LABORATORIO

Computación en internet

Por: Stick Martinez, Juan Pablo Parra, Pablo Guzmán, Thomas Brueck, Juan Esteban Eraso

Milton Sarria Paja

ICESI-27/10/2004

Implementación de Monte Carlo utilizando el Modelo Cliente-Maestro-Trabajadores con ICE

El método de Monte Carlo se basa en el uso de simulaciones aleatorias para resolver problemas complejos. En esta implementación, se utiliza para aproximar el valor de π. Se generan puntos aleatorios en un cuadrado unitario, y se cuentan los puntos que caen dentro de un círculo inscrito. La razón entre los puntos dentro del círculo y el total proporciona una estimación de π.

Fundamento del Método

Para la aproximación de π mediante Monte Carlo, se aprovecha la geometría simple de un círculo inscrito en un cuadrado. La idea es generar puntos aleatorios dentro de un cuadrado de lado 1, y luego contar cuántos de esos puntos caen dentro del círculo inscrito en ese cuadrado.

Dado que el área del círculo es πr^2y la del cuadrado es $(2r)^2$, la relación entre los puntos en el círculo y los puntos en el cuadrado es proporcional a la relación de sus áreas:

$$\boldsymbol{\pi \approx 4 \cdot \frac{puntos\ en\ el\ circulo}{puntos\ totales}}$$

El método consiste en:

1. Generar un número grande de puntos aleatorios en el cuadrado.
2. Determinar si cada punto está dentro del círculo usando la ecuación $x^2 + y^2 \leq 1$.
3. Calcular la fracción de puntos que caen dentro del círculo para aproximar el valor de π.

Este sistema sigue el patrón maestro-trabajadores, distribuido de la siguiente manera:

- Cliente: Solicita la estimación del valor de π indicando cuántos puntos usar.
- Maestro: Divide la tarea entre varios trabajadores, asignando partes iguales a cada uno.
- Trabajadores: Calculan cuántos puntos de su subconjunto caen dentro del círculo y envían los resultados al Maestro.
- Maestro: Recopila los resultados de los trabajadores y calcula la estimación final.

Cada trabajador recibe un número de puntos igual a $\frac{puntos\ totales}{Numero\ de\ trabajadores}$.

Diseño e Implementación utilizando ICE

Estructura del Sistema Cliente-Maestro-Trabajadores con ICE

Este proyecto sigue una arquitectura distribuida basada en:

- **Cliente:** Solicita la estimación de π.
- **Maestro:** Coordina el trabajo entre varios trabajadores.
- **Trabajadores:** Realizan los cálculos paralelos y devuelven los resultados al Maestro.

El framework ICE proporciona proxies para invocar métodos de manera remota, gestionando automáticamente la comunicación entre los distintos procesos. Los nodos se comunican utilizando **interfaces generadas por ICE**, asegurando una ejecución eficiente y escalable.

2. Función del Maestro

El **Maestro** es responsable de:

1. **Recibir las solicitudes del cliente:** Cuando el cliente desea estimar π, se conecta al servidor del Maestro mediante un **proxy remoto** proporcionado por ICE.
2. **Distribuir la carga:** El Maestro divide el número total de puntos aleatorios entre los trabajadores registrados.
3. **Coordinar las llamadas a los trabajadores:** Cada trabajador recibe una parte del total de puntos para procesar y calcular cuántos caen dentro del círculo unitario.
4. **Recolectar los resultados:** El Maestro recopila los resultados parciales de cada trabajador.
5. **Calcular el resultado final:** Suma los puntos obtenidos por todos los trabajadores y devuelve el total al cliente.

Mecanismo:

- El Maestro utiliza **llamadas asíncronas a los proxies de los trabajadores** para distribuir las tareas.
- La cantidad de puntos se divide uniformemente para optimizar el cálculo paralelo.
- El Maestro también maneja excepciones en caso de que alguno de los trabajadores no responda, asegurando la continuidad del proceso.

3. Funcionamiento de los Trabajadores

Cada **trabajador** es un proceso independiente que:

1. **Recibe la solicitud del Maestro:** A través de su proxy, el trabajador acepta una cantidad de puntos para procesar.
2. **Genera puntos aleatorios:** Cada trabajador utiliza un generador aleatorio para simular puntos dentro del cuadrado de lado 1.
3. **Evalúa la condición de pertenencia al círculo:** Por cada punto generado, verifica si está dentro del círculo unitario.

4. **Devuelve el resultado parcial al Maestro:** El trabajador envía de vuelta al Maestro el número de puntos que cayeron dentro del círculo.
   Mecanismo:

- La ejecución de los trabajadores es completamente paralela, y cada uno se ejecuta en su propio servidor.
- Al ser tareas independientes, se minimiza la latencia y la sobrecarga entre trabajadores.
- Los trabajadores permanecen en estado de espera (idle) hasta recibir una nueva solicitud del Maestro.

4. Rol del Cliente

El **Cliente** actúa como el punto de entrada del sistema. Su función principal es:

1. **Conectarse al servidor Maestro:** El cliente utiliza un **proxy de tipo MasterPrx** para enviar una solicitud remota al Maestro.
2. **Enviar los parámetros de la simulación:** El cliente define cuántos puntos usar en la simulación y cuántos trabajadores se utilizarán.
3. **Recibir el resultado de la estimación:** Una vez que el Maestro ha completado la tarea, el cliente recibe el número total de puntos dentro del círculo y calcula la estimación de π.
4. **Mostrar los resultados:** El cliente imprime la aproximación de π en la consola.
   Mecanismo:

- El cliente actúa como un proceso sencillo, delegando el trabajo pesado al Maestro y los trabajadores.
- Utiliza un **modelo síncrono de solicitud y respuesta** para recibir los resultados del Maestro.

5. Proxies y Comunicación entre Componentes

El **framework ICE** facilita la creación de proxies para realizar llamadas remotas entre los componentes del sistema. Los **proxies** son objetos que permiten a los clientes, al Maestro y a los trabajadores comunicarse sin necesidad de conocer los detalles internos de la red.

- **MasterPrx y WorkerPrx:** Son proxies generados automáticamente por ICE a partir de las interfaces del Maestro y los Trabajadores.
- Estos proxies gestionan las conexiones de red y garantizan la **entrega confiable** de los mensajes.
- Las **llamadas remotas son transparentes** para los desarrolladores, facilitando la implementación del sistema distribuido.

6. Gestión de Tareas Distribuidas

El Maestro utiliza un enfoque simple de **distribución balanceada de carga**. El número total de puntos se divide de manera uniforme entre los trabajadores

disponibles. Cada trabajador realiza su parte del cálculo de forma independiente, lo que garantiza la paralelización de la tarea.

- **Modelo maestro-trabajadores:** El Maestro actúa como punto central de control, mientras que los trabajadores son procesos esclavos que ejecutan las instrucciones recibidas.
- **Manejo de fallos:** Si uno de los trabajadores no responde, el Maestro puede reasignar los puntos a otros trabajadores o manejar la falta de respuesta con mensajes de error adecuados.

- Escalabilidad: A medida que aumentan los trabajadores, disminuye el tiempo total.
- Precisión: Más puntos generan una mejor aproximación de π.
- Limitaciones: La coordinación entre muchos nodos puede generar sobrecarga en el Maestro.

Resultados y análisis:

Prueba	Número de puntos (N)	Número de trabajadores (n)	Estimación de π (3.14159)	Diferencia con π (3.14159)	Tiempo de Ejecución (s)
1	1,000	1	3.136000	0.0559	0.006
2	1,000	2	3.068000	0.0735	0.007
3	10,000	1	3.129600	0.0119	0.008
4	10,000	4	3.147200	0.0056	0.007
5	100,000	1	3.139680	0.0019	0.013
6	100,000	4	3.134080	0.0075	0.012
7	100,000	8	3.135080	0.0065	0.013
8	1,000,000	4	3.142832	0.0012	0.03
9	1,000,000	8	3.140424	0.0011	0.03
10	1,000,000	16	3.142032	0.0004	0.032

1. Análisis de la precisión de la estimación de π a medida que aumenta N

- Observación General

  Como se esperaba, la precisión de la estimación de π mejora a medida que el número de puntos N aumenta. Esto se refleja en la columna Diferencia con π (3.14159), donde la diferencia con el valor real de π disminuye conforme N incrementa.

- Resultados Específicos:

  - Para N = 1,000, la diferencia con π es relativamente alta (hasta 0.0735 en la Prueba 2 con 2 trabajadores), indicando que, con pocos puntos, la estimación es menos precisa.
  - Para N = 10,000, la diferencia con π se reduce considerablemente, quedando en un rango entre 0.0119 y 0.0056, lo que indica una mejor aproximación.

- Para N = 100,000, la diferencia sigue disminuyendo y se encuentra en el rango de 0.0019 a 0.0075, mostrando una tendencia hacia la precisión a medida que aumenta el número de puntos.
- Finalmente, para N = 1,000,000, la diferencia con π es extremadamente baja, con valores de entre 0.0012 y 0.0004, lo que representa una estimación muy precisa de π.

La precisión del método de Monte Carlo para estimar π mejora significativamente con un mayor N, lo cual es consistente con el comportamiento esperado de este método probabilístico. Esto demuestra que el sistema cumple correctamente con la estimación de π, acercándose al valor real conforme aumentan los puntos.

2. Análisis del rendimiento y escalabilidad con diferentes configuraciones de trabajadores (n)

- Observación del Tiempo de Ejecución

    - Para N = 1,000 y N = 10,000, el tiempo de ejecución es bajo en general (0.006 a 0.008 segundos), y no hay una mejora notable al incrementar el número de trabajadores. Esto se debe a que la cantidad de puntos es pequeña y el tiempo de comunicación entre maestro y trabajadores puede compensar el tiempo ganado por la distribución de la carga.
    - Para N = 100,000, se observa una ligera variación en el tiempo de ejecución al aumentar el número de trabajadores. Aunque el tiempo no mejora significativamente al pasar de 1 a 8 trabajadores, las pruebas sugieren que la carga se distribuye efectivamente.
    - Para N = 1,000,000, el tiempo de ejecución aumenta (0.03 a 0.032 segundos), lo cual es esperado debido al mayor número de puntos. En este caso, el incremento en el número de trabajadores (de 4 a 16) parece ayudar a mantener el tiempo de ejecución estable, indicando que la implementación es escalable y capaz de manejar mayores cargas de trabajo.

- Escalabilidad y Eficiencia

    - Para configuraciones pequeñas de N (como 1,000 y 10,000 puntos), el beneficio de añadir más trabajadores es mínimo debido a la baja carga de trabajo.
    - Para configuraciones mayores de N (como 1,000,000 puntos), el sistema muestra un rendimiento estable al aumentar el número de trabajadores, manteniendo el tiempo de ejecución en un rango controlado. Esto indica que la implementación puede escalar para manejar mayores cargas con múltiples nodos (trabajadores).
    - Aunque el aumento en el número de trabajadores ayuda a mantener el tiempo de ejecución estable para configuraciones grandes, la ganancia marginal disminuye cuando el número de trabajadores es excesivo en relación con la tarea (por ejemplo, N = 100,000 con 8 trabajadores), debido a la sobrecarga de comunicación.

La implementación muestra una escalabilidad adecuada, con tiempos de ejecución que se mantienen bajo control al incrementar el número de trabajadores, especialmente para configuraciones grandes de N. Sin embargo, para configuraciones pequeñas, el beneficio de agregar más trabajadores es limitado debido a la sobrecarga de comunicación.