Escuela Técnica Superior de Ingeniería

Prácticas de Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

PRÁCTICA 9

Thread Pools mediante java.util.ExecutorService

El objetivo de la práctica es familiarizarse con el mecanismo ejecución de hilos mediante *Thread Pool*, a través de `ExecutorService`

La práctica consistirá en descifrar una contraseña numérica de 6 dígitos usando la fuerza bruta. El procedimiento de búsqueda de contraseña por fuerza bruta consiste en probar todas las combinaciones posibles.

Normalmente, los ataques a contraseñas se realizar sobre el archivo de contraseñas cifradas, es decir, se dispone de una contraseña cifrada y se desea conocer la original. Para ello, el ataque por fuerza bruta prueba todas las posibles combinaciones del alfabeto (conjunto de caracteres posibles) de la clave y, para cada combinación, hace el cifrado de la misma y la compara con el cifrado que se tiene de la clave correcta. Si ambos coinciden, dicha combinación corresponde con la clave original.

Lógicamente, mientras más amplio sea el alfabeto usado y más caracteres tenga la clave, más amplio es el espacio de búsqueda y más tiempo se tardará en probar todas las combinaciones.

En nuestro caso, sabemos que la clave contiene 6 dígitos (de 0 a 9), con lo cual hay un millón de posibles combinaciones (desde 000000 a 999999). Además, usaremos el algoritmo SHA256 como algoritmo de cifrado. Dicho algoritmo es un algoritmo de *hash*, que produce una salida distinta en función de la entrada proporcionada, y que es irreversible, es decir, no es posible descubrir la entrada original conociendo el *hash.*

Para acortar el proceso de búsqueda lo vamos a dividir en 10 tareas, de forma que cada tarea puedan hacer la búsqueda de forma paralela con el resto. Para tal fin, cada una de las tareas fijará el primer dígito del espacio de búsqueda, de forma que sólo hace la búsqueda de los siguientes 5 dígitos. Así la tarea 0 buscará desde 000000 a 099999, y la tarea 5 desde 400000 a 499999. Si todas las tareas se ejecutan en paralelo, dividimos por 10 el tiempo de búsqueda.

Dado que para sacar partido del paralelismo dependemos del número de procesadores disponibles, vamos a asignar cada tarea a un hilo distinto, que se debe ejecutar en un *Thread Pool,* cuyo tamaño podremos elegirlo dependiendo del hardware donde corra.

La práctica deberá ser implementada como un *Frame*, que tras ser iniciada lance un hilo de la clase `Supervisor`. El hilo `Supervisor` creará el *Thread Pool*, que correrá el número de hilos elegido según el hardware que tengamos. Una vez creado el *Thread Pool* delegará en él la ejecución de los hilos `Calculador` que serán cada una de las tareas que buscan en cada división del espacio de búsqueda.

Cada hilo `Calculador` deberá devolver un entero al terminar la búsqueda. Si la búsqueda termina sin éxito, el hilo devolverá `-1`, pero el hilo que encuentre la clave, la devolverá y finalizará al encontrarla.

El hilo `Supervisor` deberá recoger los resultados devueltos por los hilos `Calculador`, y cuando uno de ellos devuelva un valor distinto a `-1` finalizará el *Thread Pool* y mostrará el resultado.

El código que sigue calcula el SHA256 de la cadena "123456" y lo muestra en pantalla:

```
MessageDigest md = MessageDigest.getInstance("SHA-256");
String clave="123456";
byte[] hash = md.digest(clave.getBytes());
System.out.println(DatatypeConverter.printHexBinary(hash));
```