TRABAJO Nº1 PARALELA

UTEM

UNIVERSIDAD
TECNOLÓGICA
METROPOLITANA

del Estado de Chile

integrantes: Sebastian Lopez
Ezequiel Molina
Matias Sanchez
Profesor: Sebastián Salazar Molina
Sección: 411

Introducción

En el presente trabajo se responderá la solicitud del Rey Aurelius IV del reino de eldoria. Este trabajo surge de la necesidad de los "Protectores del Reino" de restaurar un sistema de identificación fiable y mejorar las vías de comunicación para poder regresar a su mundo natal. El reino de Eldoria, afectado por la falta de un sistema de identidad ciudadana y caminos peligrosos , requiere soluciones que permitan la gestión de su población y la optimización de sus rutas.
Se presenta la formulación del problema, el tratamiento de los datos, el enfoque de solución adoptado y los resultados obtenidos, demostrando la aplicabilidad práctica de los modelos de transporte en la planificación de sistemas complejos.

1. **Diseño y Solución Paralela**

Para aprovechar al máximo los recursos de cómputo disponibles y acelerar el procesamiento del archivo CSV, el programa se divide en dos grandes componentes: el módulo de lectura y parseo paralelo y el módulo de agregación de resultados.
El programa fue implementado en Lenguaje de Programación c + + , con un enfoque explícito en el paralelismo.

- **Arquitectura general**
    1. **Entrada**

    Archivo CSV con N registros, cada uno convertido a un struct Registro.
    2. **Partición de datos**
    a. Se calcula el tamaño total del archivo (bytes).
    b. Se determina el número de hilos (P) según los núcleos de CPU disponibles.
    c. Se divide el archivo en P bloques contiguos de tamaño aproximadamente igual.
    3. **Procesamiento concurrente**

    Se lanzan P hilos, cada uno ejecutando procesarBloque(ruta, inicio, fin, &resultados) sobre su porción de bytes.

    Cada hilo:
        a. Ajusta su posición de lectura al inicio de la primera línea completa.
        b. Lee línea a línea hasta alcanzar el final de su bloque.
        c. Parsea el CSV y actualiza estructuras locales (o globales protegidas) con estadísticas.
    4. **Sincronización y reducción**
        a. Para proteger el acceso a las estructuras compartidas (por ejemplo, contador de estratos), se utiliza un std::mutex con lock_guard.
        b. Alternativamente, pueden usarse buffers locales por hilo y luego combinarse (reducción en árbol) para minimizar la contención.
    5. Una vez finalizados todos los hilos, se recopilan y consolidan los resultados para su presentación o uso en posteriores análisis (menú de consultas).

Estructura del Código:

1. Lectura del archivo

Este programa implementa un sistema de lectura paralela de archivos CSV. Divide el archivo en bloques según la cantidad de núcleos del sistema y asigna a cada hilo la lectura de un bloque, asegurando que las líneas no se corten. Cada línea es parseada y transformada en un objeto Registro, conteniendo campos específicos. Esto permite un procesamiento eficiente de grandes volúmenes de datos en menos tiempo.
Funcionamiento de la lectura:
Detección de hilos disponibles:
Se consulta el número de núcleos del sistema mediante `unsigned int n = thread::hardware_concurrency();`
Si no se puede determinar, se utilizan por defecto 2 hilos.
Cálculo del tamaño del archivo:
Se obtiene el tamaño total del archivo en bytes para definir el rango de lectura de cada hilo.
División en bloques:

El archivo se divide en bloques de igual tamaño según la cantidad de hilos disponibles. A cada hilo se le asigna un bloque con una posición inicial (inicio) y final (fin).

Lectura de cada bloque por hilo:

Cada hilo accede directamente a su sección del archivo utilizando seekg(). Si el bloque no comienza al inicio del archivo, el hilo se asegura de avanzar hasta el próximo salto de línea para evitar leer una línea incompleta.

Parseo de líneas:

Cada hilo lee línea por línea dentro de su rango y transforma cada una en una estructura Registro, que almacena campos como identificador, especie, género, nombre, etc.

Almacenamiento local:

Los registros procesados por cada hilo se guardan en un vector, lo que permite mantener la independencia de lectura y luego fusionar los resultados si es necesario.

2. **Módulo de visualización y navegación del usuario mostrar menu**

Función principal: mostrar_menu()

Permitir al usuario interactuar con el programa mediante una interfaz textual basada en un menú de opciones.

Funcionamiento:

- Presenta un conjunto de consultas posibles sobre los datos leídos, relacionadas con estratificación social, edad, movilidad y demografía.
- Permite al usuario ingresar un número para seleccionar una opción.
- Cada opción está preparada para invocar una funcionalidad estadística o analítica (aún por implementar).
- Utiliza un bucle goto para volver al menú después de cada operación, hasta que el usuario elige salir.

3. **Módulo de utilidades y auxiliares**

Incluye funciones pequeñas de apoyo al funcionamiento general del sistema:

linea_separadora(): Imprime una línea divisoria para mejorar la legibilidad del menú.

Manejo básico de errores por entrada incorrecta del usuario.

Preparación para expandir funcionalidades de análisis mediante llamadas a funciones aún no desarrolladas (como conteo de estratos, cálculo de edad, etc.).

Conclusiones

A través de este trabajo se sentaron las bases para construir una herramienta capaz de leer y procesar grandes volúmenes de datos de forma eficiente. Aprovechando la capacidad de los computadores modernos para ejecutar múltiples tareas al mismo tiempo, se implementó una estrategia de paralelización que divide el archivo entre varios hilos, permitiendo acelerar la lectura y el análisis de la información.
Se diseñó un menú interactivo con diversas preguntas orientadas a entender mejor la realidad social, demográfica y territorial del mundo ficticio de Eldoria. Estas preguntas, que abarcan desde la distribución por estratos hasta análisis de edades y patrones de movilidad, ya han sido abordadas y resueltas en términos funcionales, quedando su redacción pendiente de integración al presente informe.

Más allá de la parte técnica, este proyecto también puso en práctica habilidades de diseño lógico, programación paralela y manipulación de datos, demostrando que es posible construir herramientas potentes y flexibles con una base bien estructurada. El sistema desarrollado está listo para ser extendido, visualizado o adaptado a otros contextos sin grandes modificaciones.