Cisneros Aguilar Sara Kenia

Computación Concurrente

1. ¿Qué es?

Un thread o *hilo* es un "semi-proceso"que ejecuta una porción de código, por lo que normamlmente comparte su memoria con otros threads, sin embargo a cada uno se le asigna su espacio de memoria, dentro del desarrollo de la aplicación, se definen diversas tareas para que se ejecuten a la vez. Todas estas tareas forman parte de un mismo proceso. Un grupo de threads en ejecución es un conjunto de "hilos en ejecuciónçorriendo en el mismo proceso: pueden acceder a las mismas variables globales, la misma memoria de heap, mismos descriptores de archivos, etc.

2. Ventajas y Desventajas

Como los threads dentro de un grupo comparten el mismo espacio de memoria, si uno corrompe el espacio de su memoria, los otros también sufren las consecuencias, por otro lado con un proceso el sistema operativo los protege unos de otros. Sin embargo mejora el rendimiento, podemos priorizar alguna tarea sobre del resto y permite un cambio de contexto más rápido.

3. ¿Cuándo usar multithreading?

No siempre se debe utilizar pues no implica que se tendrá una mejora, pues si la aplicación es muy simple no tiene sentido plantearse este tipo de programación, tampoco para aplicaciones excesivamente complejas porque puede provocar sobre carga. Una aplicación es candidata a ser programada en multithreading si:

- Requiere hacer varias tareas claramente diferenciadas con un coste computacional mayor al que se genera en la liberación del espacio asignado.
- El resultado de cada tarea es independiente del de otra.
- Se puede preveer que halla tareas retenidas o bloqueadas por esperar lectura de disco, en este caso esta tarea se bloquea y otra comienza a ejecutarse.

3.1. Como utilizarlo

Para importar el paquete: `import threading`
Creamos el thread `t`: `t = threading.Thread(target = 'función a ejecutar por hilo', args = 'argumentos que se le pasaran a la función de destino')`
Iniciamos el thread `t.start()`
Para detener la ejecución del programa hasta que se acomplete un subproceso: `t.join()`
`os.getpid ()` para obtener la identificación del proceso actual
Para obtener thread principal: `threading.main`$_t hread()$