Proyecto Final
Curso de Sistemas Operativos y Laboratorio

Comparación de Algoritmos: Implementación Secuencial en Python vs. Paralelización en Spark

Efraín García Valencia
Samuel Acevedo Bustamante
Nicolas Carmona Cardona

Resumen
Este proyecto propone una comparación entre un algoritmo tradicional implementado en Python y su versión paralelizada utilizando Apache Spark. A través de este análisis, se explorarán las ventajas y desventajas de cada enfoque en términos de eficiencia, tiempo de ejecución y uso de recursos del sistema. El proyecto se contextualiza en la importancia creciente de la paralelización en el procesamiento de datos masivos y su relación con los sistemas operativos modernos.

Introducción
En la actualidad, el crecimiento exponencial de datos requiere algoritmos eficientes que puedan procesarlos de manera efectiva. La computación paralela se presenta como una solución ante la limitación de procesamiento en entornos secuenciales.
La capacidad de paralelizar algoritmos se ha vuelto esencial en la era de Big Data y Machine Learning, donde los sistemas operativos deben gestionar recursos de manera eficiente para maximizar el rendimiento.

Marco Teórico
Se abordarán conceptos de programación concurrente y paralela, gestión de procesos e hilos en sistemas operativos, así como la arquitectura de Spark y su modelo de programación basado en Resilient Distributed Datasets (RDD).
Los sistemas operativos son responsables de la gestión de recursos en entornos paralelos, por lo que entender su funcionamiento es clave para optimizar algoritmos y entender cómo Spark interactúa con el sistema subyacente.

Objetivos

Objetivo principal: Implementar y comparar un algoritmo en Python y su versión paralelizada en Spark, evaluando rendimiento y eficiencia.

Objetivos específicos:

1. Seleccionar un algoritmo representativo (ej. ordenamiento, búsqueda, procesamiento de datos).
2. Desarrollar la implementación secuencial en Python.
3. Desarrollar la implementación paralelizada en Spark.
4. Realizar pruebas de rendimiento en diferentes volúmenes de datos.
5. Analizar y presentar los resultados obtenidos.

Metodología

Herramientas: Python, Apache Spark, Jupyter Notebook, herramientas de análisis de rendimiento (como Apache Bench o timeit).

Actividades necesarias:

1. Investigación sobre el algoritmo seleccionado.
2. Desarrollo de la implementación en Python.
3. Instalación y configuración de Spark.
4. Desarrollo de la implementación en Spark.
5. Ejecución de pruebas de rendimiento con diferentes tamaños de datos.
6. Análisis y redacción de los resultados.

Diseño de Experimentos

Se realizarán pruebas de rendimiento comparando el tiempo de ejecución y el uso de recursos (CPU y memoria) entre ambas implementaciones. Se analizarán métricas como el tiempo de respuesta y la escalabilidad al aumentar el tamaño de los datos.

Cronograma

Actividades	*Semana 1 (20-27 Oct)*	*Semana 2 (28 Oct-3 Nov)*	*Semana 3 (4-10 Nov)*	*Semana 4 (11-17 Nov)*	*Semana 5 (18-20 Nov)*
Investigación sobre el algoritmo	✔				
Implementación secuencial en Python		✔			
Instalación y configuración de Spark		✔			
Implementación en Spark			✔		
Pruebas de rendimiento en Python			✔		
Pruebas de rendimiento en Spark				✔	
Análisis de resultados				✔	
Redacción y presentación de resultados				✔	✔

Referencias

1. Apache Spark Documentation. (n.d.). Retrieved from spark.apache.org
2. "Operating System Concepts" by Silberschatz, Galvin, and Gagne.
3. "Python for Data Analysis" by Wes McKinney.
4. Artículos y recursos adicionales sobre programación paralela y algoritmos.