Semana 4: P3

Sincronización de hilos

En este ejercicio utilizamos el paquete java.util.cocurrent.locks, que proporciona un mecanismo de bloqueo de recursos similar al bloque synchronized, analizamos y medimos las diferencias entre ambos y combinamos operaciones atómicas entre sí.

Java Locks

Reemplazamos el bloque synchronized que realizamos en el ejercicio 3 de la Práctica 4 por la utilización de un objeto lock, de manera que bloqueamos el recurso, realizamos las operaciones críticas y luego lo desbloqueamos o liberamos.

Análisis y medidas

Ejecutamos el código de los ejercicios realizados en la Práctica 4 y el del ejercicio anterior variando la cantidad de hilos utilizados y podemos observar que al utilizar el bloque synchronized con un gran número de hilos como, por ejemplo, 10.000, el programa finaliza antes que utilizando lock por varios segundos.

Combinación de operaciones atómicas

Creamos un nuevo programa cogiendo el código del anterior, pero ahora añadiremos un nuevo contador y al otro contador, después de incrementarlo en la ejecución de cada hilo, le asignaremos el valor resultante de sumar ambos contadores en ese momento.

Conclusión

Con las pruebas realizadas podemos afirmar que es más eficiente utilizar un bloque synchronized, sin embargo, esto no implica que lock no sea útil.

Lock puede utilizar condiciones desbloqueando los recursos para determinados tipos de operaciones, permite desbloquear hilos en orden de llegada y no tendremos el problema de que nos bloquee los recursos indefinidamente como pasa en synchronized.

En consecuencia, lock es mucho más versátil que synchronized aunque consuma más recursos, por lo que su utilización dependerá del uso que le dé el programa en cuestión.

Fuentes

Portal Chuwiki, artículo "Synchronized vs lock"
http://chuwiki.chuidiang.org/index.php?title=Synchronized_vs_lock

Java SE 7 Documentation, "Interface lock"
https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/Lock.html