Prácticas Concurrencia y Distribución (17/18)

Arno Formella, Anália García Lourenço, Lorena Otero Cerdeira, David Olivieri

semana 19 febrero – 25 febrero

2. Semana 2 (19–25/02): Comportamiento básico de los hilos

Objetivos: sincronización simple con los hilos al final de ejecución, medición de tiempo de ejecución

1. Determinación de la terminación del hilo **(P4: para entregar en grupo de práctica)**.

   Este ejercicio explora cómo determinar cuando termina un hilo o un grupo de hilos. Para hacer esto, siga estos pasos:

   a) **Configuración del problema:** Duplicar el código de la semana pasada. En particular, escriba una clase `MyThread` que *extends* `Thread` (o *implements* `Runnable`) e imprima el nombre del hilo de ejecución. (Ten en cuenta que tu versión podría tener una estructura ligeramente diferente y/o con diferentes nombres de clase, pero esencialmente lograr el mismo resultado).

   b) **Detalles:** en el método `main` de la clase principal), cree una lista (o matriz) de hilos e inícielos. Inmediatamente después de iniciar los hilos, desde el hilo principal, imprima a la pantalla un mensaje (algo como: *el programa ha terminado*). Una estructura en Java para crear una lista de hilos podría ser algo como lo siguiente:

   ```
   final int NUMBER_OF_THREADS = 32;

   List<Thread> threadList =
     new ArrayList<Thread>(NUMBER_OF_THREADS);

   for(int i=1; i<=NUMBER_OF_THREADS; ++i) {
     //...
   }
   ```

   ¿Cuál es el resultado de tu código? ¿En qué orden se imprimen los hilos?

   c) **Modificar la clase principal:** ahora queremos imprimir un mensaje desde el hilo principal cuando todos los otros hilos han terminado. Usando el método `isAlive()` en un bucle de control en la rutina principal, se puede determinar el estado de cada hilo (si todavía está vivo). La estructura del bucle/`isAlive()` para capturar el estado de cada subproceso debería tener una estructura similar a este:

   ```
   while (t.isAlive ())
     try {
     }
     catch () {
     }
   ```

   donde `t` es uno de los hilos.

d) **Modificación con `join`:** Dado que la técnica anterior de bucle/`isAlive()` se utiliza con tanta frecuencia, Java tiene un método especial llamado `join()` que hace lo mismo. Reemplace el bucle/`isAlive()` de arriba con `join()` ¿Funciona exactamente igual? ¿Afecta la ejecución de los subprocesos en ejecución?

2. Medición del tiempo de ejecución **(P3: para entregar dentro de una semana)**.

   Queremos mapear el ciclo de vida de los hilos en el programa concurrente registrando los tiempos cuando cambian sus estados. Sigue los pasos aquí para investigar esto:

   a) **Configuración del problema:** escriba una clase principal (que contiene `main`) y una clase que *extends* `Thread` como en el problema 1. En la clase principal, comienza la ejecución de una lista de hilos que van a realizar una operación matemática (que se explica en el siguiente paso).

   b) **Implementación de la clase `MyThread`:** el hilo debe ejecutar una operación de cálculo intensivo. Para esto, una prueba históricamente interesante es la *prueba de remojo PDP-11*, (vea `https://en.wikipedia.org/wiki/PDP-11`) que se basa en la aplicación continua de funciones trascendentales. Para esto, podemos aplicar continuamente el tangente y su inverso, seguido por la raíz cubo. Una implementación es la siguiente:

```
for(int i=0; i<1000000; i++) {
  double d = tan(atan(
    tan(atan(
      tan(atan(
        tan(atan(
          tan(atan(123456789.123456789))
        ))
      ))
    ))
  ));
  cbrt(d);
}
```

   c) **Diagrama de ejecución de hilos:** ahora queremos entender lo que hace cada hilo durante su vida. Inserte diagnósticos (utilizando `System.Print.` o `Logger`) en tu código para mapear el ciclo de vida de cada hilo. Debes distinguir entre el momento en que se ejecuta y el momento en que termina. ¿Puedes distinguir entre la creación del hilo, la ejecución y la terminación final? ¿Qué problemas encuentras cuando intentas determinar las diferentes fases? El siguiente segmento de código podría ser útil (el mensaje debe también incluir el tiempo):

```
LOGGER.debug("Soy {}", Thread.currentThread().getName());
```

   d) **Análisis. Tiempo de ejecución vs número de hilos:** Estudia el comportamiento de tu código a medida que aumentas la cantidad de subprocesos implementados; es decir, comienza con un hilo y aumenta a un gran número de hilos, imprimiendo el tiempo total de ejecución. Sería útil ejecutar su código Java en un script bash (o en Windows, utilizando algo similar) y guarda tus resultados en un archivo.

   e) **Haz un gráfico de tiempo de ejecución:** desde el archivo del paso anterior, utiliza un programa gráfico como por ejemplo *gnuplot*, *matplotlib* o *seaborn* para hacer gráficos del tiempo de ejecución en función del número de subprocesos (es decir, el eje X es el número de subprocesos, mientras que el eje Y es el tiempo de ejecución). Debes hacer el gráfico con distintos escalas: lineal, semilogarítmica y log-log. A partir de estos gráficos, ¿qué conclusiones puedes sacar sobre el aumento de la cantidad de hilos?