Búsqueda Distribuida de Números Perfectos utilizando ICE y el Modelo Cliente-Maestro-Trabajadores

Objetivo:

El objetivo de este proyecto es que los estudiantes implementen un sistema distribuido para encontrar números perfectos en un rango dado alto, haciendo uso del modelo Cliente-Maestro-Trabajadores (Master-Workers) con llamadas asíncronas sobre ICE (Internet Communications Engine). El sistema debe permitir escalar la computación distribuyendo el trabajo entre varios nodos trabajadores y permitir al cliente consultar de manera eficiente los resultados.

Descripción del Problema:

Un número perfecto es aquel que es igual a la suma de sus divisores propios positivos, excluyéndose a sí mismo. Por ejemplo, 28 es perfecto porque $1 + 2 + 4 + 7 + 14 = 28$. Este proyecto busca encontrar todos los números perfectos en un rango específico, de forma eficiente y paralela.

Debido a que el problema es altamente costoso computacionalmente (su complejidad es $O(n\sqrt{n})$), se utilizará una arquitectura distribuida donde:

- Un cliente solicita encontrar todos los números perfectos en un rango determinado.
- Un maestro divide el rango en subrangos para ser procesados.
- Múltiples trabajadores procesan cada subrango y devuelven los números perfectos encontrados al maestro.

Requisitos Funcionales:

- Cliente:
  - Debe permitir ingresar el rango de búsqueda (inicio y fin).
  - Debe enviar una petición al maestro indicando el rango.
  - Debe esperar de forma asíncrona la respuesta con los números perfectos encontrados.

- Debe mostrar al usuario el resultado final y el tiempo de ejecución.

- Maestro:
  - Debe recibir las solicitudes del cliente.
  - Debe dividir el rango en partes iguales entre el número de trabajadores disponibles.
  - Debe enviar a cada trabajador un subrango para procesar.
  - Debe recibir las respuestas de los trabajadores de forma asíncrona.
  - Debe consolidar los resultados y enviarlos al cliente.
- Trabajador:
  - Debe recibir un subrango (inicio, fin).
  - Debe realizar el cálculo de todos los números perfectos dentro del subrango.
  - Debe devolver la lista de números perfectos encontrados al maestro.

Implementación Técnica:

- Utiliza ICE (ZeroC) como middleware para comunicaciones distribuidas.
- Las llamadas entre maestro y trabajadores deben ser asíncronas.
- El sistema debe manejar errores de comunicación y fallos de nodos de forma robusta.
- Se debe permitir modificar fácilmente el número de trabajadores.
- El código debe estar organizado en módulos: Cliente, Maestro, Trabajador, y definición de interfaces en Slice.

Pruebas y Evaluación:

- Realizar pruebas con distintos rangos de búsqueda (por ejemplo, hasta 10,000; 100,000; 1,000,000).
- Evaluar el rendimiento y escalabilidad variando el número de trabajadores (2, 4, 8...).
- Medir y reportar el tiempo total de procesamiento y la eficiencia.

- Verificar que el sistema entrega correctamente los números perfectos en cada rango.

Entregables:

1. Informe final en PDF que incluya:
    - Descripción del problema y análisis del algoritmo utilizado.
    - Arquitectura general del sistema distribuido.
    - Detalle del diseño Cliente-Maestro-Trabajadores con ICE.
    - Explicación del mecanismo de distribución del rango y coordinación.
    - Resultados experimentales y análisis de rendimiento.
    - Conclusiones y posibles mejoras.
2. Código fuente completo:
    - Interfaces Slice.
    - Implementaciones del Cliente, Maestro y Trabajadores.
    - Scripts para compilar y ejecutar el sistema distribuido.
3. Instrucciones de ejecución:
    - Detalles para compilar el Slice.
    - Cómo correr cada componente.
    - Parámetros que deben ingresarse al cliente.

Notas Adicionales:

- Se valorará positivamente la modularidad, claridad del código y uso correcto de las interfaces ICE.
- Se recomienda implementar primero una versión sincrónica para pruebas y luego transformarla a asincronía.
- Puede utilizar herramientas de logging para ver el progreso de cada trabajador.
- Si se desea, puede implementar una interfaz gráfica para el cliente, pero no es obligatorio.