Análisis Paralelo de Sentimiento en Reseñas de Productos

Jesús Israel Velez Gonzalez  
*Ingenieria de Sistemas*  
*Universidad de Antioquia*  
Medellín, Colombia  
israel.velez@udea.edu.co

Juan Esteban Ortiz Velasquez  
*Ingenieria de Sistemas*  
*Universidad de Antioquia*  
Medellín, Colombia  
juan.ortiz26@udea.edu.co

*Abstract*— This project presents the development of a program in C for analyzing user reviews, focusing on performance metrics such as CPU usage and memory consumption. The main objective is to process a large volume of reviews, classify their polarity (positive, negative, or neutral), and analyze the frequency of associated keywords. The program leverages multithreading (pthread) to optimize performance by processing reviews in parallel, ensuring efficient use of multicore processors. Mutexes are employed to manage shared resources and maintain data consistency during concurrent operations. The methodology involves reading data from a file, processing reviews in parallel threads, and generating relevant statistics, including keyword counts, review classifications, and system performance metrics. Performance is measured using functions like clock and getrusage to evaluate execution time and peak memory usage. Results demonstrate significant improvements in execution time compared to sequential implementations, confirming the advantages of parallel processing in handling text-based data at scale. This work contributes to understanding efficient data processing in parallel applications and highlights the potential of multithreading for enhancing performance in data-intensive tasks. Future work may explore more complex sentiment analysis models and further optimization of keyword search operations to broaden the application scope.

Keywords— Procesamiento paralelo, multithreading, análisis de reseñas, pthread, rendimiento computacional

# Introducción

El proyecto se centra en la construcción de un programa en C para analizar reseñas de usuarios. Este programa mide el rendimiento de procesamiento mediante el uso de métricas de uso de CPU y memoria. El objetivo es evaluar la cantidad de reseñas procesadas, identificar su polaridad (positiva, negativa o neutral), y analizar la frecuencia de palabras clave asociadas.

La motivación detrás del proyecto incluye:

*1. Ampliar el conocimiento sobre el manejo eficiente de datos en aplicaciones paralelas.*

*2. Experimentar con técnicas de multithreading para optimizar el rendimiento.*

*3. Generar estadísticas útiles basadas en datos textuales, algo aplicable en múltiples industrias.*

La metodología incluye la lectura de datos de un archivo, procesamiento paralelo de reseñas con hilos (pthread), y la generación de estadísticas relevantes como conteo de palabras clave, clasificaciones de reseñas, y métricas de rendimiento del sistema. Se mide el tiempo de CPU y la memoria usada para evaluar la eficiencia del programa.

# Marco teórico

Procesamiento paralelo: El programa utiliza pthread para procesar varias reseñas simultáneamente, lo que reduce el tiempo total de ejecución. El concepto clave es dividir tareas independientes entre múltiples hilos, aprovechando la capacidad multicore de los procesadores modernos.

Manejo de recursos compartidos: Se utilizan mutexes (pthread\_mutex\_t) para garantizar la consistencia en el acceso a las estadísticas globales y estructuras de datos, evitando condiciones de carrera.

Procesamiento de texto:

*- Se analiza la frecuencia de palabras clave (positivas y negativas) usando funciones de búsqueda como strstr.*

*- Las reseñas se clasifican con base en palabras positivas y negativas, reflejando conceptos básicos de análisis de sentimientos.*

Medición de rendimiento: El programa emplea funciones como clock y getrusage para medir métricas de tiempo de CPU y uso máximo de memoria. Esto es esencial para evaluar la eficiencia del sistema en aplicaciones intensivas en datos.

# Metodología

El desarrollo del proyecto siguió los pasos siguientes:

*1. Diseño inicial: Se definieron las estructuras de datos (estadísticas y palabras clave) y las funciones para el manejo de reseñas.*

*2. Lectura y preprocesamiento de datos: Se cargan las reseñas desde un archivo de texto, expandiendo dinámicamente el almacenamiento a medida que crece el número de entradas.*

*3. Procesamiento concurrente: Cada reseña se asigna a un hilo independiente, donde se analizan palabras clave y se actualizan las estadísticas.*

*4. Sincronización: Se utilizaron mutexes para proteger el acceso concurrente a las estadísticas y las estructuras de datos.*

*5. Validación y medición de resultados: Se midieron las métricas de rendimiento y se generaron estadísticas finales.*

# RESULTADOS

Los resultados obtenidos muestran:

*1. Estadísticas de reseñas:*

*-Número total de reseñas procesadas.*

*-Porcentaje de reseñas positivas, negativas y neutrales.*

*-Conteo total de palabras clave positivas y negativas.*

*2. Frecuencia de palabras clave:*

*-Se imprimen las palabras clave más comunes y su frecuencia.*

*3. Rendimiento:*

*-Tiempo de CPU utilizado: valor calculado en segundos.*

*-Memoria máxima utilizada: valor calculado en KB.*

El procesamiento paralelo mostró una mejora significativa en el tiempo de ejecución en comparación con una implementación secuencial.

# Implementación

Software:

El programa fue implementado en C y tiene las siguientes características:

1. Estructuras de datos:

- ReviewStats: Almacena estadísticas generales como el número total de reseñas, las clasificadas como positivas, negativas, neutrales y los conteos de palabras clave.

- WordCount: Guarda palabras clave y su frecuencia.

2. Multithreading:

- Uso de pthread\_create y pthread\_join para procesar reseñas en paralelo.

- Mutexes para garantizar la seguridad en el acceso a variables globales.

3. Funciones clave:

- update\_review\_stats: Clasifica una reseña según conteos de palabras clave.

- update\_word\_frequency: Rastrea la frecuencia de palabras clave en las reseñas.

- measure\_performance: Mide el tiempo de ejecución y memoria máxima utilizada.

4. Entrada y salida:

- Entrada: Archivo de texto (reviews\_output.txt) con una reseña por línea.

- Salida: Estadísticas consolidadas impresas en la consola.

Hardware:

El proyecto no incluye componentes de hardware. Sin embargo, su diseño es óptimo para entornos con procesadores multicore.

Protocolo de experimentación

El protocolo experimental considera los siguientes aspectos:

1. Seguridad y consistencia:

- Se implementaron mutexes para evitar condiciones de carrera al actualizar estructuras de datos compartidas.

- Las reseñas se procesan independientemente para minimizar interdependencias.

2. Herramientas utilizadas:

- Medición de rendimiento con clock para tiempos de CPU y getrusage para el uso de memoria.

- pthread para procesamiento paralelo eficiente.

- Funciones de C estándar (strtok, strstr, etc.) para el análisis de texto.

3. Validación:

- Verificación manual de resultados con un subconjunto de datos.

- Comparación de métricas como tiempo total de procesamiento y uso de memoria.

# Concluciones

1. El uso de multithreading en el procesamiento de reseñas permite una mejora notable en el rendimiento, especialmente en sistemas con múltiples núcleos.

2. El manejo adecuado de recursos compartidos mediante mutexes garantiza la consistencia en los resultados.

3. La implementación es extensible, pudiendo adaptarse para incluir más categorías de palabras clave o realizar análisis más complejos, como modelos de clasificación.

4. Las métricas de rendimiento indican que el diseño del programa es eficiente para grandes volúmenes de datos, aunque se podrían optimizar las operaciones de búsqueda de palabras clave.

##### Referencias

[1] B. Lewis and D. J. Berg, Multithreaded Programming with Pthreads. Prentice Hall, 1998.

[2] D. Bovet and M. Cesati, Understanding the Linux Kernel, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2005.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[4] R. M. Stallman, R. P. McGrath, and A. Oram, GNU Make: A Program for Directed Compilation. Sebastopol, CA, USA: O'Reilly Media, 2002.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in Communications of the ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[6] P. S. Pacheco, An Introduction to Parallel Programming. Burlington, MA, USA: Morgan Kaufmann, 2011.

[7] J. B. Dennis and J. Van Horn, "Programming semantics for multiprogrammed computations," Communications of the ACM, vol. 9, no. 3, pp. 143–155, Mar. 1966.

[8] J. C. Liu, "Efficient dynamic memory allocation strategies for parallel programs," in IEEE Transactions on Computers, vol. 50, no. 3, pp. 214–228, Mar. 2001.

[9] S. Basu, "Sentiment analysis and opinion mining: A survey," in International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 5, pp. 432–437, May 2015.

[10] ISO/IEC 14882:2017, Programming Languages – C++, International Organization for Standardization, 2017.