Comparación de rendimiento de multiplicación de matrices en diferentes lenguajes de programación.

*Brayan Steven Bernal Martínez-.* [*brayans.bernal@usa.edu.co*](mailto:brayans.bernal@usa.edu.co)

*Escuela de Ingeniería y Ciencias Exactas*

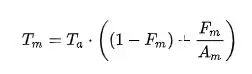
*Resumen:*

*En este informe se busca evaluar el rendimiento general de las computadoras actuales, esto aplicándolo a dos máquinas para determinar el mejor algoritmo y el equipo que se comporta con mayor eficiencia empleando 4 benchmark. Se realiza una batería de experimentos con los benchmark 1, 2, 3, y 4, que contienen un algoritmo clásico de multiplicación de matrices (filas por columnas) con diferentes tamaños a fin de evaluar los tiempos de ejecución. Se concluye que los nodos tienden a tener un mejor rendimiento en el algoritmo empleado en C en el cual se usa Openmp (Hilos), con lo que se realiza una mejor distribución de las cargas.*

*Introducción:*

*Para comenzar, durante esta práctica se realizará una comparativa de rendimiento entre diferentes computadoras, las cuales denominamos nodos A y B, con especificaciones distintas. Se realizarán una serie de experimentos utilizando 4 algoritmos (benchmark), el primero contiene el algoritmo para la multiplicación de matrices clásica (filas por columnas) en C utilizando punteros, el segundo contiene la multiplicación de matrices clásica en C++, el tercero que contiene la multiplicación clásica de matrices filas por columnas en C++ referencia y por último otro algoritmo en C usando distribución con hilos con OpenMP. Posteriormente se analizarán los resultados a fin de comprender a fondo la jerarquía de memoria en la configuración de cada nodo.*

*También, se debe tener en cuenta la ley de Amdahl, la cual hace referencia a la mejora obtenida en el rendimiento del sistema analizado, cuando se realiza una alteración a sus componentes estando limitada directamente a la fracción de tiempo que está usando ese componente, expresado con la siguiente fórmula:*

**

*Figura 1: Ley de Amdahl mejorada*

*(FM): Se considera la fracción de tiempo durante la cual podemos utilizar la mejora que hemos introducido. En donde podemos encontrar:*

*Fm: En donde representa la fracción de tiempo que el sistema utiliza del subsistema mejorado.*

*Am: Es el factor de mejora que se ha introducido en el sistema.*

*Ta: Es el tiempo de ejecución antiguo.*

*Tm: Es el tiempo de ejecución mejorado*

1. **Marco teórico**

**Lenguaje de programación C:** C es un lenguaje de programación de propósito general que ha sido estrechamente asociado con el sistema UNIX en donde fue desarrollado puesto que tanto el sistema como los programas que corren en él están escritos en lenguaje C. Sin embargo, este lenguaje no está ligado a ningún sistema operativo ni a ninguna máquina, y aunque se le llama “lenguaje de programación de sistemas” debido a su utilidad para escribir compiladores y sistemas operativos, se utiliza con igual eficacia para escribir importantes programas en diversas disciplinas. Muchas de las ideas importantes de C provienen del lenguaje BCPL, desarrollado por Martin Richards. La influencia de BCPL sobre C se continuó indirectamente a través del lenguaje B, el cual fue escrito por Ken Thompson en 1970 para el primer sistema UNIX de la DEC PDP-7.

C proporciona las construcciones fundamentales de control de flujo que se requieren en programas bien estructurados: agrupación de proposiciones, toma de decisiones (if-else), selección de un caso entre un conjunto de ellos (switch), iteración con la condición de paro en la parte superior (while, for) o en la parte inferior (do), y terminación prematura de ciclos (break). [1]

**Benchmark:** Un método simple de medir el desempeño es por medio de un programa de referencia. A menos que un programa de este tipo se construya cuidadosamente, es poco probable que sea típico de los muchos miles de programas que se ejecutan en una instalación. Se presenta un punto de referencia de ejemplo para medir la potencia del procesador de las computadoras científicas: se compara con otros métodos para evaluar la potencia de la computadora.[2]

Algoritmo\_1: multiplicación de matrices clásica filas por columnas C++ Punteros.

Algoritmo\_2: multiplicación de matrices clásica filas por columnas C con OpenMp.

Algoritmo\_3: multiplicación de matrices clásica filas por columnas C con matrices llenas con números fijos y punteros.

Algoritmo\_4: multiplicación de matrices clásica filas por columnas C con matrices llenas con números ramdom y punteros.

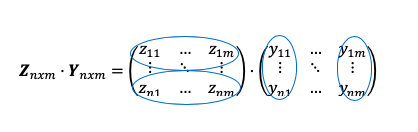
**Punteros:** Un puntero es una variable que almacena la dirección de memoria de un objeto. Los punteros se usan ampliamente en C y C++ para tres propósitos principales:

* Para asignar nuevos objetos en el montón,
* Para pasar funciones a otras funciones
* Para recorrer en iteración los elementos de matrices u otras estructuras de datos.[3]

**Linux:** El enorme mercado de consumidores de IBM PC y compatibles los ha vuelto asequibles. Ahora, con un sistema operativo gratuito llamado Linux, estas máquinas económicas se pueden convertir en potentes estaciones de trabajo para la enseñanza, la investigación y el desarrollo de software. Para los profesionales que utilizan estaciones de trabajo basadas en Unix en el trabajo, Linux permite entornos de trabajo prácticamente idénticos en sus máquinas personales domésticas. Para las instituciones educativas conscientes de los costos, especialmente en los países en desarrollo, Linux puede crear entornos informáticos de clase mundial a partir de clones de PC económicos y de fácil mantenimiento. Y para los estudiantes universitarios, especialmente en ciencia e ingeniería, Linux ofrece un camino bastante gratuito hacia Unix y X Windows.[4]

**Multiplicación de matrices:**

La multiplicación de matrices consiste en combinar linealmente dos o más matrices mediante la adición de sus elementos dependiendo de su situación dentro de la matriz origen respetando el orden de los factores. [5]



*imagen1 Estructura de la multiplicación de matrices*

**Thread**: Un thread es básicamente una sección de código independiente que el procesador puede ejecutar de forma concurrente junto a otros threads o hilos de ejecución. En Linux y otros UNIX un thread se define como un contexto de ejecución mientras que Windows define un thread como algo separado de un proceso.[6]

**Cores:** Los núcleos o Cores se dedican a leer instrucciones y a ejecutar acciones específicas. Si llevamos a cabo cualquier tipo de acción en nuestro teléfono inteligente, tableta, televisor y ordenador –entre otros dispositivos-, todo esto pasa por el procesador. Los núcleos del procesador, como decíamos, son capaces de hacer ciclos de procesamiento de forma independiente, en cuatro partes: lectura, decodificación, ejecución y escritura.[7]

**CPU (Unidad Central de procesos)**: La CPU puede ser un único chip o una serie de chip que realizan cálculos aritméticos y lógicos y que temporizan controlan las operaciones de los demás elementos del sistema. [8]

**Memoria de acceso aleatorio o RAM**: Esta memoria está basada en semiconductores que puede ser leída y escrita por el microprocesador u otros dispositivos de hardware. Es un acrónimo del inglés (Random Access Memory). El acceso a las posiciones de almacenamiento se puede realizar en cualquier orden.[8]

**Lenguaje de programación C++:** Es un lenguaje de programación que enriquece la escritura estática tipo Python by C, la capacidad de llamar directamente a las funciones C y muchas otras características. Esto permite alcanzar un rendimiento de nivel C sin dejar de utilizar una sintaxis similar a Python.[2]

**Bibliotecas:**

**<Vector>:** Uno de los grandes aciertos cuando se creó C++ fue introducir esta plantilla, ya que nos permite un manejo más dinámico de arreglos o arrays, afortunadamente este »tipo de dato» no se limita solo a arreglos de tipos primitivos sino que nos permite crear y manipular arreglos de tipos predefinidos o también definidos por el usuario.[9]

**<random>:** Un generador de números aleatorios es un objeto que crea una secuencia de valores pseudoaleatorios. Un generador que crea valores distribuidos uniformemente en un rango especificado es un generador de números aleatorios uniformes (URNG). Una plantilla de clase diseñada para funcionar como UN URNG se conoce como motor si esa clase tiene ciertos rasgos comunes, que se deban tratar más adelante en este artículo. Un URNG se puede combinar (de hecho, suele combinarse) con una distribución si el URNG se pasa como un argumento al de la distribución con el fin de generar valores que se distribuyen de la manera especificada por la distribución.[10]

**<time.h>** Este archivo de encabezado contiene definiciones de funciones para obtener y manipular información de fecha y hora.[11]

1. **Metodología**

Se unificaron los tres diferentes Benchmark en una sola carpeta y se realizó un ejecutable que abarcara los 3 Benchmark para facilitar la toma de muestras en cada Nodo, los cuales se encuentran disponibles en el siguiente repositorio: https://github.com/LuffyBB1/Arquitectura-de-Computadores-Rendimiento

Con el ejecutable cada Nodo creó una carpeta que guarda los resultados de cada Benchmark.

Estos resultados se guardan en formato csv para luego subir al repositorio de cada miembro del equipo.

Teniendo las carpetas de los resultados en formato csv subidas al github

Se desarrolló un código en python para generar los promedios, desviaciones, etc. En una hoja de cálculo, el código recibe como variables los links a las carpetas de resultados de los repositorios en github

Y como resultado se obtuvo una hoja de cálculo con todos los datos necesarios para hacer las comparaciones entre Nodos.

El punto de comparación se establece como velocidad de ejecución / tamaño de la matriz y se crean las gráficas respecto al mismo.

1. **Resultados**

# **Especificaciones de los equipos**



**Nodo A – MSI**

* Modelo: MSI GF63 Thin 10SCXR
* Procesador Intel ® I5-10500 H CPU
* Número de núcleos de CPU: 6
* Reloj base: 2.5 GHz
* Memoria RAM: 16 GB



**Nodo B – DELL**

* Modelo: ASUS Notebook SKU
* Procesador AMD Ryzen 5 3450U
* Número de núcleos de CPU: 4
* Reloj base: 2.1 GHz
* Memoria RAM: 8 GB

* Ver Anexos.

# CONCLUSIONES

* Una gran diferencia entre los lenguajes de programación C y C++, C es un lenguaje diminuto, es decir, no está basado en ningún otro lenguaje o interfaces, tiene un nivel de programación muy bajo, lo cual permite que la codificación sea mucho más cercana al lenguaje de máquina, esto se traduce en menor tiempo de ejecución en tareas y eficiencia en la administración de recursos.
* Se identifica una clara diferencia en los tiempos de ejecución cerca de un 25% entre los diferentes entre el mejor alhoritmo y el peor, esto para valores de matrices relativamente pequeños, ya que la diferencia para tamaños de matrices NxN superiores a 2000 la diferencia es mayor a 400%.
* El mejor lenguaje fue C con el método de Openmp, ya que este permite realizar una mejor distribución de las tareas a través de los hilos que se indiquen.
* Se observó una diferencia considerable entre los algoritmos 02 y 03, que, pese a que se usaron los mismos métodos y tamaño de variables, la forma de llenar las matrices es distinta, consumiendo menos con respecto al llenado manual.

# INTEGRANTES

Brayan Steven Bernal Martínez nació en Soacha (Cundinamarca) el 16 de octubre de 1997. Estudiante de ingeniería en ciencias de la computación e inteligencia artificial de 8° semestre con amplia experiencia en herramientas de Backups, virtualización y almacenamientos 3PAR y Primera. Actualmente se desempeña como junior en la compañía Storage Availability Solutions.

# REFERENCIAS

1. BRIAN W, K DENNIS ,M. RITCHIE. EL LENGUAJE DE PROGRAMACIÓN C.Mexico, Pearson Educacion, 2002.edicion 2
2. H. J. Curnow, B. A. Wichmann, A synthetic benchmark, The Computer Journal, Volume 19, Issue 1, 1976, Pages 43–49, <https://doi.org/10.1093/comjnl/19.1.43>
3. "Punteros (C++)". Developer tools, technical documentation and coding examples | Microsoft Docs. <https://docs.microsoft.com/es-es/cpp/cpp/pointers-cpp?view=msvc-170> (accedido el 12 de marzo de 2022).
4. S. N. Bokhari, "The Linux operating system," in Computer, vol. 28, no. 8, pp. 74-79, Aug. 1995, doi: 10.1109/2.402081.
5. "Multiplicación de matrices - Economipedia". Economipedia. <https://economipedia.com/definiciones/multiplicacion-de-matrices.html> (accedido el 8 de marzo de 2022).
6. O. Campos. "Introducción al multiprocesamiento en C++". Genbeta - Software, descargas, aplicaciones web y móvil, desarrollo. <https://www.genbeta.com/desarrollo/introduccion-al-multiprocesamiento-en-c> (accedido el 9 de marzo de 2022).
7. "¿Qué son los núcleos de un procesador?" ADSLZone. <https://www.adslzone.net/reportajes/nucleos-cpu-procesador> (accedido el 9 de marzo de 2022).
8. O. Campos. "Introducción al multiprocesamiento en C++". Genbeta - Software, descargas, aplicaciones web y móvil, desarrollo. <https://www.genbeta.com/desarrollo/introduccion-al-multiprocesamiento-en-c> (accedido el 9 de marzo de 2022).
9. Developer tools, technical documentation and coding examples | Microsoft Docs. <https://docs.microsoft.com/es-es/cpp/standard-library/vector?view=msvc-170> (accedido el 12 de marzo de 2022).
10. Developer tools, technical documentation and coding examples | Microsoft Docs. <https://docs.microsoft.com/es-es/cpp/standard-library/random?view=msvc-170> (accedido el 12 de marzo de 2022).
11. " (time.h) - C++ Reference". cplusplus.com - The C++ Resources Network. <https://www.cplusplus.com/reference/ctime/> (accedido el 12 de marzo de 2022).