**RETO 02 OPTIMIZACIÓN EN ALGORITMO MONTE CARLO Y DARTBOARD CON HILOS Y COMPARACIÓN EN C++ Y OpenMP**

#### Oscar M. Giraldo Herrera

Universidad Tecnológica De Pereira

Facultad De Ingenierías

Andrés Ramiro Barrios

#### Octubre 2021

# TABLA DE CONTENIDO

Contenido

[AGRADECIMIENTOS 3](#_Toc89817310)

[TABLA DE CONTENIDO 4](#_Toc89817311)

[RESUMEN 5](#_Toc89817312)

[ABSTRACK 6](#_Toc89817313)

[INTRODUCCIÓN 7](#_Toc89817314)

[MARCO CONCEPTUAL 8](#_Toc89817315)

[METODO MONTE CARLO 8](#_Toc89817316)

[SIMULACION DARDOS DE MONTE CARLO 9](#_Toc89817317)

[PROBLEMA DE LAS AGUJAS DE BUFFON MONTE CARLO 10](#_Toc89817318)

[COMPUTACION SECUENCIAL 11](#_Toc89817319)

[COMPUTACION PARALELA 12](#_Toc89817320)

[MARCO CONTEXTUAL 13](#_Toc89817321)

[DESARROLLO 14](#_Toc89817322)

[MONTE CARLO 14](#_Toc89817323)

[Implementación Monte Carlo 16](#_Toc89817324)

[DARTBOARD 17](#_Toc89817325)

[Implementación DartBoard 18](#_Toc89817326)

[OPTIMIZACION 18](#_Toc89817327)

[RESULTADOS 19](#_Toc89817328)

[MONTE CARLO 19](#_Toc89817329)

[DARTBOARD 21](#_Toc89817330)

[CUAL ES MAS EFICIENTE 22](#_Toc89817331)

[CONCLUSIONES 23](#_Toc89817332)

[REFERENCIAS 24](#_Toc89817333)

# RESUMEN

En la búsqueda para la resolución de problemas computaciones con mínimo uso de los recursos necesarios, posibles como rendimiento en CPU. En el mundo de la computación, los ingenieros en sistemas y computación, informática, de software, etc. Se han visto en la necesidad de hacer uso de la computación de alto desempeño o HPC *(high performance computing)* para optimizar el uso del software aplicando estos conceptos y practicas al desarrollo de software, utilizando algunas herramientas, para sacar el mayor provecho posible del desempeño del hardware que dispone el mismo. Para este caso de estudio se van a desarrollar implementaciones sobre los problemas de mesa de dardos y agujas de buffon, las cuales fueron realizadas de forma secuencial y simuladas por medio del método Monte Carlo, esta implementación se puede mejorar si a la misma se le añade el uso de Hilos Y Forks, (Threads And Forks)

# ABSTRACK

In the search for the resolution of computer problems with minimal use of the necessary resources, possible as CPU performance. In the world of computing, engineers in systems and computing, informatics, software, etc. They have seen the need to make use of high-performance computing or HPC (high performance computing) to optimize the use of software by applying these concepts and practices to software development, using some tools, to get the most out of the performance. of the hardware that it has. For this case study, implementations on the dart table and buffon needles problems will be developed, which were carried out sequentially and simulated by means of the Monte Carlo method, this implementation can be improved if it is added the use of Threads And Forks, (Threads And Forks)

# INTRODUCCIÓN

La solución de problemas matemáticos los cuales cuentan con una gran cantidad de datos, ecuaciones, procesos o algoritmos complejos, ha tenido un gran avance gracias al uso de las computadoras, por sus avanzados cerebros o mejor llamados procesadores.

Existen operaciones conjuntas y grandes cantidades para analizar de datos que generalmente no son problema para las máquinas que existen actualmente, las cuales cuentan con procesadores multi núcleos o de varios núcleos. Cabe aclarar que, aunque estas máquinas son muy potentes en términos de memoria y unidades de procesamiento, incluso para estas máquinas hay límites de procesamiento y calculo y tiempos relativamente largos de ejecución, esto dependiendo de que tan complejo se pueda volver el problema a resolver, de que hardware dispongan y la forma en que este estructurado el código que se ejecute en este maquina o se esté ejecutando. Existen computadores que realizan estos procesos mucho más rápidos, estos equipos cuentan con tecnología cuántica, es decir son computadores cuánticos. Para agilizar estos procesos y la ejecución más ágil, la computación de alto desempeño HPC toma relevancia para el desarrollo de este proyecto, el cual consiste en estudiar el uso de la computación de alto desempeño, por medio del caso de estudio practico del uso del método de Monte Carlo, para la resolución de problemas de mesa de dardos () y agujas de Buffon (), la cual consiste en implementar dos estructuras basadas en hilos y forks respectivamente, a partir del código secuencial de la resolución de los problemas que se mencionaba anteriormente, en este se comprara el rendimiento de las soluciones basadas en hilos y forks con respecto al código secuencial.

# MARCO CONCEPTUAL

## METODO MONTE CARLO

Es un método no determinista, estadístico numérico, el cual es usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud.

El método Monte Carlo fue desarrollado en primera instancia por los científicos John Von Neumann y Stanislaw Ulam en 1944, Aunque durante las primeras etapas de esta investigación, John Von Neumann y Stanislaw Ulam, refinaron esta ruleta y los métodos “de división” de tareas. Aunque para el año de 1948 el desarrollo sistemático de estas ideas tuvo que esperar el trabajo de Harris y Herman Kahn. Durante el mismo año, Enrico Fermi, Nicholas Metropolis y Ulam obtuvieron estimadores para los valores característicos de la ecuación de Schrodinger para la captura de protones a nivel nuclear.

La idea le surgió a Ulam, mientras jugaba a las cartas. Se le ocurrió un método en el que, mediante la generación de números aleatorios, pudieran determinar soluciones a ecuaciones complejas que se aplican al estudio de los neutrones. Era como generar los números con la ayuda de una ruleta, de ahí su nombre.

## SIMULACION DARDOS DE MONTE CARLO

La simulación del tablero de dardos de Monte Carlo se utiliza para llegar a un valor aproximado del número irracional PI. El concepto para trabajar la simulación está basado en un circulo con radio 1 y un cuadrado.

Lo primero es construir el entorno de trabajo. Este sería el siguiente.

* Construiremos un cuadrado de lado 4. Lógicamente su área será 16.
* Construimos un círculo inscrito en el cuadrado, que tiene de centro, el centro del cuadrado y de radio 2. Su área será 4 PI.
* Generaremos puntos al azar dentro del cuadrado. Para entenderlo mejor es como lanzar dardos sobre una diana con los ojos vendados, de tal forma que siempre acertamos dentro de los límites de ese cuadrado.

Aplicamos ahora el Método Monte Carlo:

* Contaremos el total de pun tos generados
* Contaremos el total de puntos que cayeron dentro del círculo.
* Realizaremos el siguiente razonamiento

Imagen que contiene Texto

Descripción generada automáticamente

El desarrollo consiste en lanzar dardos repetidamente al tablero y estos deben aterrizar aleatoriamente dentro de los límites del cuadrado, algunos aterrizan en el cuadrado y otros aterrizan en el tablero de dardos circular. Si estos lanzamientos son verdaderamente aleatorios, entonces el número de dardos que caen en el tablero de dardos, dividido por el numero de dardos lanzados se describe en la imagen anterior. (PI / 4). Si se multiplica ese numero por 4, se obtiene la estimación de PI.

Gráfico de dispersión

Descripción generada automáticamente

Ilustración : Geogebra Dardos

## PROBLEMA DE LAS AGUJAS DE BUFFON MONTE CARLO

La aguja de buffon es un problema planteado por el naturalista y francés Georges Louis Leclare, conde de Buffon en 1773. Este problema consiste en calcular la probabilidad que una aguja caiga en una línea de una red de líneas paralelas de anchura constante. Este método permite obtener un valor aproximado del numero irracional Pi.

El planteamiento del problema determina si L en la longitud de las agujas y c la distancia entre las líneas. Los pasos experimentales para la probabilidad de que la aguja se cruce con una línea son:

1. Tome un trozo de papel blanco y dibuje muchas líneas paralelas con un espaciado a.
2. Tome una aguja con una longitud de l (l≤a) y tírela al azar n veces en el papel con líneas rectas paralelas. Observe el número de veces que la aguja se cruza con la línea recta y regístrela como metro.
3. Calcule la probabilidad de que la aguja cruce la línea recta.  
   En el siglo XVIII, el matemático francés Buffon planteó el "problema del lanzamiento de agujas", que se registró en el trabajo de Buffon publicado en 1777: "Un conjunto de líneas paralelas con un espaciado de a se dibujan en un plano. , Arroje una aguja de longitud l (l≤a) en este plano arbitrariamente, encuentre la probabilidad de que esta aguja cruce cualquiera de las líneas paralelas ".  
   Buffon mismo demostró que la probabilidad es:

Imagen que contiene Texto

Descripción generada automáticamente

La solución muestra que, si la longitud de la aguja no es mayor que la anchura de las  
líneas paralelas puede aproximarse al número π por el método de Monte Carlo. La probabilidad  
en este caso sería una aproximación experimental del caso teórico, vamos, sería una estadística. Y cuantas más agujas se cuenten, más nos aproximaremos al número π.

## 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.Forma

Descripción generada automáticamente con confianza media

Ilustración Esquema Secuencial

## 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 más pequeños 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

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.

# MARCO CONTEXTUAL

Para realizar las pruebas de este caso de estudio, el Método Monte Carlo, las pruebas se implementaron en una instancia EC2 de Amazon Web Services ***AWS,*** la cual tenía las siguientes especificaciones.

* Modelo: Ubuntu Server 20.04 LTS (HVM)
* Procesador: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz, 8 CPU virtuales
* Arquitectura: x86\_64
* BIOS: Vendor: Xen. Version: 4.2. Amazon
* Memoria: 32GiB
* Disco duro: 8GiB E/S por segundo: 100/3000

# DESARROLLO

## MONTE CARLO

Monte carlo es una simulación de números aleatorios para la estimación de PI y se usan valores aleatorios para aproximar un valor determinístico.

El algoritmo consiste en dibujar un cuadro y dentro de ese cuadro dibujar un círculo con un diámetro igual a la medida de los lados del cuadrado. Luego se generan puntos aleatorios por todo el cuadro sin importar si se generan dentro o fuera del círculo.

ya que esto sirve como estimadores del área interna y externa del círculo. El cuadro se extiende desde x= -1 hasta x = 1 e y = -1 hasta y = 1 de modo que el círculo y el cuadrado estén el el origen (fig).

Gráfico

Descripción generada automáticamente

Podemos decir que la probabilidad de que un punto se genera dentro del círculo es la cantidad de puntos generados dentro del círculo dividido la cantidad de puntos totales y esto es directamente proporcional al área del círculo.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

El área del círculo en este caso es:



El área del cuadrado es de 4 unidades por lo tanto la ecuación anterior cambia a:

Texto

Descripción generada automáticamente con confianza media

Entonces si se generan números que cumplan la siguiente condición:



entonces las coordenadas de los puntos que caen dentro del círculo deben cumplir la siguiente condición:

Diagrama

Descripción generada automáticamente

De esta manera se puede contar los puntos que se generan o caen dentro del círculo, finalmente aplicación la (E.1) se puede hallar el valor de PI.

### Implementación Monte Carlo

Para la implementación de la simulación de Monte Carlo se realiza un algoritmo en código C++, el código se encuentra en el siguiente enlace.

<https://github.com/oscarmauriciogiraldo/HPC/blob/Master/RETOS/Reto01-Intro-HPC/Reto01-MonteCarlo.cpp>

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

El código genera número aleatorio cumpliendo la condición de la (E.2), luego de que se genera el punto se aplica la (E.3) para saber si el punto cae efectivamente dentro del círculo y por último, luego de que se generan todos los puntos se hace el aplica la (E.4) para hallar un número cercano a PI, cabe recalcar que cada vez que se compile el código puede dar un número diferente ya que se están generando número aleatorios.

## DARTBOARD

El problema se puede imaginar en términos de jugar a los dardos. Deje que el tablero de dardos consista en un objetivo cuadrado con un objetivo circular dentro de él. Para resolver el problema simplemente arrojaría un montón de dardos al objetivo y registraría el porcentaje que aterriza en el objetivo circular interior.

Podemos extender esta idea para aproximarnos con PI bastante facilidad. Suponga que el objetivo cuadrado tiene una longitud de dos pies y el objetivo circular tiene un radio de un pie.

Gráfico

Descripción generada automáticamente

Para la implementación de la simulación de dartboard se realiza un algoritmo en código C++, el código se encuentra en el siguiente enlace.

### Implementación DartBoard

para realizar la implementación del desarrollo hay que tener en cuenta que tenemos que calcular el valor aproximado de PI con lanzamientos aleatorios en un tablero de dardos. Se considera un impacto si cae dentro del círculo unitario (0,0) y el radio 1.

Enlace o imagen

Como se puede apreciar en la imagen 2 el algoritmo dartboard solo utiliza la mitad del tablero para disparar los dardos en la parte superior por lo tanto solo se generan números aleatorios entre 0 y 1. en la implementación del método ***get\_RandomNum(double,double).***



A partir de la condición de la (E.3), ya tenemos los puntos aleatorios y asignarlos a las coordenadas X y Y, después le sacamos el cuadrado a X y a Y (X^2 y Y^2) al momento de realizar la suma de (X^2 y Y^2) se debe validar los siguiente: Si el resultado es menor 0 igual a 1 entonces tenemos un intento de éxito un hits , al momento de terminar el ciclo se obtiene el valor aproximado de PI multiplicando el número de hits \*4 dividido por los números de dardos lanzados.

## OPTIMIZACION

El tiempo de ejecución es un factor importante cuando se generan y se calculan grandes cantidades de puntos, para esto se modifican los algoritmos y se implementan hilos, el motivo por el cual se eligió hilo es que los hilos permiten una granularidad más fina que hacerlo por procesos, cada hilo se encarga de generar y calcular una cantidad x de puntos, de esta manera se entrega un resultado más rápido cuando se habla de simular grandes cantidades de puntos. Esta implementación por hilos se hizo tanto para el algoritmo de monte carlo como para el de dartboard.

GCC ofrece muchas opciones para optimizar la CPU por compilador, eso ayuda a que el compilador realice las optimizaciones basándose en lo que sabe del programa, de esta manera intenta mejorar el rendimiento y/o el tamaño del código a expensas del tiempo de compilación y posiblemente la capacidad de depurar el programa. Para el caso de los algoritmos de monte carlo y dartboard se realiza una optimización nivel 3 para bucles ya que es una parte importante de cada uno de los algoritmos.

En cuanto a la optimización en memoria se hace el respectivo análisis, pero no se encuentra nada o ninguna manera de optimizar los algoritmos por memoria.

# RESULTADOS

## MONTE CARLO

Se compilan el código serial y paralelo un total de 5 veces con 1.000.000 puntos generados y con diferentes parámetros de optimización:

* Sin ninguna optimización
* Optimización general (-O)
* Optimización para ciclos y también para hilos ( -pthread-fmove-loop-invariants)

Se logra evidenciar una disminución significativa en tiempo de respuesta como se muestra en la gráfica 1.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

En el siguiente enlace se puede observar las gráficas generadas, los tiempos y el resultado de en cada ejecución.

La tabla 1 muestra los resultados obtenidos del código serial y del código en paralelo sin optimización.

Tabla

Descripción generada automáticamente

Número PI en cada tipo de compilación es el algoritmo de monte Carlo, para cada tipo de compilación se obtiene el número más cercano a PI y a cuál algoritmo pertenece el número obtenido y se pinta la casilla de color amarillo, danto como resultado que el algoritmo de Montecarlo se acercó al número PI un total de 4 veces en comparación al algoritmo de dartboard el cual se acercó 2 veces.

Tabla

Descripción generada automáticamente

En el código paralelo se logra reducir significativamente en tiempo de respuesta gracias al el uso de hilos, pero se logra evidenciar en la tabla 1 que los resultados arrojados por el código serial son más cercanos a en comparación con el código paralelo

## DARTBOARD

Se realiza el mismo procedimiento de compilaciones que se utilizó para obtener los puntos en Monte Carlo con los mismos parámetros de optimización. con el fin de realizar estudios en base a los diferentes tipos de compilación y tener una base estadística para sustentar cuál de los 2 puede ser más eficiente para hallar el valor más cercano de PI en recursos, memoria y tiempo de ejecución.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Se puede observar que haciendo una comparación lineal entre el rango mínimo y máximo de los resultados en milisegundos de las implementaciones en paralelo y secuencial contra los tipos de optimización que se mencionaron anteriormente, se obtiene una disminución considerable en tiempo de ejecución.

## CUAL ES MAS EFICIENTE

Luego de realizar las pruebas para dartboard y para Montecarlo surgen las siguientes preguntas, ¿cuál es más eficiente ?, ¿cuál es más exacto hallando el número PI? luego de analizar los datos tomados anteriormente se hace un análisis para cada tipo de ejecución y se logra evidenciar que en tiempo de ejecución total el algoritmo de monte Carlo es mejor en comparación al dartboard, aunque la diferencia no es tan significativa. De igual manera se evidencia que el algoritmo que más se acerca al número PI en cada tipo de compilación es el algoritmo de monte Carlo, para cada tipo de compilación se obtiene el número más cercano a PI y a cuál algoritmo pertenece el número obtenido y se pinta la casilla de color amarillo, danto como resultado que el algoritmo de Montecarlo se acercó al número un total de 4 veces en comparación al algoritmo de dartboard el cual se acercó 2 veces.

# CONCLUSIONES

Está claro que al momento de realizar una comparación entre 2 algoritmos hay que tener las suficientes pruebas para poder decir cuál es más eficiente en los aspectos que necesitamos tanto en CPU, MEMORIA y tiempo de ejecución con una base estadística sólida.

Haciendo énfasis en el estudio de optimización con hilos y poder realizar comparaciones de compilación para obtener la información de tiempos de ejecución es necesario por lo menos realizar mínimo 5 intentos para tener la media y tener unos resultados más significativos.

Gracias a la implementación en paralelo en ambos algoritmos se pudo obtener utilizando de forma correcta los hilos con una granularidad fina para que los tiempos de compilación sean menores de una forma considerable y demostrable a la implementación serial para obtener el valor más cercano a PI.

# REFERENCIAS

1. Fernando Celano. (Agosto 2016). MODELOS Y SIMULACIÓN PARA APROXIMAR EL VALOR DE [pdf web en línea]. Recuperado de https://ucema.edu.ar/publicaciones/download/docume ntos/591.pdf
2. Manas Sharma. Cálculo del valor de Pi usando la técnica de Monte Carlo - PROGRAMA C [página web en línea]. Recuperado de <https://www.bragitoff.com/2018/06/calculating-valuepi-using-monte-carlo-technique-c-program/>
3. Sin nombre. Options That Control Optimization [página web en línea] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options. html