

Prácticas Concurrencia y Distribución (18/19)

Arno Formella, Anália García Lourenço, Hugo López Fernández, David Olivieri

semana 15 febrero – 21 febrero

4. Práctica 4: Sincronización de hilos

Objetivos: Sincronización de hilos

1. **(P4: para entregar en grupo de práctica):** Contador sincronizado. El objetivo de este problema es escribir un contador concurrente y sincronizar el acceso de un grupo de hilos a un acumulador. Sigue estos pasos:
 - a) **Clase Counter:** crea una clase `Counter` que tenga un método `increment()` que incremente el valor del contador en una unidad y un método para obtener el valor actual del contador. El valor inicial del contador debe ser 0.
 - b) **Clase MyTask:** crea una clase `MyTask` que extienda `Thread` o implemente `Runnable` y que se pueda construir recibiendo como parámetro un objeto de tipo `Counter`. Esta tarea debe dormir un tiempo aleatorio entre 0 y 100 milisegundos y a continuación invocar el método `increment()` del contador.
 - c) **Método principal:** crea un método principal que construya varios objetos de la clase `MyTask` que compartan un objeto de tipo `Counter`, ejecuta cada tarea en un hilo y espera a que todos los hilos hayan finalizado. Cuando esto ocurra, imprime el valor actual del contador. Ejecuta el programa varias veces y con distintos números de hilos (1, 2, 4, 8, 16, 32, ..., 1024). Explica el comportamiento de la salida.
 - d) **Bloques sincronizados:** Modifica la clase `MyTask` para que el incremento del contador se haga desde un bloque sincronizado (el bloque con la palabra reservada `synchronized` debe actuar sobre el objeto contador compartido). ¿Cómo cambia esto el comportamiento del programa principal?
 - e) **Utilizando Java AtomicInteger/LongAdder:** Haz las modificaciones necesarias en el código anterior para usar un `AtomicInteger` y/o `LongAdder`. Para realizar la operación del incremento, usa un método adecuado del gran conjunto disponible para estos objetos (consulta la documentación de la API de Java). Explica lo que observas.
2. **(P3: para entregar dentro de una semana):** Mas sobre sincronización.

El propósito de este problema es estudiar más a fondo la sincronización de hilos utilizando el código del problema 1.

- a) **Java Locks:** En lugar de bloques sincronizados, la API de Java tiene un conjunto de objetos optimizados de alto nivel para implementar una concurrencia eficiente. En particular, aquí vas a utilizar el paquete `java.util.concurrent.locks`, que proporciona un mecanismo de bloqueo similar al método sincronizado. Reemplaza el bloque sincronizado con un `lock`.

- b) **Análisis/Medidas:** Ejecuta el código del contador para todos los métodos de sincronización empleados variando la cantidad de hilos utilizados. ¿Puedes notar una diferencia en el rendimiento? Explica tus resultados.
- c) **Between Atomic Operations:** El código anterior ilustra que las operaciones atómicas están protegidas. Sin embargo, ahora considera la adición de dos contadores, p y q , donde el acumulador es: $q \leftarrow q + p$. Este problema demuestra que las operaciones atómicas están protegidas, mientras que las operaciones entre sí no lo están. Modifica tu código para resolver este problema de calcular correctamente q usando la sincronización según corresponda.