Proyecto Final

Curso de Sistemas Operativos y Laboratorio

Hilos y procesos múltiples para acelerar procesos de búsqueda

Miembros del equipo[1]

– Multithreading –
Daniel Lujan Agudelo
Guillermo Sanchez Cubides

– Multiprocessing –
Jose David Gomez Muñeton
Juan Pablo Arango Gaviria

[1] Ambos equipos realizaremos el experimento descrito en este documento. Sin embargo, un equipo se encargará de la ejecución y comparación del paralelismo con múltiples hilos, y el otro con múltiples procesos

Índice

Resumen

El proyecto tiene como objetivo comparar el rendimiento de tres técnicas de búsqueda de registros en grandes volúmenes de datos almacenados en archivos: búsqueda secuencial, paralela con *multithreading*, y paralela con *multiprocessing*. Dada la creciente necesidad de gestionar eficientemente grandes cantidades de información, especialmente en sectores como ciencia de datos e inteligencia artificial, este proyecto busca identificar qué enfoque es más eficiente según el entorno y la carga de trabajo.

La **metodología** incluye pasos como la implementación de las tres técnicas en Python, utilizando herramientas para *mulithreading* y *multiprocessing*. El proyecto se desarrollará en fases: implementación de las técnicas, pruebas controladas, recolección de datos y análisis estadístico de los resultados.

El proyecto busca no solo medir el tiempo de ejecución de cada técnica, sino también proporcionar un análisis más amplio sobre el uso de CPU y la eficiencia en la búsqueda de grandes volúmenes de datos.

1. Introducción

En la actualidad, el manejo eficiente de grandes volúmenes de datos es un desafío crítico para el rendimiento de diversas aplicaciones. Con el crecimiento exponencial en la cantidad de información generada y almacenada en sistemas distribuidos, la búsqueda rápida y precisa de registros específicos se ha convertido en una necesidad imperiosa para sectores como la ciencia de datos, el comercio electrónico y la inteligencia artificial. Tradicionalmente, las técnicas de búsqueda secuencial han sido suficientes para conjuntos de datos pequeños, pero a medida que estos crecen, dichas técnicas se vuelven ineficientes y poco escalables. Es aquí donde el paralelismo, tanto con hilos (multithreading) como con procesos (multiprocessing), ofrece un potencial considerable para mejorar el rendimiento.

2. Antecedentes o marco teórico

La eficiencia en la ejecución de programas es fundamental para cualquier sistema o aplicación moderna y un enfoque para lograr mejoras en el rendimiento es la *paralelización*. Durante el proyecto se explorarán dos técnicas clave de paralelismo: *multiprocessing* y *multithreading*.

Primero se quiere comparar cada una de las técnicas con la ejecución secuencial tradicional para identificar las situaciones en las que cada técnica ofrece ventajas.

Hay que aclarar que el rendimiento de cada técnica depende en gran medida del hardware disponible, en particular del número de núcleos y la capacidad de la CPU para gestionar múltiples hilos (hyper-threading).

Multiprocessing:

Según [1], es una técnica que permite ejecutar múltiples procesos en paralelo, cada uno con su propio espacio de memoria y recursos. Esta técnica se aprovecha de CPU multi-core, donde cada núcleo puede ejecutar un proceso independiente. La independencia entre procesos evita problemas relacionados con la sincronización de memoria compartida.

Multithreading:

Cómo se describe en [2], *multithreading* es la técnica que permite la ejecución de múltiples hilos dentro de un mismo proceso, los hilos comparten el mismo espacio de memoria, lo que hace más rápida su creación y facilita la comunicación entre ellos. Sin embargo, esto también introduce el riesgo de condiciones de carrera y *deadlocks*, lo que requiere mecanismos de sincronización como semáforos y bloqueos.

Ambas técnicas están altamente relacionadas con el curso, por ejemplo, la creación, planificación y terminación de procesos es fundamental para implementar ***multiprocessing***, las dos son ejemplos de concurrencia, pero aplicadas en diferentes niveles (procesos vs. hilos) e incluso los problemas de concurrencia en ***multithreading*** requieren la implementación de mecanismos como semaforización para evitar condiciones de carrera.

3. Objetivos

Objetivo general

Comparar el rendimiento de tres técnicas de búsqueda de registros en grandes volúmenes de datos almacenados en archivos en disco: búsqueda secuencial, búsqueda paralela utilizando multithreading, y búsqueda paralela utilizando multiprocessing.

Objetivos específicos

- Implementar programas que permitan realizar búsqueda de forma secuencial y paralela (con múltiples hilos y procesos).
- Ejecutar los programas implementados en un entorno controlado.
- Realizar mediciones de las Métricas necesarias para comparar el rendimiento.
- Analizar los resultados experimentales obtenidos.

4. Metodología

Para implementar la solución propuesta, se utilizarán herramientas clave del ecosistema de Python, como las bibliotecas *threading* y *multiprocessing* para la implementación del paralelismo, junto con *time* o *perf_counter* (según [3], esta librería sería preferible sobre *time*) para la medición precisa de los tiempos de ejecución. Además, herramientas de monitoreo como *psutil* permitirán analizar el uso de CPU y memoria durante las pruebas. El desarrollo del proyecto seguirá un enfoque estructurado, comenzando por la definición de los requisitos, incluyendo el tamaño de los archivos y los criterios de búsqueda.

Luego, se procederá con la implementación de las tres técnicas de búsqueda: secuencial, *multithreading* y *multiprocessing*. Posteriormente, se llevará a cabo la configuración de los experimentos, incluyendo la creación de un entorno de pruebas controlado. Una vez completada la implementación, se realizarán pruebas preliminares para verificar el correcto funcionamiento de las tres soluciones, seguidas por pruebas formales de rendimiento donde se recolectarán los datos de tiempo de ejecución y uso de recursos.

Finalmente, los resultados se someterán a un análisis estadístico para extraer conclusiones y cumplir con los objetivos del proyecto.

5. Diseño de Experimentos

5.1. Breve descripción

El experimento consistirá en realizar operaciones de búsqueda sobre un conjunto de datos (archivos en disco), aplicando las técnicas ya mencionadas y evaluando el rendimiento de cada una.

5.2. Hipótesis

- **H0 (Hipótesis nula):** No hay diferencias significativas en el tiempo de búsqueda entre las tres técnicas de búsqueda: secuencial, *multithreading* y *multiprocessing*.
- **H1 (Hipótesis alternativa):**Existen diferencias significativas en el tiempo de búsqueda entre las técnicas de búsqueda: secuencial, *multithreading* y *multiprocessing*.

5.3. Predicciones

- Se espera que la búsqueda paralela con ***multiprocessing*** sea más rápida que con ***multithreading*** debido a la capacidad de aprovechar múltiples núcleos de CPU y evitar la limitación del *Global Interpreter Lock* (GIL) en Python.
- La búsqueda secuencial será la más lenta, ya que no aprovecha el paralelismo ni el procesamiento en múltiples hilos o procesos.
- En archivos pequeños o en situaciones con sobrecarga alta de administración de procesos o hilos, el rendimiento de ***multiprocessing*** o ***multithreading*** podría no ser significativamente superior.

5.4. Metodología (experimental)

El diseño es un experimento comparativo donde se ejecutarán múltiples pruebas de búsqueda en un conjunto de datos grande utilizando las tres técnicas. Se medirán los tiempos de ejecución para cada una de ellas.

5.4.1. Factores de Tratamiento

- **Técnica de búsqueda**
    - Niveles: Secuencial, Paralelo (multithreading), Paralelo (multiprocessing)

5.4.2. Factores de Bloqueo

Se utilizarán los siguientes factores de bloqueo para controlar la variabilidad externa:

- **Tamaño de los archivos**: Se evaluarán archivos de diferentes tamaños (100 MB, 500 MB, 1 GB).
- **Número de archivos**: Se variará la cantidad de archivos en los que se realiza la búsqueda (5, 10, 50 archivos).
- **Especificaciones de hardware**: Las pruebas se realizarán en la misma máquina, además se configurará para que el número de hilos sea igual al número de cores, y así evitar que diferencias en el hardware afecten los resultados.

5.4.3. Procedimiento

- Se creará un conjunto de datos grande con varios archivos en disco.
- Se implementarán tres versiones del script de búsqueda en Python:
    - **Búsqueda secuencial:** Leer y buscar en cada archivo de manera secuencial.
    - **Búsqueda paralela con *multithreading*:** Utilizar múltiples hilos (*threads*) para buscar en diferentes archivos simultáneamente.

    - **Búsqueda paralela con *multiprocessing*:** Utilizar múltiples procesos para aprovechar múltiples núcleos de CPU y buscar en diferentes archivos simultáneamente.

- Cada técnica se ejecutará repetidamente (al menos 10 veces) para reducir la variabilidad en los tiempos de ejecución.
- Los tiempos de búsqueda se medirán y registrarán para cada ejecución.

5.4.4. Control de Variabilidad

- Cada técnica se ejecutará el mismo número de veces para cada condición.
- Todas las pruebas se realizarán bajo las mismas condiciones de hardware y software.
- La creación y sincronización de hilos o procesos será realizada de manera uniforme para evitar que el *overhead* de administración afecte injustamente los resultados.

5.4.5. Factores Controlados

- **Algoritmo de búsqueda**: El mismo algoritmo de búsqueda se utilizará en las tres técnicas (por ejemplo, búsqueda lineal) para garantizar que las diferencias en el tiempo de ejecución sean debidas al enfoque (secuencial, hilos o procesos).
- **Lenguaje**: Python será el lenguaje utilizado en las tres implementaciones, con las mismas bibliotecas para manejo de archivos y paralelismo (*threading* y *multiprocessing*).

5.5. Métricas

La métrica principal será el **tiempo de búsqueda** en segundos. Además, se evaluarán las siguientes métricas secundarias:

- **Uso de CPU**: Para medir la eficiencia en el uso de los núcleos de CPU en cada técnica.

- ***Overhead* de administración**: Se medirá el tiempo dedicado a la creación y sincronización de hilos o procesos, para identificar si introduce una penalización significativa en términos de rendimiento.

- **Escalabilidad**: Cómo varía el rendimiento cuando el tamaño del conjunto de datos o el número de archivos aumenta.

5.6. Análisis de Resultados

Para evaluar las diferencias entre las técnicas, se utilizará un análisis estadístico. Dependiendo de la distribución de los datos, se aplicará una prueba ANOVA o Kruskal-Wallis para determinar si existen diferencias significativas entre los tiempos medios de las tres técnicas.

Si el análisis indica que existe una diferencia relevante, se realizará una prueba Tukey para identificar qué técnicas presentan diferencias relevantes entre sí.

5.7. Limitaciones

- El rendimiento de las técnicas paralelas puede depender fuertemente de las especificaciones del hardware, por lo que los resultados pueden variar en otros entornos.

- **Multiprocessing** introduce una mayor sobrecarga debido a la creación de procesos, lo que puede afectar los resultados en archivos pequeños o cuando la cantidad de archivos es baja.

- El *Global Interpreter Lock* (GIL) en Python podría limitar la eficiencia de ***multithreading***, especialmente en tareas intensivas de CPU.

6. Cronograma

Paso	Actividad	Fecha límite
Formación del equipo y selección del tema	- Formar equipo de tres integrantes. - Seleccionar tema del proyecto - Registrar miembros y tema en archivo compartido	3/10/2024
Investigación y propuesta de proyecto	- Realizar investigación preliminar sobre el tema - Elaborar propuesta en un informe	17/10/2024
Presentación de la Propuesta	Presentar la propuesta en clase	29/10/2024
Avances del proyecto	- Preparar presentación sobre los avances - Mostrar porcentaje de ejecución, retos superados y por superar - Planificar estrategias en caso de retraso	7/11/2024
Presentación de Avances	Exponer los avances en clase	7/11/2024
Avance final del proyecto	- Organizar el repositorio de código con documentación completa y archivo README - Elaborar reporte técnico	25/11/2024
Envío del repositorio e informe técnico	- Subir el repositorio y el informe técnico a Moodle	25/11/2024
Presentación final	Exponer el proyecto final con demo funcional en clase	25/11/2024

7. Referencias

[1] What is multiprocessing? Disponible en:
https://www.techtarget.com/searchdatacenter/definition/multiprocessing

[2] Entendiendo los procesos, hilos y multihilos. Disponible en:
https://medium.com/@diego.coder/entendiendo-los-procesos-hilos-y-multihilos-9423f6e40ca7

[3] Python time.time() vs time.perf_counter(). Disponible en:
https://superfastpython.com/time-time-vs-time-perf_counter/