el funcionamiento de los procesos Fork(), el cual es el fundamento de la presente práctica.

Se analiza e interioriza su funcionamiento y se comprende su diferencia teórica principal, y es que a comparación de los hilos, los cuales ejecutan una función o un bloque de instrucciones particular, los procesos, generan un hilo, el cual es una copia literal de los bloques de código posteriores; lo cual implica un problema en cuanto al control, *(este fue un descubrimiento post práctica, con ayuda de compañeros de clase con los cuales hable un par de veces sobre muchas de las practicas, para avanzar en estas y lograr culminarlas)* de memoria.

Con el concepto dominado, se lleva a cabo el desarrollo del código correspondiente a la práctica.

Inicialmente, se delimitan las librerías a usar para el desarrollo de la práctica, donde se denota el uso de la librería <unistd> para el uso de Fork en un entorno tipo Unix soportado por gcc. También se denota el uso de string.h para la copia de datos vista posteriormente.

Se define el rango de valores aleatorios a llenar en la matriz (entre 0 y 5, denotado en MAX\_RANDOM) y 2 variables relacionadas a guardar las matrices a resolver secuencialmente.

Texto

Descripción generada automáticamente

Se ejecuta la función principal del código la cual tiene variantes con respecto a la práctica  de hilos, donde la separación de espacios de memoria se da de una manera diferente,  gracias a que, dicho anteriormente, los procesos Fork() requieren de memoria compartida, y  su creación y en general gestión es radicalmente diferente al manejo de memoria  tradicional.

7

Se crean los bloques de memoria compartida con la función create\_shared\_memory()  creada manualmente. Se observa que únicamente se llena la matriz secuencial y luego se  copia en la matriz compartida, finalmente, se ejecuta todo lo relacionado al cálculo mediante  la función generateTimes() y finalmente se libera memoria según el tipo de memoria usada.

Imagen que contiene Texto

Descripción generada automáticamente

En la creación de memoria compartida, se requiere saber cuanta memoria de este tipo se va  a utilizar y se deben otorgar permisos de lectura y escritura; esta memoria será de acceso  exclusivo para los procesos Fork() y se reservará para que, durante su vida útil, no pueda  ser usada ni accedida por otras partes del programa.

Texto

Descripción generada automáticamente

Después de generar la memoria, se crea la función generateTimes(), la cual se encarga de  ejecutar los diferentes tipos de ejecuciones del código, donde se hará la multiplicación de  matrices de tamaño m 10 veces para obtener los diferentes resultados.

Se separa la ejecución secuencial de la ejecución paralela, donde la ejecución secuencial se lleva a cabo con la función runSecuential(), pero al no tener cambios con respecto a la  propuesta en la entrega pasada, se omitirá su desarrollo.

En el caso de Fork(), se usa la función measureProcessesOperation(), la cual recibe como  argumento qué función va a ejecutar y con cuántos procesos Fork(),

Texto

Descripción generada automáticamente

La función measureProcessesOperation() toma los tiempos de cada ejecución; lo relevante  de dicha función es la manera en la que se deja en espera el programa mientras todos los  procesos Fork() se resuelven. La función que crea los procesos Fork() es  createProcesses().

Texto

Descripción generada automáticamente

En la función createProcesses(), se definen cosas tales como el tamaño de los trozos de  matrices a compartir, se crean estructuras de datos para los procesos Fork() y se crea cada  uno de ellos. Desde la ejecución de la instrucción Fork() se crea un hijo copia, y se genera  un control para que ejecute únicamente lo necesario y se destruya finalmente. Inicialmente,  se verifica si su identificador es 0, si es menor, entonces sucedió un problema a nivel de  sistema operativo y la operación se cancela, si su valor es 0, significa que el Fork() es hijo  del proceso original y se le asignan los datos suficientes para que pueda realizar su proceso  de multiplicación de matrices, finalmente, se aniquila el procesos con exit(0) para asegurar  que el proceso no ejecute instrucciones adicionales o consuma memoria de manera

Texto

Descripción generada automáticamente

Durante el Fork() de createProcesses() se ejecuta la función operation, que viene heredada  de antes, la cual define si la paralelización de la multiplicación de matrices será por filas o  columnas, en este caso, solo se estudiará la multiplicación dividida por columnas.

En la función multiplyMatrixByColumn(), se recibe un bloque de la matriz a multiplicar y, en  vez de delimitar el fin del ciclo for por el tamaño de la matriz, se delimita por los bloques  dados en la función anterior.

# 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.

|  |  |
| --- | --- |
| 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.