Práctica 1

Fernando Cruz Pineda
Alexys Gómez Elizalde
Moisés Corpus García

Ejercicios

1. Lee lo siguiente `https://www.evanjones.ca/software/threading-linus-msg.html` y comparte en máximo 4 líneas de computadora a qué se refiere Linus Torvalds con un contexto de ejecución y cómo se relaciona con la definición en la sección 1 de esta práctica.

   Linus explica que un contexto de ejecución es todo lo que necesita un hilo o proceso para ejecutarse: su stack, registros y estado. Se relaciona con la definición de la práctica porque ahí se habla de los hilos como "un flujo de control con su propio contexto de ejecución" que permite correr varias tareas en paralelo

2. ¿Cuántos hilos tiene disponibles tu computadora?
   Ejecuta `Runtime.getRuntime().availableProcessors()`, si son más de uno en el equipo escriban el de cada uno.

   Número de hilos de Fernando Cruz: 8

   ```
   [fernandocruz@fedora programas]$ java HilosDisponibles
   Número de hilos/procesadores lógicos disponibles: 8
   [fernandocruz@fedora programas]$
   ```

   Número de hilos de Alexys Gómez:12

```
alexysge@AlexLap:~/Documentos/Concurrente$ java hil
Hilos disponibles: 12
alexysge@AlexLap:~/Documentos/Concurrente$
```

Número de hilos de Moises:16

```
PS C:\Users\Moyo_\OneDrive\Escritorio\Java Concurrente> java HilosDisponibles
NÃºmero de hilos/procesadores lÃ³gicos disponibles: 16
```

3. Revisa el programa Determinante Concurrente y responde ¿Cuánto tiempo tarda en ejecutarse?

Tiempo en equipo de Fernando: 1755939 ns

```
[fernandocruz@fedora programas]$ java DeterminanteConcurrente
Program took 1755939ns, result: -18
[fernandocruz@fedora programas]$
```

Tiempo en equipo de Alexys: 2000939 ns

```
alexysge@AlexLap:~/Documentos/Concurrente$ javac De
alexysge@AlexLap:~/Documentos/Concurrente$ java De
Program took 2000939ns, result: -18
alexysge@AlexLap:~/Documentos/Concurrente$
```

Tiempo en equipo de Moises: 570900 ns

```
PS C:\Users\Moyo_\OneDrive\Escritorio\Java Concurrente> java DeterminanteConcurrente
Program took 570900ns, result: -18
```

4. El programa Determinante Concurrente está implementado extendiendo la clase Thread. Implementa el programa utilizando la interfaz Runnable.

   Programa en programas/DeterminanteConcurrenteRunnable.java

5. Implementa el programa Determinante Concurrente de forma secuencial.

   Programa en programas/DeterminanteSecuencial.java

6. Implementa el programa Determinante Concurrente para dos hilos (en vez de seis).

   Programa en programas/DeterminanteDosHilos.java

7. Compara las 3 implementaciones: el programa Determinante Concurrente para dos hilos, para seis hilos y el programa secuencial. Responde: ¿A qué se debe el orden en el que se ordenan los tiempos de ejecución de cada programa?

   - **Secuencial:** 4518 ns
   - **2 Hilos:** 1996043 ns
   - **6 Hilos:** 1638567 ns

   La implementación de 6 hilos es muy ineficiente porque cada hilo solo hace una multiplicación simple, pero el programa debe esperar con join() a que todos terminen para hacer las sumas. El tiempo de sincronización de hilos es mayor que hacer todas las operaciones secuencialmente. Esto ocurre porque enfrentamos un problema de granularidad muy fina el trabajo por hilos es tan mínimo que el overhead de crear y sincronizar hilos supera enormemente el beneficio de la paralelización. Para operaciones computacionalmente sencillas como una matriz 3x3, la concurrencia introduce más costo que beneficio.

8. Si utilizas la Ley de Amdahl entre el programa Determinante Concurrente para dos hilos y el programa secuencial. ¿El resultado es mayor o menor a 1? ¿Por qué?

$$S = \frac{T_{secuencial}}{T_{paralelo}} = \frac{4518}{1996043} \approx 0.00226$$

   El resultado de aplicar la Ley de Amdahl es menor a 1, porque el tiempo del programa concurrente con dos hilos fue mayor que el del secuencial. Esto se debe a la sobrecarga (overhead) de manejar hilos para un problema tan pequeño.

9. Describe con tus propias palabras en máximo dos líneas para qué sirve el método `join()`. Si no utilizas el método `join()` en Determinante Concurrente, ¿sigue funcionando?

   El método join() sirve para que un hilo espere a que otro termine su ejecución antes de continuar. Si no se usa en Determinante Concurrente, el programa podría dar resultados incorrectos, porque el hilo principal podría intentar usar los valores antes de que los hilos terminen de calcularlos.