Concurrencia y Paralelismo

Grado en Informática 2022

Práctica 3 – MD5 en Erlang

En esta práctica vamos a reimplementar el cálculo de hashes md5 usando paso de mensajes. El ćodigo proporcionado hace el cálculo usando un único proceso.

```
1> break_md5:break_md5("76a2173be6393254e72ffa4d6df1030a").
76A2173BE6393254E72FFA4D6DF1030A: passwd
ok
```

La barra de progreso corre en su propio proceso, y se imprime cada vez que recibe un mensaje con los casos probados desde la última comunicación.

Partiendo de este código se pide:

Ejercicio 1 (**Implemente la comprobación de múltiples hashes**) Implemente una función `break_md5s/1`, que dada una lista de hashes, obtenga los passwords correspondientes. Un ejemplo de ejecución sería:

```
1> break_md5:break_md5s(["76a2173be6393254e72ffa4d6df1030a",
"e80b5017098950fc58aad83c8c14978e"]).
E80B5017098950FC58AAD83C8C14978E: abcdef
76A2173BE6393254E72FFA4D6DF1030A: passwd
```

Ejercicio 2 (**Añada una estimación del número de passwords comprobados por segundo**) Añada a la barra de progreso una estimación del número de passwords comprobados por segundo. Para medir tiempos puede utilizar la función `erlang:monotonic_time(microseconds)`. Esta función devuelve un valor que crece monotamente entre llamadas a la función con los microsegundos que han pasado entre esas llamadas (es decir, no está afectado por cambios en la hora del sistema). Por ejemplo:

```
T1 = erlang:monotonic_time(microseconds),
...
T2 = erlang:monotonic_time(microseconds),
T2 - T1.
```

Calcularía el tiempo pasado entre las dos llamadas.

Ejercicio 3 (**Haga el programa multiproceso**) Al igual que en la práctica anterior, modifica la implementación para que el cálculo de los hashes se haga en multiples procesos. Cuando se encuentre el password para un hash, hay que avisar al resto de procesos para que no lo continuen comprobando. El programa debería parar cuando se hayan encontrado todos los passwords, es decir, no debería quedar en ejecución ninguno de los procesos creados.

Comprobación Finalización Procesos

Para comprobar la finalización correcta de los ficheros puede utilizarse el debugger, que se inicial con `debugger:start()`. A continuación seleccione los módulos de la práctica en `Module=>Interpret`. Durante la ejecución del programa podrá ver los procesos que ejecutan código en esos módulos.

Entrega

La fecha límite de entrega es el 20 de marzo. El código inicial está disponible en github classrooms, en https://classroom.github.com/a/4fONKAef. La corrección se hará sobre el código subido al repositorio el día 20 de marzo.