Universidad Tecnológica Metropolitana
Departamento de Informática
Primer semestre 2024
Sede Macul

Informe
Computación Paralela y Distribuida

Asignatura: Computación Paralela y Distribuida
Profesor: Sebastián Salazar Molina
Sección: 411
Integrantes:

Javier Bravo
Pablo Castillo
Tomás Díaz-Valdés

Fecha de entrega: 17 de julio 2024

Índice

Introducción

La inflación es un fenómeno económico que afecta a todos los países, alterando el poder adquisitivo de las personas y el costo de vida. Comparar la inflación entre diferentes países puede proporcionar una visión sobre sus economías y las políticas económicas aplicadas. En este contexto, el presente informe tiene como objetivo analizar y comparar la inflación de Chile y Perú utilizando técnicas de computación paralela.

La computación paralela proporciona las capacidades necesarias para manejar grandes volúmenes de datos de manera eficiente, permitiendo el procesamiento simultáneo de múltiples tareas y mejorando así los tiempos de ejecución y la utilización de recursos. En este proyecto, hemos desarrollado un programa en C++ utilizando la biblioteca OpenMP para paralelizar el proceso de cálculo de la inflación basado en los precios de una canasta de productos. Esta paralelización nos permite mejorar significativamente los tiempos de ejecución y la eficiencia del procesamiento de datos.

A través de este proyecto, buscamos no solo cumplir con los objetivos de la asignatura, sino también estudiar el entendimiento de la inflación en ambos países y cómo puede ser medida y comparada utilizando herramientas modernas de computación.

Enunciado

Inflación Chile vs Perú

Existen fundadas dudas sobre la fiabilidad de los datos de inflación, siendo un tema sensible, se recurre a su grupo de la asignatura de Computación Paralela y Distribuida para hacer una comparación de la inflación de Perú y Chile, hallando la inflación de dichos países a través de los precios de una canasta de productos, que se encuentran en un archivo de texto plano comprimido, disponible para descargar desde este enlace:
https://drive.google.com/file/u/0/d/1id7EMrzPiu-zZvC2yrISjE_wJ21DRXBG/view?usp=sharing&pli=1

Una vez descomprimido el archivo, se encontrará un fichero separado por punto y coma (';'), cuyos datos está entre comillas dobles (que hacen la función de separador de celdas) con la siguiente estructura:

1. **Fecha de creación**, ejemplo "2023-04-24 18:31:31.310237", esta es una representación de fecha y hora en formato ISO 8601 extendido. La parte antes del espacio ("2023-04-24") es la fecha en formato año-mes-día, mientras que la parte después del espacio ("18:31:31.310237") representa la hora, los minutos, los segundos y los milisegundos en formato hora:minuto:segundo.milisegundo.
2. **Número de Boleta**, ejemplo "171321", es un número entero que identifica y agrupa los productos adquiridos en una misma compra.
3. **Número de tienda**, ejemplo "192", es un número entero que representa a la tienda en que se realizó la transacción.
4. **Nombre de fantasía de la tienda**, ejemplo "MAXICOMPRA", es un texto con la etiqueta de la tienda que realizó la venta.
5. **Categoría de la tienda**, ejemplo "736", es un texto o un número que representa la categoría del producto.
6. **Tipo de envío**, ejemplo "Despacho a domicilio", es un texto que indica la forma en que el producto fue entregado al cliente.
7. **Identificador del producto**, ejemplo "100101", es un texto que representa el identificador del producto en la tienda.
8. **Cantidad**, ejemplo "1", es un ordinal que indica las unidades adquiridas del producto.
9. **Nombre del producto**, ejemplo Refrigeradora MabeTop Freezer RMA255FYPG 239L Grafito, es el texto que indica el nombre comercial del producto.
10. **Monto**, ejemplo "1500.99", un número decimal (el punto es el separador decimal) que indica el valor en soles peruanos (PEN) del producto.

Para los cálculos, se debe usar el siguiente link con el archivo con las paridades históricas entre el Sol Peruano y el Peso Chileno:
https://docs.google.com/spreadsheets/d/1wBHOOiTiNqlb8QZsL0uoK4es7eSz33sj/edit?usp=sharing&ouid=116003055334086306 82&rtpof=true&sd=true

Además, se debe hacer una comparación de las inflaciones calculadas de cada país respecto a la inflación real de cada uno, considerando que la canasta de productos entregada difiere respecto a la canasta original con la que se calcula la inflación real de cada país.
Para lo anterior, se debe hacer uso de la librería OpenMP en lenguaje C/C++, la que permitirá paralelizar el proceso de cálculo. También deberá ser calculado con código secuencial, para comparar las métricas explicadas en la teoría de la asignatura.
El programa debe desplegar la siguiente información:

- Inflación real entre ambos países e inflación calculada.
- Métricas de performance entre código secuencial y paralelo.
- Los tiempos anteriores sólo serán de la lógica de cálculo de la inflación, no deberán medirse tiempos de declaración de variables, impresiones gráficas, ingreso de datos, o cualquier parte del programa que no sea parte de la lógica de inflación.

Especificaciones técnicas del PC

- CPU: Procesador Intel® Core™ i5-10400F 2.90 GHz.
- Memoria RAM: 16GB RAM (2 x 8192 MB) a 2666 Mhz.
- Almacenamiento:
    - Principal: SSD Kingston SA400S37120G (120.0 GB), 500MB/seg (lectura) y 320MB/seg (escritura)
    - Secundario: HDDToshiba HDWD110 (1000.2 GB), ratio de transferencia 6 Gbps, velocidad del disco 7200 rpm.
- Sistema Operativo: Windows 11 Pro 64 bits.

Desarrollo

Para empezar a desarrollar el programa en lenguaje C++, se utilizó el entorno de desarrollo integrado (IDE) y editor de código Embarcadero Dev-C++. Este utiliza el puerto Mingw de GCC (Colección de Compiladores GNU) como su compilador, y al ser una aplicación nativa de Windows, ocupa poco espacio de memoria.

Como primera instancia, al empezar a desarrollar el programa, se revisó el archivo CSV que contenía la información de Perú, con el programa EmEditor, que es un editor de texto avanzado para Windows, con capacidad para manejar archivos de gran tamaño de manera eficiente. Con este se pudo ver que el archivo no estaba ordenado por fecha, así que se tuvo que idear y planificar cómo trabajar con ese archivo.
Con los primeros códigos que realizamos (que se encuentran en el repositorio de GitHub), se logró realizar unos primeros esbozos de cálculos, pero se tenía conciencia de que estaban errados.
Se tuvo la idea de ordenar el archivo con algún algoritmo de ordenamiento, pero tomó muchos intentos y tiempo, y no hubo éxito. Entonces, surgió la idea de dividir ese archivo principal por años, que fue la forma elegida para realizar el programa, y que funcionó. Sin embargo, este no quedaba ordenado, así que nuevamente se dividió, esta vez por meses por año, quedando así un archivo dividido y manejable para poder realizar los cálculos.
También, después de dividirse por meses, el programa hace que vuelvan a juntarse por años, pero esta vez ordenados cronológicamente cada mes, y al momento de empezar a realizar los cálculos de las canastas básicas, se ocupan estos archivos divididos por mes, en vez de por años, lo que ayuda a quitarle tiempo de cálculo.
En resumidas cuentas, el programa se basa en dividir el archivo principal por años, luego por meses, y después, juntarlos por años, de manera ordenada.

Finalmente, el programa realizado quedó de la siguiente manera, y sigue los siguientes pasos en su ejecución:

1. **Inicialización y Configuración:**

- El programa comienza configurando la localización para el entorno de C++ y estableciendo el número máximo de hilos disponibles para OpenMP.
- Inicia un temporizador para medir el tiempo total de ejecución del programa.

2. **División del Archivo Principal:**

- Función splitFileByYearDirectly: Lee el archivo pd.csv y lo divide en varios archivos más pequeños, uno para cada año, desde 2021 hasta 2024.
- Crea un directorio de salida (output) y escribe los datos correspondientes a cada año en archivos separados.

3. **División de Archivos Anuales en Mensuales:**
- Función processYearFile: Cada archivo anual se procesa para dividirlo en archivos mensuales. Se paraleliza esta tarea usando OpenMP para mejorar la eficiencia.

4. **Procesamiento de SKUs Comunes:**
- Funciones getCommonSKUsForYear y saveCommonSKUs: Para cada año (2021-2024), se identifican y guardan los SKUs (productos) comunes que aparecen en múltiples meses.
- Los SKUs comunes se almacenan en archivos CSV específicos para cada año.

5. **Cálculo de Costos Mensuales de la Canasta Básica:**
- Función calculateMonthlyBasketCosts: Calcula los costos mensuales de la canasta básica utilizando los SKUs comunes identificados previamente.
- Los resultados se almacenan en un mapa (monthlyCosts) que asocia fechas con costos mensuales.

6. **Cálculo de la Inflación:**
- Función calculateInflation: Calcula la inflación intermensual y anual a partir de los costos mensuales de la canasta básica.
- Los resultados de la inflación se almacenan en un archivo de texto (output/resultados.txt).

7. **Lectura y Promedio de Tasas de Cambio:**
- Función readExchangeRates: Lee las tasas de cambio desde un archivo CSV (PEN_CLP.csv).
- Función calculateMonthlyAverages: Calcula los promedios mensuales de las tasas de cambio entre dos fechas especificadas.

8. **Conversión de Resultados a CLP:**
- Función convertAndSaveResultsToCLP: Convierte los resultados de los costos mensuales de la canasta básica a pesos chilenos (CLP) utilizando las tasas de cambio calculadas previamente.
- Los resultados convertidos se guardan en archivos de salida específicos para cada año.

9. **Impresión de Resultados:**
- El programa imprime los contenidos de varios archivos de resultados, mostrando los costos de la canasta básica mensual y la inflación calculada en diferentes años.

10. **Finalización:**

- Detiene el temporizador y calcula la duración total de la ejecución del programa.
- Imprime el tiempo total de ejecución.

Al descomprimir el archivo, se obtiene el respectivo CSV con las fechas correspondientes, que deben ser divididas en archivos CSV más pequeños, tomando la columna de fechas del archivo original, para luego ser organizados, primero por años y luego por meses, utilizando técnicas de paralelización con OpenMP para mejorar la eficiencia.

Para llevar a cabo el análisis y procesamiento de datos, se desarrollaron diversas funciones específicas que abarcan desde la manipulación básica de cadenas hasta el cálculo de la inflación y la conversión de moneda. Estas funciones están organizadas en varias categorías según su propósito y funcionalidad. A continuación, se detallan las funciones utilizadas en cada categoría, las cuales son esenciales para el correcto funcionamiento del programa y la obtención de resultados precisos:

1. **Funciones de utilidad**

- **“split”**: Divide una cadena de texto en tokens usando un delimitador específico, manejando correctamente las comillas que encapsulan los valores con delimitadores internos.
- **“trim”**: Elimina espacios en blanco y comillas de una cadena de texto.
- **“getYear”**: Extrae el año de una fecha en formato “YYYY-MM-DD”.
- **“getYearMonth”**: Extrae el año y el mes de una fecha en formato “YYYY-MM-DD”.

2. **División de archivos por año**

- **“splitFileByYearDirectly”**: Esta función toma un archivo CSV grande y lo divide en varios archivos más pequeños, uno por cada año, dentro de un rango específico.

3. **Procesamiento de archivos anuales en mensuales**

- **“processYearFile”**: Esta función toma un archivo anual y lo divide en archivos mensuales.

4. **Manejo de SKUs**

- **“readSKUs”**: Esta función lee los detalles de los SKUs desde un archivo CSV.
- **“getCommonSKUsForYear”**: Esta función obtiene los SKUs comunes para un año específico a partir de múltiples archivos mensuales.
- **“saveCommonSKUs”**: Esta función guarda los SKUs comunes en un archivo CSV.

5. **Cálculo de costos e inflación**

- **“calculateMonthlyBasketCosts”**: Esta función calcula los costos mensuales de la canasta básica.
- **“saveResultsToFile”**: Esta función guarda los resultados de los costos y la inflación en un archivo de texto.
- **“calculateInflation”**: Esta función calcula la inflación intermensual y anual.

6. **Conversión de moneda**

- **“readExchangeRates”**: Esta función lee las tasas de cambio desde el archivo CSV.
- **“calculateMonthlyAverages”**: Esta función calcula los promedios mensuales de las tasas de cambio entre dos fechas.
- **“convertAndSaveResultsToCLP”**: Esta función convierte los resultados a CLP y los guarda en un archivo de salida.

7. **Función principal**

- La función principal “main” implementa el flujo de trabajo completo, desde la lectura y división de archivos hasta el cálculo y almacenamiento de resultados.

Tipo de estrategia utilizada:

La estrategia utilizada para abordar el problema fue paralelizar un programa secuencial. La creación del programa se dividió en 2 fases:

1. **Manipulación de Archivos:**

Primero, se desarrolló la manipulación del archivo con los productos de las canastas básicas, dividiendo el archivo original en archivos más pequeños, según el año y el mes. Después, se paralelizó para mejorar la eficiencia.

2. **Calculos y conversión a pesos chilenos:**

Posteriormente, se implementan funciones adicionales para obtener los SKUs (por cada año, y uno para todos los años), calcular los montos de cada uno, comparar los meses para obtener los porcentajes y calcular la inflación. Luego, se buscó cómo paralelizar para optimizar el rendimiento.

Porcentajes de paralelización y secuencial del programa

Alrededor de un 20% del programa está paralelizado mientras que el 80% restante es secuencial.

Testeo del programa:

Después de compilar y obtener los resultados esperados utilizando el editor de código Embarcadero, se realizó la prueba en un entorno de GNU/Linux, específicamente Lubuntu versión 24.04, en una máquina virtual del software Oracle VM VirtualBox.

Antes de compilar:

1. Comprobar la instalación del compilador de C++ llamado G++ en Linux.

```
sudo apt-get update
sudo apt-get install g++
g++ --version
```

2. Verificar la instalación de la biblioteca libopm-dev que facilita la utilización de OpenMP.

```
sudo apt-get install libomp-dev
```

3. Comprobar la instalación de la herramienta CMake para gestionar la configuración y construcción del proyecto, lo que facilita su portabilidad.

   ```
   sudo apt-get install cmake
   ```

   ```
   cmake --version
   ```

4. Asegurar que los archivos de datos “pd.csv” y “PEN_CLP.csv” estén en la carpeta principal del proyecto.

Pasos para compilar:

1. Acceder al terminal y usar “cd” para acceder a la carpeta del proyecto:

   ```
   cd /ruta/a/la/carpeta/del/proyecto
   ```

2. Ejecutar el siguiente comando que lee el archivo CMakeLists.txt con las instrucciones sobre cómo construir el proyecto, qué versión de C++ ocupar, entre otras cosas:

   ```
   cmake .
   ```

3. Una vez realizado, se compila usando el siguiente comando

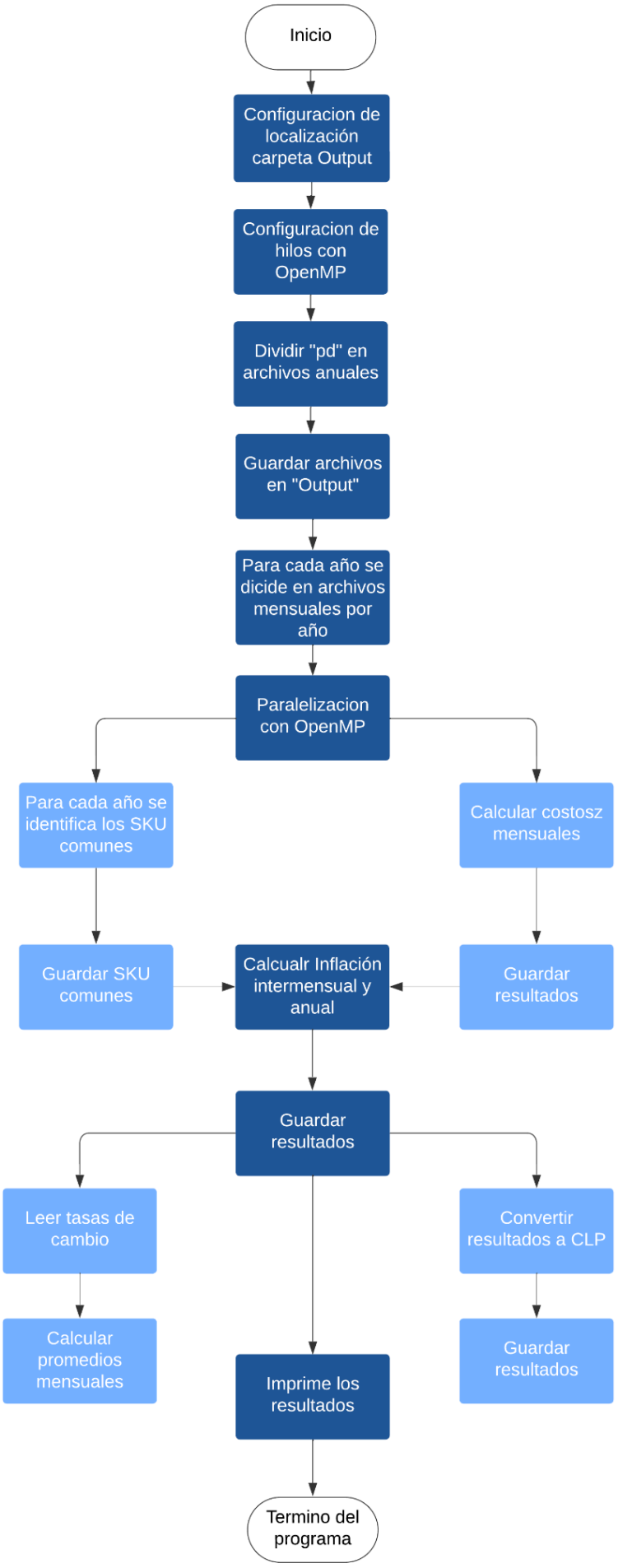
   ```
   make
   ```

4. Por último, ejecutar el programa con el siguiente comando que es el nombre indicado en CMakeLists.txt:

   ```
   ./mi_programa
   ```

Anexo

Se adjunta el diagrama de flujo detallado del programa, que ilustra de manera gráfica los pasos seguidos en la ejecución del código. Este diagrama ayuda a visualizar la estructura y el flujo de control del programa, proporcionando una comprensión clara de las diferentes etapas y la secuencia de operaciones realizadas.


Inicio
Configuracion de localización carpeta Output
Configuracion de hilos con OpenMP
Dividir "pd" en archivos anuales
Guardar archivos en "Output"
Para cada año se dicide en archivos mensuales por año
Paralelizacion con OpenMP
Para cada año se identifica los SKU comunes
Calcular costosz mensuales
Guardar SKU comunes
Calcualr Inflación intermensual y anual
Guardar resultados
Guardar resultados
Leer tasas de cambio
Convertir resultados a CLP
Calcular promedios mensuales
Imprime los resultados
Guardar resultados
Termino del programa

Conclusiones

El desarrollo y ejecución del proyecto para calcular y comparar la inflación de Chile y Perú utilizando técnicas de computación paralela ha permitido demostrar la eficacia y beneficios de estas tecnologías en el manejo y procesamiento de grandes volúmenes de datos. La implementación del programa en C++ utilizando la biblioteca OpenMP mostró mejoras significativas en los tiempos de ejecución y eficiencia en comparación con una ejecución secuencial.

Dividir el archivo de datos principal por años y meses y procesarlos de manera paralela resultó ser una estrategia efectiva para optimizar el rendimiento y manejar los datos de forma más eficiente que elegir un algoritmo de ordenamiento y empezar a hacerlo. Ya que estos pueden llegar a ser cuadráticos y su tiempo de ejecución (como fue en nuestro caso) extenderse por horas sin llegar a un buen resultado

Gracias al uso de técnicas de paralelización estas no solo facilitaron el procesamiento de los datos, sino que también permitió obtener resultados precisos en un tiempo considerablemente menor al esperado al tener un archivo tan grande que procesar y manipular.

En resumen, este proyecto no solo cumplió con los objetivos académicos de la asignatura, sino que también aportó una comprensión más profunda de la inflación en Chile y Perú y cómo puede ser medida y comparada utilizando herramientas modernas de computación. La metodología seguida y los resultados obtenidos proporcionan una guía detallada y práctica para futuros trabajos en el campo de la computación paralela, subrayando la importancia de estas tecnologías en la resolución de problemas complejos del mundo real.