SOPAS

Integrantes:

- Diego A. Polanco L.
- Oscar S. Muñoz R.
- Sebastian Erazo O.
- Ricardo A. Chamorro M.
- Luis M. Rojas C.

Informe de implementación de montecarlo con ICE para el cálculo de π

Método de montecarlo

El Método de Montecarlo es una técnica de simulación numérica que utiliza números aleatorios para resolver problemas complejos o estimar valores difíciles de calcular analíticamente. Su enfoque principal es realizar experimentos aleatorios que simulan el comportamiento de un sistema y luego, a través del promedio de muchos resultados, se obtiene una solución aproximada. Este método es ampliamente utilizado en diversas áreas como la física, finanzas y matemáticas, ya que permite abordar problemas que involucran incertidumbre o muchas variables.

Un ejemplo típico del Método de Montecarlo es la estimación de π. Se inscribe un círculo dentro de un cuadrado y se generan puntos aleatorios dentro de este. La proporción de puntos que caen dentro del círculo respecto al total de puntos generados permite aproximar el valor de π. Cuantos más puntos se utilicen, mayor será la precisión de la estimación.

La fórmula es la siguiente:

$$\pi \approx 4 \frac{Puntos\ dentro\ del\ círculo}{Puntos\ lanzados}$$

Paso a paso con la distribución planeada

En este caso, el master se encarga de realizar el cálculo final, mientras que los workers se enfocan únicamente en generar los puntos y determinar cuántos caen dentro del círculo. El proceso sería así:

Client: Solicita la estimación de π al master, indicando cuántos puntos se deben utilizar para la aproximación.

Master: Recibe la solicitud del cliente y divide la tarea en subtareas que envía a los workers. Cada worker recibe una cantidad de puntos a generar.

Worker: Los workers generan puntos aleatorios dentro de un cuadrado y cuentan cuántos de esos puntos caen dentro de un círculo inscrito en el cuadrado. Luego, envían esta

información (cantidad de puntos dentro del círculo y total de puntos generados) de vuelta al master.

Master: Recolecta los resultados de los workers (cuántos puntos cayeron dentro del círculo) y realiza el cálculo final para estimar π usando la fórmula y finalmente, envía el resultado al client.

Este enfoque distribuye el trabajo de generar puntos entre los workers, mientras que el master se queda con la tarea de calcular el valor de π basado en la información que recibe.

Estrategia de distribución de tareas entre los trabajadores

1. El maestro divide el total de puntos N entre los n trabajadores, asignando a cada uno N/n puntos.

2. Cada trabajador genera aleatoriamente los puntos en el cuadrado de lado 2 y verifica cuántos caen dentro del círculo inscrito.

3. Los trabajadores envían sus resultados (el número de puntos dentro del círculo) al maestro.

4. El maestro recibe los resultados de todos los trabajadores, suma los puntos que cayeron dentro del círculo y utiliza la fórmula del método de Monte Carlo para estimar el valor de π:

$$\pi \approx 4 \frac{Total\ de\ puntos\ dentro\ del\ círculo}{Total\ de\ puntos\ lanzados}$$

Método implementado en ICE

Estructura

- Worker: Consume el servicio ‘Master’.
  Se ejecuta con el comando java -jar worker/build/libs/worker.jar

- Client: Consume el servicio ‘Master’
  Se ejecuta con el comando java -jar Master/build/libs/Master.jar

- Master: Expone el servicio Master’
  Se ejecuta con el comando java -jar client/build/libs/client.jar

Es necesario poner la ip actual del worker y estar en la misma red.

Comunicación con ICE

Adaptadores de Objetos

En ICE, cada componente que expone un servicio utiliza un adaptador de objetos. Por ejemplo, el maestro utiliza el adaptador MasterAdapter para escuchar en el puerto 5000 y coordinar las solicitudes de los clientes y los trabajadores. Los trabajadores utilizan WorkerAdapter para escuchar en sus respectivos puertos y recibir las tareas asignadas por el maestro.

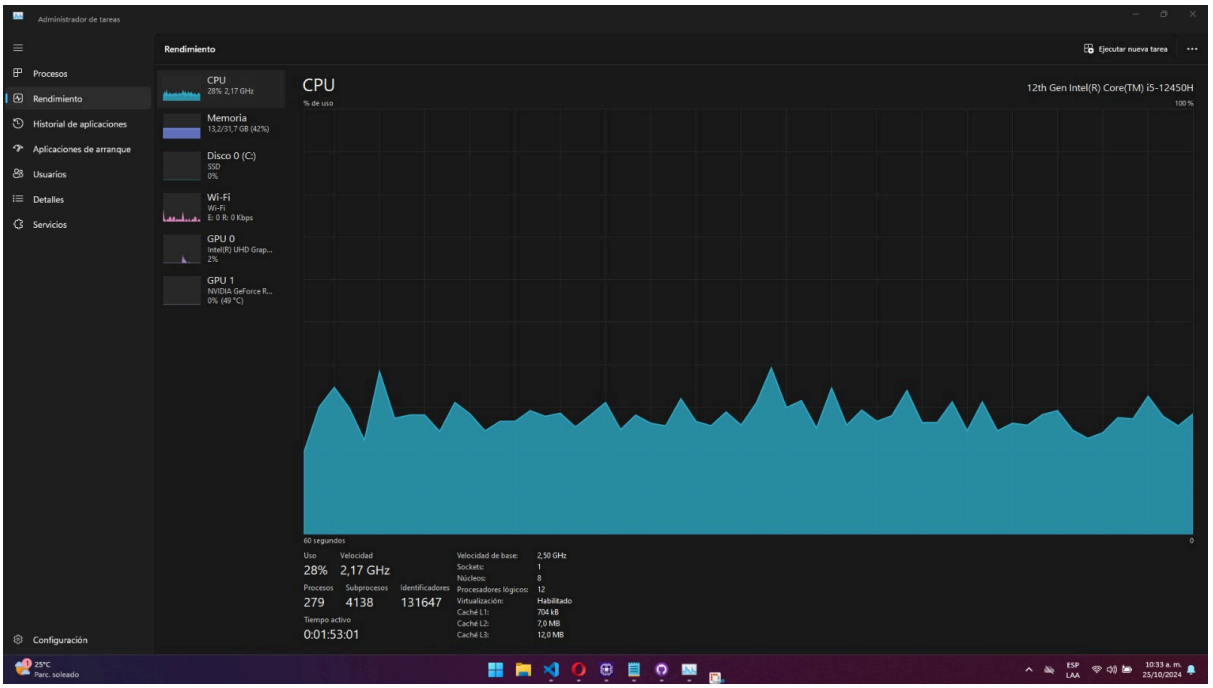
Proxies y Comunicación Remota

La comunicación entre el cliente, el maestro y los trabajadores se realiza a través de proxies generados por ICE. El cliente crea un proxy para conectarse al maestro y enviar peticiones de cálculo. El maestro, a su vez, crea proxies para los trabajadores y distribuye las tareas entre ellos de forma remota.

Resultado de las pruebas

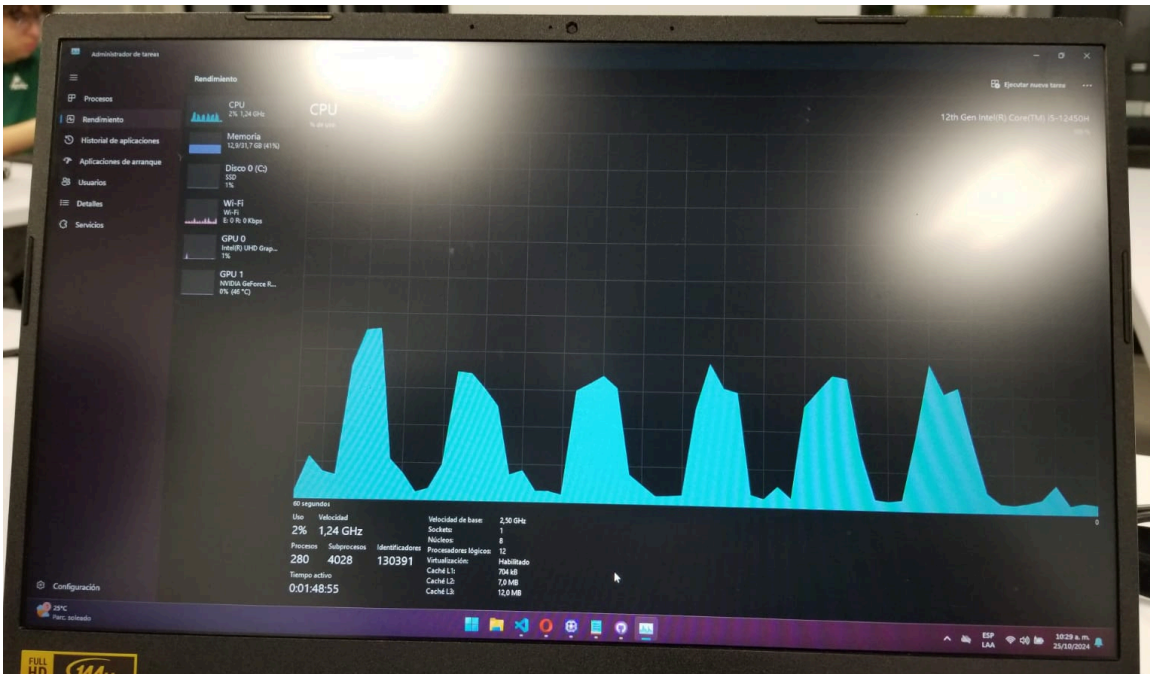
Rendimiento:

Ejecución utilizando un solo Worker: El único worker está siendo sobrecargado con todas las tareas asignadas por el maestro, lo que resulta en un uso elevado y constante de la CPU. Dado que un solo worker está procesando todos los puntos para la estimación de π, su carga de trabajo es muy alta, generando esos picos.

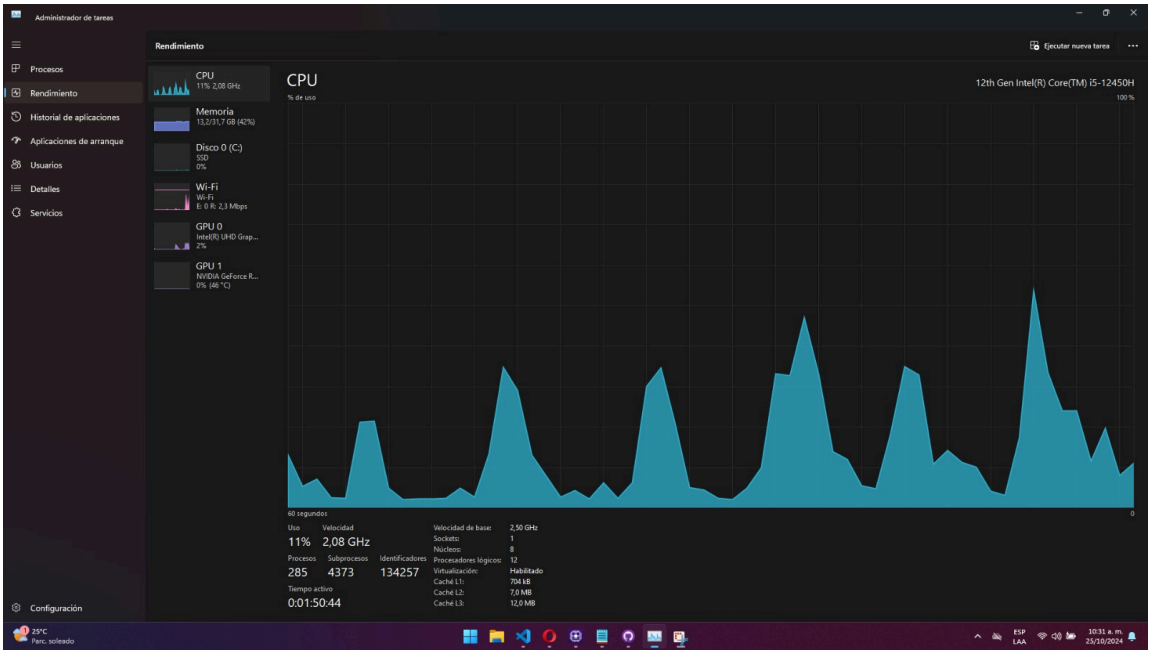


SOPAS

Ejecución utilizando dos Worker: La carga de trabajo ahora está dividida entre dos workers, lo que reduce el estrés sobre cada uno. Como consecuencia, cada worker tiene que procesar menos puntos, lo que disminuye el uso total de la CPU en comparación con un solo worker. Sin embargo, todavía hay picos, lo que la cantidad de puntos asignados sigue siendo significativa para cada worker, o que el sistema no distribuye las tareas de manera completamente uniforme.



Ejecución utilizando tres Worker: La adición de un tercer worker ha distribuido mejor la carga de trabajo, lo que explica la reducción de los picos en algunos tramos. Sin embargo, se siguen viendo picos altos en ciertos momentos, pero cada vez menores a comparación de utilizar menos worker para la distribución de tareas.



Estrategia para un análisis más preciso:

1. **Exclusión de las primeras iteraciones**: Ignoraremos los resultados con pocos puntos, por ejemplo, los que tienen menos de 1,000 puntos generados. Esto permitirá que el análisis refleje con mayor precisión el rendimiento a medida que aumenta el número de puntos.
2. **Análisis progresivo**: Vamos a observar cómo mejora la estimación de π a medida que se incrementa el número de puntos, examinando únicamente los datos de simulaciones con un número de puntos significativo, para evaluar si la estimación de π converge a un valor más preciso.
3. **Comparación de workers**: Evaluaremos cómo los distintos workers (1, 2 y 3) afectan el tiempo de ejecución y la precisión sin dar peso excesivo a las primeras iteraciones.

A partir de esto se generaron CSV con distintos N, de aquí solo se tomaron 5 casos en los que a partir de los registros mayores a 1000 se les calculó la media del valor de Pi obtenido, la desviación estándar y la duración promedio en segundos, en el repositorio se puede observar en la carpeta resultados_pi todos las pruebas (en formato csv) entre los cinco computadores y todos los registros a partir de un N específico dado por el cliente.

Simulation	PiEstimate_mean	PiEstimate_std	Duration Mean (s)
1_worker_diego	3.147421	0.018438	2.179904
1_worker_sebas	3.142064	0.005819	4.693716
2_worker_diego_ricardo	3.144276	0.007265	6.780763
2_workers_sebas_diego	3.141478	0.008159	4.313481
3_workers_sebas_diego_ricardo	3.146600	0.013892	4.844773

Tras filtrar los resultados con menos de 1,000 puntos generados, el análisis muestra un resultado mucho más preciso y relevante del comportamiento de las simulaciones con mayor número de puntos. Aquí están los puntos clave del análisis:

1. Estimación de π más precisa

- **1_worker_diego**: Estimación promedio de π es 3.1474 con una desviación estándar de 0.0184.
- **1_worker_sebas**: Estimación promedio de π es 3.1420 con una desviación estándar de 0.0058.
- **2_worker_diego_ricardo**: Estimación promedio de π es 3.1442 con una desviación estándar de 0.0072.
- **2_workers_sebas_diego**: Estimación promedio de π es 3.1414 con una desviación estándar de 0.0081.

- **3_workers_sebas_diego_ricardo**: Estimación promedio de π es 3.1466 con una desviación estándar de 0.0138.

Interpretación: Después de excluir los primeros datos con pocos puntos, las estimaciones de π son mucho más consistentes y cercanas al valor real de 3.14159. La simulación con un worker ("1_worker_sebas") ofrece la estimación más precisa y con menor variabilidad, lo que indica que, aunque usar más workers puede mejorar el tiempo, también puede introducir más variabilidad en los resultados.

2. Variabilidad en la estimación de π

- La desviación estándar varía entre 0.0058 y 0.0184, lo cual es mucho más ajustado que cuando considerábamos los puntos iniciales con menos precisión.
- A medida que el número de workers aumenta, la desviación estándar también aumenta ligeramente, lo que indica que las estimaciones son menos consistentes.

3. Duración del cálculo

- Los tiempos de duración promedio son similares a los observados en el análisis anterior, con las simulaciones que involucran más workers mostrando tiempos más largos (por ejemplo, "2_worker_diego_ricardo" tiene una duración promedio de 6.78e9 ns).
- Las simulaciones con menos workers ("1_worker_diego", "1_worker_sebas") tienen tiempos de ejecución más cortos, lo cual sugiere que el uso de múltiples workers introduce sobrecarga en la comunicación y sincronización, lo que reduce la eficiencia.

4. Eficiencia

- Aunque el uso de más workers permite distribuir la carga, no necesariamente resulta en una mejora significativa en el tiempo de cálculo para estimaciones de π con un gran número de puntos.
- La simulación con un solo worker ("1_worker_sebas") logra un equilibrio adecuado entre precisión y tiempo de ejecución, mostrando que en algunos casos menos es más cuando se trata de balancear eficiencia y precisión.

Nota: El link del repositorio es el siguiente:

https://github.com/Lrojas898/ICE_MONTECARLO