Programación Paralela

Proyecto final

Análisis Paralelo de Datos Financieros

- Integrantes del equipo

Juan Sebastian 2023-1121

Ryan Ortiz 2024-0246

Keison Cerda 2024-0228

Hector Martinez 2023-1789

- Líder:

Keison Cerda 2024-0228

- Fecha de entrega

12/12/2026

INDICE

Tabla de contenido

1. Introducción

Análisis Paralelo de Datos Financieros mediante División Recursiva de Series de Tiempo

Procesar y analizar grandes series temporales de precios mediante un enfoque divide-and-conquer recursivo que fragmenta la serie en segmentos y procesa cada segmento en paralelo para calcular indicadores y. Los resultados parciales se combinan para formar métricas globales consistentes.

Objetivo General

Diseñar e implementar un sistema de análisis paralelo para procesar grandes series temporales financieras mediante un enfoque divide-and-conquer recursivo, con el fin de calcular métricas estadísticas y financieras como por ejemplo: promedio, desviación estándar, volatilidad, máximos/mínimos y retornos, de forma eficiente y escalable, evaluando el desempeño comparado entre la ejecución secuencial y paralela.

.

Objetivos Específicos

1. Implementar un módulo de lectura y generación de datasets simulados

Con el objetivo de producir series temporales financieras sintéticas de gran tamaño que permitan probar el comportamiento del sistema bajo distintas condiciones de volatilidad y volumen de datos.

2. Desarrollar un algoritmo divide-and-conquer que fragmente la serie temporal en segmentos procesables en paralelo

Permitiendo que cada tarea calcule métricas estadísticas locales y que los resultados parciales puedan combinarse para obtener métricas globales coherentes.

3. Construir un módulo de análisis secuencial y otro paralelo

Para comparar el rendimiento entre ambos enfoques y validar los beneficios del procesamiento concurrente en series temporales extensas.

4. Integrar un componente de sincronización y combinación de resultados

Que permita fusionar las métricas locales obtenidas en cada hilo, garantizando consistencia en métricas globales como media, varianza, desviación estándar, máximos, mínimos y volatilidad.

2. Descripción del Problema

Los mercados financieros generan enormes cantidades de datos en forma de series temporales, como precios y volúmenes que cambian cada segundo. Analizar estos datos con métodos tradicionales resulta lento e ineficiente cuando el tamaño del dataset crece, dificultando obtener métricas importantes como retornos, volatilidad o promedios. Este problema requiere soluciones capaces de procesar grandes volúmenes de información de manera rápida y eficiente.

El análisis paralelo de datos financieros llegaría a tener múltiples aplicaciones directas en industrias donde el tiempo de procesamiento es crítico. En el trading algorítmico, por ejemplo, los sistemas deben evaluar miles de señales basadas en series de precios en segundos para ejecutar decisiones de compra o venta. En el sector bancario, los modelos de gestión de riesgos requieren calcular volatilidades y distribuciones de retornos sobre enormes ventanas de datos para estimar métricas. Asimismo, empresas de análisis financieros y plataformas de inversión utilizan estos cálculos para identificar tendencias, detectar anomalías, realizar simulaciones Monte Carlo y evaluar el comportamiento de activos bajo diferentes condiciones del mercado.

El paralelismo es fundamental para este tipo de problema debido a la naturaleza masiva y continua de los datos financieros. Dividir la serie temporal en múltiples segmentos independientes permite que cada uno sea analizado simultáneamente por diferentes núcleos de la CPU. Esto conduce a tiempos de respuesta significativamente menores, especialmente cuando se trabajan datasets que superan cientos de miles o millones de registros.

3. Cumplimiento de los Requisitos del Proyecto

Justificación del cumplimiento de las especificaciones

1. **Ejecución simultánea de múltiples tareas:** El enfoque recursivo utilizado permite dividir el dataset financiero en múltiples segmentos independientes que pueden procesarse al mismo tiempo. Cada subdivisión del problema genera tareas paralelas que se ejecutan simultáneamente en distintos núcleos de la CPU.
2. **Incluir la necesidad de compartir datos entre tareas:** El análisis financiero propuesto requiere que todas las tareas paralelas trabajen sobre partes diferentes del mismo dataset, lo cual crea una necesidad natural de compartir datos. Aunque cada tarea procesa un segmento específico, los resultados parciales deben reunirse y combinarse para generar métricas globales coherentes como promedio total, desviación estándar, máximos o mínimos.
3. **Tener la capacidad de escalar con más recursos:** El enfoque recursivo-paralelo permite que el proyecto escale eficientemente a medida que aumenta la cantidad de datos o la disponibilidad de recursos de hardware. Al incrementarse el número de núcleos del procesador, el sistema puede generar más tareas paralelas que se ejecutan simultáneamente, reduciendo aún más los tiempos de procesamiento.
4. **Incluir métricas de evaluación del rendimiento:** Permite medir fácilmente la diferencia entre procesar los datos de forma secuencial y de forma paralela. Al trabajar con un dataset grande, es posible registrar tiempos de ejecución, calcular el speedup y comparar cómo cambia el rendimiento cuando se usan más tareas o más núcleos.
5. **Aplicar a escenarios del mundo real:** El análisis paralelo de series temporales financieras es un problema real en sectores como la banca, el trading algorítmico, la predicción de mercados y la gestión de riesgos. En estos escenarios, se deben procesar grandes volúmenes de datos históricos con rapidez para generar métricas, detectar patrones y apoyar decisiones críticas. El proyecto replica esta necesidad, ya que trabaja con datos financieros reales o simulados y aplica técnicas paralelas para acelerar el análisis

4. Diseño de la Solución

La arquitectura del sistema está organizada en varios módulos que trabajan juntos para permitir el análisis paralelo de series temporales financieras. Cada módulo cumple una función específica y se integra con los demás de manera coordinada:

1. Módulo de DataLoader

Responsable de generar y cargar los datasets utilizados en el análisis. Lee archivos CSV simulados en formato double[], listas para ser procesadas tanto por el análisis secuencial como por el paralelo.

2. Módulo de Statistics

Encargado de calcular métricas estadísticas y financieras como promedio, máximo, mínimo, desviación estándar, volatilidad y retornos. Es usado por ambos enfoques: secuencial y paralelo.

3. Módulo de Análisis Secuencial

Implementa el procesamiento tradicional de la serie temporal de inicio a fin en un solo hilo. Sirve como referencia para comparar el rendimiento contra la versión paralela.

4. Módulo de Análisis Paralelo (Divide-and-Conquer)

Fragmenta la serie temporal en segmentos y los procesa simultáneamente en múltiples hilos. Cada segmento genera métricas locales, que luego se combinan para formar un resultado global coherente.

5. Módulo de Sincronización y Combinación

Recopila los resultados parciales generados en paralelo y los integra en una única métrica final por medio de reducción, garantizando consistencia en estadísticas globales.

6. Módulo de TimerUtil y Métricas

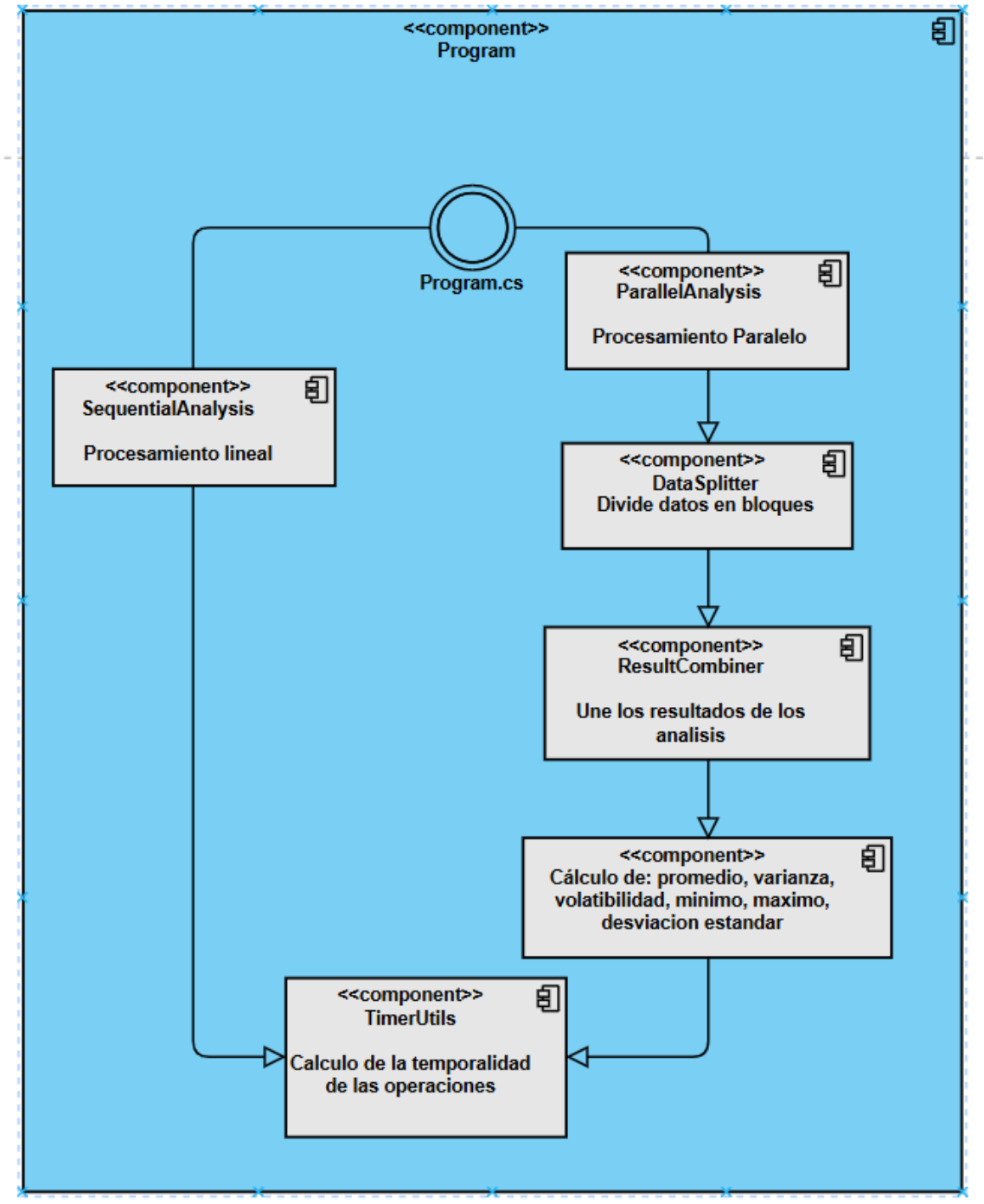
Mide los tiempos de ejecución tanto del análisis secuencial como del paralelo

7. Capa de Presentación / Consola

Interfaz simple donde se ejecutan las pruebas, se muestran resultados y se comparan los tiempos entre ambos enfoques.

La estrategia de paralelización empleada se basa en el modelo Divide and Conquer. Este enfoque consiste en descomponer el problema principal en subproblemas más pequeños e independientes que pueden ejecutarse en paralelo. Cada subproblema se procesa de manera concurrente en distintos hilos o procesos, y finalmente sus resultados se combinan para obtener la solución global.

Diagrama de componentes sobre el sistema:



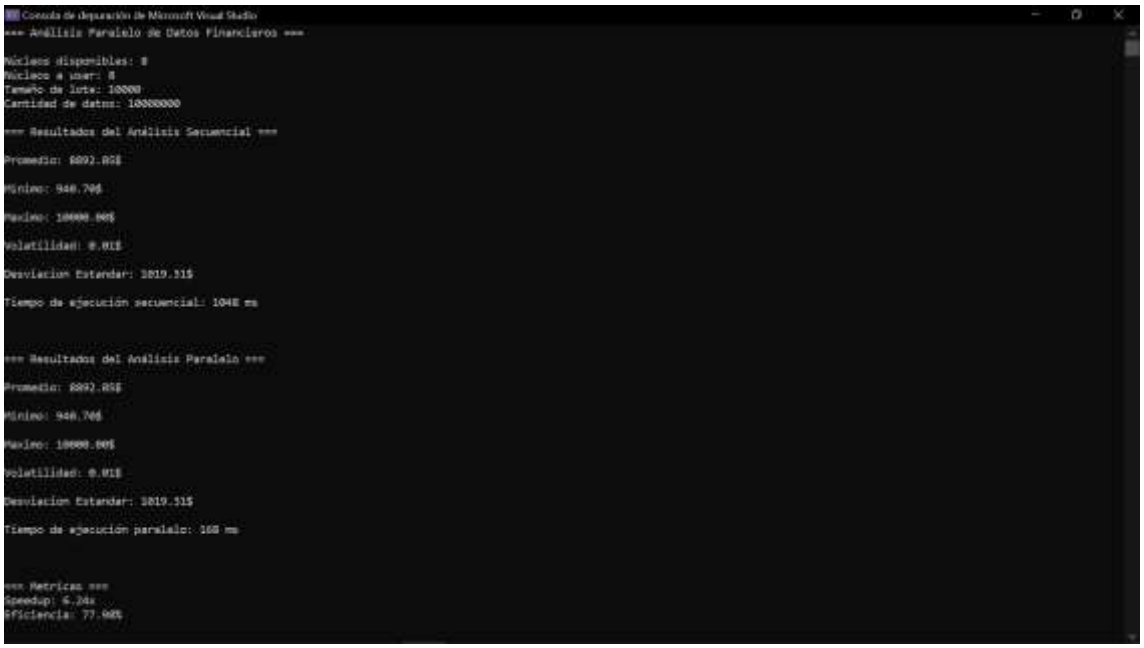
5. Implementación Técnica

El proyecto está organizado en módulos independientes que permiten separar responsabilidades y facilitar la mantenibilidad del código. El módulo DataLoader se encarga de generar o cargar los datos necesarios para el procesamiento. El módulo Statistics realiza los cálculos principales, como retornos promedio y volatilidad, utilizando algoritmos paralelos. El componente TimerUtil mide los tiempos de ejecución para comparar la versión secuencial y paralela.

El código central del proyecto está enfocado en el procesamiento paralelo de los datos numéricos. La función principal en Statistics recorre cada elemento del conjunto de datos para calcular métricas como promedios, variaciones y retornos porcentuales. La paralelización se logra mediante bucles paralelos, donde cada hilo procesa una parte del dataset de forma simultánea, reduciendo significativamente el tiempo de cálculo. El DataLoader utiliza funciones generadoras para construir arrays de tipo double[] que luego son entregados al módulo estadístico. Por su parte, TimerUtil utiliza métodos de alta precisión para medir el rendimiento, comparando la ejecución secuencial vs. paralela para evaluar la ganancia obtenida.

Durante el procesamiento paralelo se requiere asegurar que los hilos no generen conflictos al escribir o leer datos compartidos. Para esto se utilizan mecanismos de sincronización como locks, reducciones o barreras, dependiendo de la operación. En cálculos acumulativos, se aplican operaciones de reducción que permiten combinar los resultados parciales de cada hilo sin crear condiciones de carrera.

6. Evaluación de Desempeño



Característica	Análisis Secuencial	Análisis Paralelo
Promedio	$8892.85	$8892.85
Mínimo	$940.78	$940.78
Máximo	$10000.00	$10000.00
Volatilidad	0.01	0.01
Desviación Estándar	$1019.31	$1019.31
Tiempo de Ejecución	1048 ms	168 ms

Métrica	Valor
Speedup (Aceleración)	6.24x
Eficiencia	77.98%

Análisis de cuellos de botella o limitaciones

Cuello de Botella de Hardware: El límite físico de 8 núcleos disponibles restringe el Speedup máximo a aproximadamente 9x, independientemente de cuántos núcleos se soliciten.

Cuello de Botella de Overhead/Gestión: Solicitar más núcleos que los físicos disponibles introduce una sobrecarga de paralelismo significativa, reduciendo la eficiencia.

Limitación de Caché (Inversa): La baja eficiencia en la ejecución secuencial hace que el paralelismo no solo divida la carga de trabajo, sino que también optimice el acceso a la memoria.

7. Trabajo en Equipo

Juan Sebastian– SequentialAnalysis.cs
Fue responsable de la implementación del análisis secuencial del dataset. Su labor consistió en desarrollar los métodos que ejecutan los cálculos estadísticos sin paralelismo, funciones que sirven como referencia para comparar el rendimiento frente a la versión paralela.

Ryan Ortiz – ParallelAnalysis.cs, ResultCombiner.cs y DataLoader.cs
Encargada de los módulos vinculados directamente al procesamiento paralelo y al manejo inicial de los datos.

- En ParallelAnalysis.cs, implementó los métodos que dividen la carga de trabajo entre múltiples hilos o tareas, aplicando técnicas de paralelización eficientes.
- En ResultCombiner.cs, desarrolló la lógica para combinar los resultados parciales generados en paralelo, garantizando la consistencia y exactitud del resultado final.
- En DataLoader.cs, construyó el componente encargado de generar o cargar datasets simulados y convertirlos a estructuras manejables como double[].

Keison Cerda – Statistics.cs y TimerUtils.cs
Esta persona se ocupó tanto del núcleo matemático como de la medición del desempeño.

- En Statistics.cs, implementó las funciones que calculan métricas como retornos, promedios y volatilidades, asegurando precisión matemática y eficiencia.
- En TimerUtils.cs, desarrolló las herramientas para medir el tiempo de ejecución de cada implementación, permitiendo comparar objetivamente la versión secuencial con la paralela.

Hector Martinez – DataSplitter.cs
Responsable de diseñar e implementar el módulo que divide el dataset en segmentos más pequeños para ser procesados de manera simultánea. Su trabajo fue clave para mejorar el balance de carga entre hilos y optimizar el rendimiento en el análisis paralelo.

Para la coordinación del trabajo en equipo se utilizó GitHub como plataforma principal de gestión y control de versiones. A través del repositorio del proyecto, cada integrante pudo subir, actualizar y revisar el código desarrollado en su módulo correspondiente.

8. Conclusiones

El proyecto permitió comprender y aplicar conceptos fundamentales del procesamiento paralelo, incluyendo la división recursiva de tareas, el uso de Parallel.For, la sincronización de resultados parciales y la medición del rendimiento mediante speedup y eficiencia.Finalmente, el equipo adquirió experiencia en el uso de herramientas como GitHub, control de versiones y trabajo colaborativo mediante ramas y pull requests.

Uno de los principales desafíos fue diseñar una arquitectura que permitiera dividir grandes datasets en segmentos equilibrados para optimizar la ejecución paralela. También se presentaron dificultades relacionadas con la lectura del dataset en formato csv, evitando lecturas vacias o datos basura.

9. Referencias

- Microsoft Docs – *Task Parallel Library (TPL)*
- Microsoft Docs – *Parallel.For and Parallel LINQ (PLINQ)*
- StackOverflow

  https://es.stackoverflow.com/

10. Anexos

Manual del sistema:

Proyecto: Análisis Paralelo de Datos Financieros
Tecnología: C# (.NET 6 / .NET 7)
Interfaz: Consola

1. Requisitos previos

Antes de ejecutar el sistema, asegúrate de contar con:

.NET SDK 6.0 o superior
Descargar desde: https://dotnet.microsoft.com/en-us/download

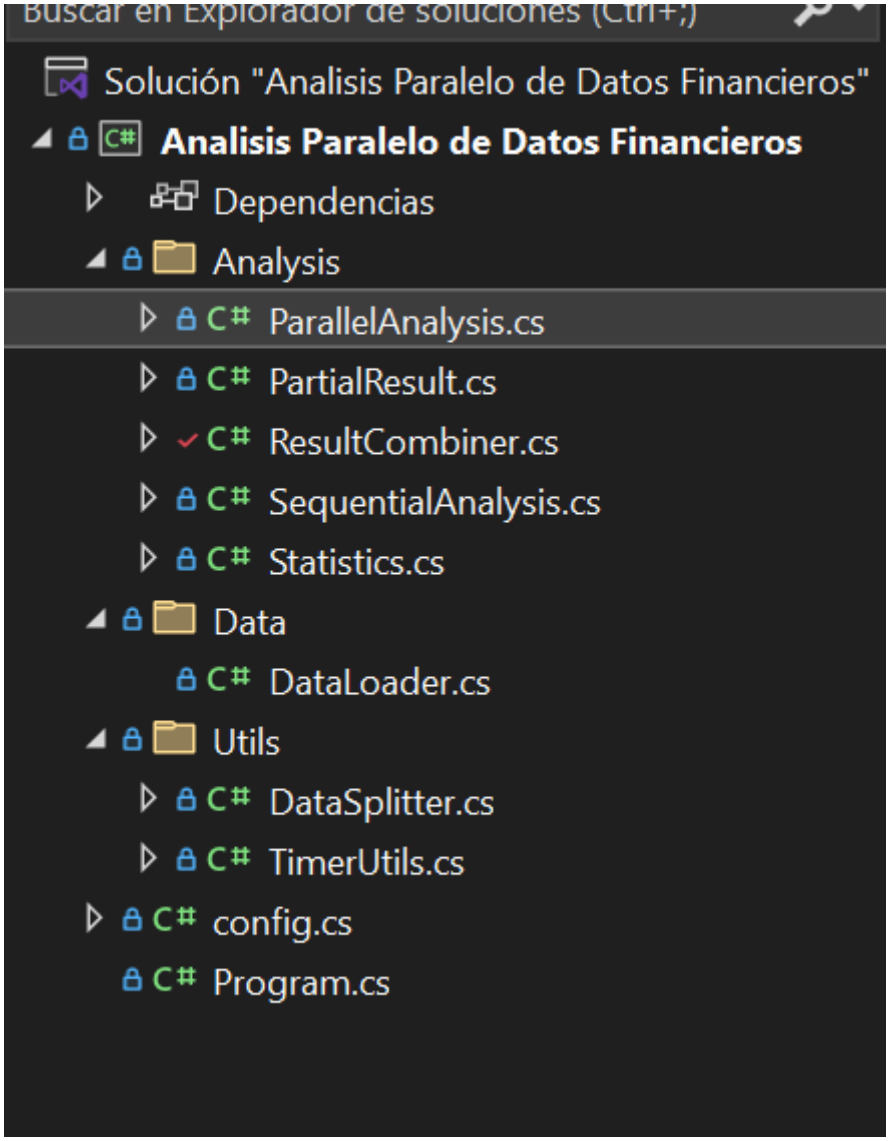
Editor o IDE recomendado:

- Visual Studio
- Visual Studio Code

Git para clonar el repositorio

2. Estructura del proyecto

El proyecto debe tener una estructura similar:



3. Ejecución del programa

Opción A: desde Visual Studio

1. Abrir el proyecto.
2. Seleccionar 'Analisis_Paralelo_de_Datos_Financieros' como proyecto de inicio.
3. Presionar **F5** o '*Start*'.
4. La consola mostrará un menú.

Opción B: desde consola

Ubícate en la carpeta del proyecto y ejecuta:

dotnet run

Resultados mostrados por pantalla

Tanto secuencial como paralelo imprimen:

```
=== Resultados del Análisis Secuencial ===

Promedio: 8887.45$

Minimo: 890.42$

Maximo: 10000.00$

Volatilidad: 0.01$

Desviacion Estandar: 1047.42$

Tiempo de ejecución secuencial: 1461 ms


=== Resultados del Análisis Paralelo ===

Promedio: 8887.45$

Minimo: 890.42$

Maximo: 10000.00$

Volatilidad: 0.01$

Desviacion Estandar: 1047.42$

Tiempo de ejecución paralelo: 153 ms


=== Metricas ===
Speedup: 9.55x
Eficiencia: 59.68%
```

Capturas adicionales, pruebas complementarias

```
=== Análisis Paralelo de Datos Financieros ===

Núcleos disponibles: 8
Núcleos a usar: 8
Tamaño de lote: 10000
Archivo CSV: cacao_10millones.csv

Cargando datos desde: cacao_10millones.csv
  Datos cargados: 0
  Datos omitidos: 10000000
Usando datos generados para pruebas...
Generando 1000000 datos de prueba...
Datos generados: 1000000

Total de datos a procesar: 1000000

--- Ejecutando Análisis Secuencial ---

--- Ejecutando Análisis Paralelo ---

=== Resultados del Análisis Secuencial ===
Promedio: 8897.54$
Mínimo: 693.65$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1071.01$
Tiempo de ejecución: 133 ms

=== Resultados del Análisis Paralelo ===
Promedio: 8897.54$
Mínimo: 693.65$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1071.01$
Tiempo de ejecución: 49 ms
Número de lotes procesados: 100

=== Métricas de Rendimiento ===
Speedup: 2.71x
Eficiencia: 33.93%
```

```
=== Análisis Paralelo de Datos Financieros ===

Núcleos disponibles: 8
Núcleos a usar: 6
Tamaño de lote: 10000
Archivo CSV: cacao_10millones.csv

Cargando datos desde: cacao_10millones.csv
  Datos cargados: 0
  Datos omitidos: 10000000
Usando datos generados para pruebas...
Generando 1000000 datos de prueba...
Datos generados: 1000000

Total de datos a procesar: 1000000

--- Ejecutando Análisis Secuencial ---

--- Ejecutando Análisis Paralelo ---

=== Resultados del Análisis Secuencial ===
Promedio: 8800.32$
Mínimo: 940.03$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1139.76$
Tiempo de ejecución: 130 ms

=== Resultados del Análisis Paralelo ===
Promedio: 8800.32$
Mínimo: 940.03$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1139.76$
Tiempo de ejecución: 39 ms
Número de lotes procesados: 100

=== Métricas de Rendimiento ===
Speedup: 3.33x
Eficiencia: 55.56%
```

```
=== Análisis Paralelo de Datos Financieros ==

Núcleos disponibles: 8
Núcleos a usar: 4
Tamaño de lote: 10000
Archivo CSV: cacao_10millones.csv

Cargando datos desde: cacao_10millones.csv
  Datos cargados: 0
  Datos omitidos: 10000000
Usando datos generados para pruebas...
Generando 1000000 datos de prueba...
Datos generados: 1000000

Total de datos a procesar: 1000000

--- Ejecutando Análisis Secuencial ---

--- Ejecutando Análisis Paralelo ---

=== Resultados del Análisis Secuencial ===
Promedio: 8812.17$
Mínimo: 866.21$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1189.46$
Tiempo de ejecución: 130 ms

=== Resultados del Análisis Paralelo ===
Promedio: 8812.17$
Mínimo: 866.21$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1189.46$
Tiempo de ejecución: 27 ms
Número de lotes procesados: 100

=== Métricas de Rendimiento ===
Speedup: 4.81x
Eficiencia: 120.37%
```

```
=== Análisis Paralelo de Datos Financieros

Núcleos disponibles: 8
Núcleos a usar: 2
Tamaño de lote: 10000
Archivo CSV: cacao_10millones.csv

Cargando datos desde: cacao_10millones.csv
  Datos cargados: 0
  Datos omitidos: 10000000
Usando datos generados para pruebas...
Generando 1000000 datos de prueba...
Datos generados: 1000000

Total de datos a procesar: 1000000

--- Ejecutando Análisis Secuencial ---

--- Ejecutando Análisis Paralelo ---

=== Resultados del Análisis Secuencial ===
Promedio: 8890.14$
Mínimo: 937.87$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1046.71$
Tiempo de ejecución: 134 ms

=== Resultados del Análisis Paralelo ===
Promedio: 8890.14$
Mínimo: 937.87$
Máximo: 10000.00$
Volatilidad: 0.0112
Desviación Estándar: 1046.71$
Tiempo de ejecución: 33 ms
Número de lotes procesados: 100

=== Métricas de Rendimiento ===
Speedup: 4.06x
Eficiencia: 203.03%
```

Enlace al github

https://github.com/Keison28/Analisis-Paralelo-de-Datos-Financieros