Laboratorio: Concurrencia y paralelismo en Python

Objetivos

El objetivo de esta actividad es probar los distintos mecanismos que existen en Python para simular procesos de acceso en simultáneo o concurrente, así como para optimizar un código que requiera de cálculos extensos. De esta manera se podrá comparar el rendimiento.

Se aplicarán los distintos métodos tanto a la ejecución de algoritmos como al procesamiento de datos.

Para el desarrollo de esta actividad es necesario haber estudiado previamente las funciones de creación de hilos y procesos en Python, sus mecanismos de sincronización y comunicación, y las consideraciones para realizar la paralelización de manera adecuada.

Descripción de la actividad

La actividad consta de dos tareas:

* Simulación de un problema de sincronización de hilos. Se trabajará en la creación de una simulación que involucre hilos y donde se explorará el manejo de la sincronización entre ellos.
* Ejecución de una función sobre un conjunto de datos extenso de tres formas distintas: secuencial, con hilos y con Pool de procesos. Posteriormente se analizarán los rendimientos obtenidos en cada caso.

Tarea 1. Simulación de sincronización entre hilos

Se va a tratar el problema clásico del barbero durmiente, que simula lo que ocurre con algunos procesos gestionados por el sistema operativo. El objetivo es simular el funcionamiento de una barbería.

La barbería funciona de la siguiente manera: cuando no hay clientes, el barbero duerme. Al llegar un cliente, si el barbero está durmiendo, este pasa directamente al sillón para que le corten el pelo. Sin embargo, si el barbero está ocupado, el cliente se sienta en la sala de espera hasta que pueda ser atendido. La sala de espera cuenta con tres sillas, lo que significa que puede haber hasta tres clientes esperando como máximo. Si llega un cliente adicional y no hay espacio en la sala, deberá esperar fuera de la puerta. Opcionalmente, se puede implementar un protocolo para que el cliente que no logre entrar, debido a que la sala está llena, no espere o se vaya y vuelva a intentarlo después de un tiempo. No es necesario implementar este último aspecto. Una vez que los clientes reciben su corte de pelo, se marchan.

El objetivo de esta tarea es resolver el problema mediante programación concurrente, utilizando hilos de threading. Para ello se proporciona el código implementado en el *notebook* BarberoDurmienteLaboratorio.ipynb, que se adjunta en la actividad.

En esta tarea se plantea:

* Definir e inicializar semáforos, bloqueos, condiciones y variables compartidas en la implementación proporcionada para completar la simulación concurrente.
* Explicar con comentarios en el *notebook* las decisiones tomadas en la implementación.
* Probar el ejemplo con 1 barbero y 6 clientes.
* Implementar los protocolos antes y después de cortar, que realizará el barbero.
* Implementar los protocolos para el ingreso a la barbería, el protocolo de espera en la sala (si es el caso), el protocolo de salida de la sala y el ingreso al sillón para el corte de pelo.

El objetivo es lograr una simulación adecuada y funcional del problema del barbero durmiente, teniendo en cuenta los aspectos concurrentes y las interacciones entre los hilos y procesos involucrados.

Tarea 2. Procesamiento de un corpus de texto

Una de las aplicaciones del paralelismo, en el que se consigue un rendimiento considerable, es en el procesamiento de lenguaje natural, en el que se deben realizar distintas tareas que gana intensidad de cómputo. En esta tarea se va a cargar uno de los corpus que proporciona NLTK, concretamente el corpus *Brown*, sobre un *DataFrame*.

En el corpus se realizará un procesamiento con varias transformaciones en las palabras del texto y, además, contará el número de palabras de cada frase y se añadirá como columna al *DataFrame*.

Se proporciona el código base para poder realizar la implementación. Este código se debe de ejecutar sobre el intérprete, o bien usar algún entorno de desarrollo como Spyder, pero no se podrá ejecutar sobre un *notebook*.

En el *notebook* procesamientocorpus.ipynb, se explican las distintas partes del código dejando los huecos en los que se debe añadir la implementación pedida para esta tarea.

El código será ejecutado en secuencial y por un Pool de procesos. Se deben comparar los rendimientos.

Por tanto, en esta tarea se pide:

* Implementación de las instrucciones de transformación.
* Implementar la función que cuenta palabras e incorporar la columna al *DataFrame*.
* Implementación de la ejecución en secuencial.
* Implementación de la ejecución paralela mediante un pool de procesos.
* Mostrar los resultados de algunas de las frases del corpus en la ejecución secuencial y paralela.
* Mostrar los tiempos de ejecución.
* Comparar y analizar los rendimientos.

El objetivo es observar cómo afecta el paralelismo en el rendimiento del procesamiento de lenguaje natural en este contexto específico, y si la utilización de un Pool de procesos mejora la eficiencia en comparación con la ejecución secuencial.

Todo el código debe estar comentado y se debe explicar las decisiones tomadas.

Para esta tarea se deben seguir las siguientes indicaciones:

* Adjuntar los archivos o *notebooks* que se describen a continuación. Aunque no se pueda ejecutar el código paralelo, es importante proporcionarlos para su revisión.
* El código Python se debe entregar en un archivo .py para que pueda ser ejecutado por el docente, de forma que pueda revisar su contenido de manera adecuada.

Forma de entrega

En este trabajo se deberá entregar:

* *Notebook* de la tarea uno, debidamente comentado y explicado.
* *Notebook* de la tarea 2 o archivo de texto editable, debidamente comentado y explicado. Se podrá incluir y ejecutar la parte secuencial. Mostrando resultados y análisis de los rendimientos.
* Documento en el que se explique la parte paralela, mostrando resultados. Se puede incluir en el *notebook* anterior, aunque no se pueda ver la ejecución, o en un archivo de texto editable, no PDF.
* Archivo de código Python (.py) con todo el código de las dos partes para que pueda ser ejecutado.

Rúbrica

|  |  |  |  |
| --- | --- | --- | --- |
| Concurrencia y paralelismo en Python | Descripción | Puntuación máxima  (10) | Peso  100% |
| Tarea 1 (definiciones e inicializaciones) | * Se definen e inicializan todos los mecanismos de sincronización. * Se definen e inicializan todos los hilos. * Se realiza el análisis de la ejecución. | 1 | 10% |
| Tarea 1 (implementación del barbero) | * Se implementa el protocolo inicial del barbero, antes de atender a un cliente. * Se implementa el protocolo del barbero tras atender a un cliente. * Se explican todas las implementaciones y las decisiones tomadas. | 2 | 20% |
| Tarea 1 (implementación de los clientes) | * Se implementa el protocolo inicial del cliente cuando llega a la barbería. * Se implementa el mecanismo para despertar al barbero. * Se implementa el protocolo del cliente cuando se le da acceso al sillón de corte. * Se explican todas las implementaciones y las decisiones tomadas. | 2 | 20% |
| Tarea 2 (funciones) | * Se implementan las funciones pedidas y la función principal. * Se explican todas las implementaciones y las decisiones tomadas. | 1 | 10% |
| Tarea 2 (ejecución seuencial) | * Se implementa la ejecución secuencial. * Se muestran resultados parciales. * Se obtiene el tiempo de ejecución. * Se explican todas las implementaciones y las decisiones tomadas. | 1 | 10% |
| Tarea 2 (ejecución paralela) | * Se implementa la ejecución paralela, con el pool de procesos. * Se muestran resultados parciales. * Se obtiene el tiempo de ejecución. * Se explican todas las implementaciones y las decisiones tomadas. | 2 | 20% |
| Laboratorio (general) | * Se hace el análisis del rendimiento y se extraen conclusiones. | 1 | 10% |
|  |  | **10** | **100%** |