Programación concurrente

1. Introducción

La programación concurrente es un modelo de programación que permite ejecutar varias tareas de forma simultánea o intercalada, con el objetivo de mejorar la eficiencia y el rendimiento de los programas. A diferencia de la programación secuencial, donde las instrucciones se ejecutan una tras otra, en la programación concurrente varios procesos pueden avanzar al mismo tiempo, compartiendo los recursos del sistema como la memoria y el procesador.

Este tipo de programación se utiliza cuando es necesario realizar varias acciones que pueden ocurrir al mismo tiempo, por ejemplo, en un sistema operativo que ejecuta múltiples programas, o en una aplicación que descarga archivos mientras permite al usuario seguir trabajando. Gracias a la concurrencia, los programas se vuelven más rápidos, eficientes y fluidos, especialmente en sistemas con varios núcleos de procesamiento.

En lenguajes como Java, la concurrencia se maneja mediante hilos (threads), que son unidades de ejecución que pueden correr de manera independiente. Por ejemplo

```
Thread hilo1 = new Thread(() -> {
    System.out.println("Descargando archivo...");
});

Thread hilo2 = new Thread(() -> {
    System.out.println("Mostrando progreso...");
});

hilo1.start();
hilo2.start();
```

En este ejemplo, ambos hilos se ejecutan de forma concurrente: uno realiza la descarga mientras el otro muestra el progreso, sin que el programa se detenga.

La programación concurrente es esencial en el desarrollo de aplicaciones modernas, como videojuegos, sistemas bancarios, servidores web y programas que manejan grandes cantidades de datos. Sin embargo, también implica mayor complejidad, ya que es necesario coordinar las tareas para evitar errores al compartir recursos.

2. Objetivo General

Comprender la programación concurrente, a través de algunos ejemplos prácticos con el lenguaje de programación de Java.

3. Objetivo Especifico

Codificar ejemplos de programación con hilos en el lenguaje de programación de Java.

4. Materiales/utensilios y equipos

Cantidad	Descripción	Especificaciones	Observaciones
1	Equipo de cómputo con acceso a internet, con Netbeans instalado	Cuenta con un procesador de AMD A9-9420, con una RAM de 12,00Gb y un sistema operativo de 64 bits	
2	Equipo de cómputo con acceso a internet, con Netbeans instalado	Cuenta con un procesador de 12th Gen Intel(R) Core(TM), con una RAM de 8,00Gb y un sistema operativo de 64 bits	
3	Equipo de cómputo con acceso a internet, con Netbeans instalado	Cuenta con un procesador de 12th Gen Intel(R) Core(TM), con una RAM de 16,0Gb y un sistema operativo de 64 bits	

5. Desarrollo de la actividad práctica

Para hablar de la programación concurrente debemos tener primero en claro un par de conceptos como lo son: concurrencia, paralelismos y *programación estructurada*, puntualmente estructura secuencial.

Programación estructurada

La programación estructurada es un paradigma de programación que busca mejorar la claridad, calidad y tiempo de desarrollo de un programa al organizar el código en bloques lógicos utilizando solo tres estructuras de control básicas: secuencia (instrucciones se ejecutan una tras otra), selección (condicionales como if y switch) e iteración (bucles como for y while). Su objetivo principal es evitar el uso de saltos incondicionales (goto) que generan el "código espagueti", el cual es difícil de entender y mantener.

Ejemplo.

```java
public class PomedioEstructurado {


    // Método principal: donde inicia el programa
    public static void main(String[] args) {

        // 1️⃣ Declaración de variables
        double numero1 = 8.5;
        double numero2 = 7.3;
        double numero3 = 9.2;

        // 2️⃣ Proceso: cálculo del promedio
        double promedio = calcularPromedio(numero1, numero2, numero3);

        // 3️⃣ Salida: mostrar el resultado
        mostrarResultado(promedio);
    }

    // Método para calcular el promedio
    public static double calcularPromedio(double a, double b, double c) {
        double suma = a + b + c;
        double promedio = suma / 3;
        return promedio;
    }

    // Método para mostrar el resultado
    public static void mostrarResultado(double promedio) {
        System.out.println("El promedio es: " + promedio);
    }
}
```

Programación paralela

La programación paralela es un modelo de programación que permite ejecutar varias tareas al mismo tiempo, aprovechando múltiples procesadores o núcleos de una computadora.

En palabras simples:

Imagina que tienes que lavar ropa, cocinar y limpiar. Si haces una cosa a la vez, eso es programación secuencial, pero si tus cocinas mientras la lavadora lava y otra persona limpia, entonces las tareas se realizan en paralelo.

Ejemplo.

```
public class PorgramacionParalelo {


    public static void main(String[] args) {
        // Crear dos tareas (hilos)
        Thread tarea1 = new Thread(() -> {
            for (int i = 1; i <= 5; i++) {
                System.out.println("Tarea 1 - Paso " + i);
            }
        });

        Thread tarea2 = new Thread(() -> {
            for (int i = 1; i <= 5; i++) {
                System.out.println("Tarea 2 - Paso " + i);
            }
        });

        // Iniciar ambas tareas al mismo tiempo
        tarea1.start();
        tarea2.start();
    }
}
```

Programación concurrente

La programación concurrente es una técnica informática que permite que múltiples tareas se ejecuten al mismo tiempo, ya sea en un solo procesador o en varios, para mejorar el rendimiento y la capacidad de respuesta de una aplicación. A diferencia del paralelismo, la concurrencia no implica necesariamente que todas las tareas se ejecuten simultáneamente en el mismo instante, sino que el sistema puede intercalar la ejecución de diferentes tareas para dar la impresión de que están sucediendo a la vez.

Conceptos Clave

- Tareas: Son las unidades de trabajo que se ejecutan de forma concurrente, como procesos o hilos de ejecución.
- Hilos (Threads): Son la forma más común de implementar la concurrencia, permitiendo que dentro de un mismo programa se ejecuten múltiples secuencias de instrucciones simultáneamente.
- Ejecución simultánea: La concurrencia busca que las tareas se superpongan en tiempo, no que se intercalen sin parar.

Características y Problemas

- Beneficios: Permite mejorar el rendimiento de las aplicaciones al gestionar eficientemente el hardware y al mantener la capacidad de respuesta del sistema.

- Complejidad: Introduce problemas complejos relacionados con la comunicación entre tareas y la sincronización para acceder a recursos compartidos.
- Problemas comunes: Pueden surgir condiciones de carrera (carreras de acceso a recursos), bloqueo mutuo (deadlocks), o inanición (starvation), donde una tarea nunca puede ejecutarse.

Cómo funciona

En un solo procesador: El sistema operativo puede dividir el tiempo de ejecución de la CPU entre las distintas tareas, dándoles la sensación de que se ejecutan simultáneamente.

En múltiples procesadores: Si el hardware tiene múltiples núcleos o procesadores, las tareas realmente pueden ejecutarse al mismo tiempo, aprovechando el paralelismo.

Ejemplo práctico

Imagina una persona que está navegando por internet, escuchando música y escribiendo un correo electrónico. Aunque sea una sola persona, tiene la capacidad de hacer estas tres cosas a la vez, intercalando entre ellas. En la programación concurrente, la aplicación hace algo similar: cambia rápidamente entre las diferentes tareas para que parezcan estar funcionando a la vez.

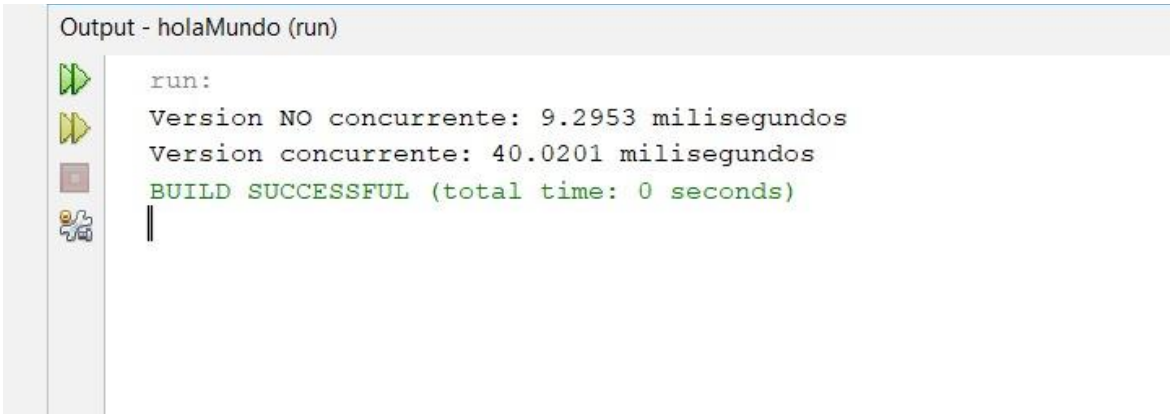
6. Resultados

Código

```java
import java.util.Random;


public class principal extends Thread {
    private static double[] vec = new double[1000000];
    private int inicio, fin;

    public principal(int inicio, int fin) {
        this.inicio = inicio;
        this.fin = fin;
    }

    public static void main(String[] args) {
        iniciavec();
        vec_NOConcurrente();
        vec_concurrente();
    }

    private static void iniciavec() {
        Random rand = new Random(System.nanoTime());
        for (int i = 0; i < vec.length; i++) {
            vec[i] = rand.nextInt();
        }
    }

    // INSTRUCCIONES SECUENCIALES
    private static void vec_NOConcurrente() {
        double tiempo = System.nanoTime();
        for (int i = 0; i < vec.length; i++) {
            vec[i] /= 10;
            vec[i] *= 10;
```

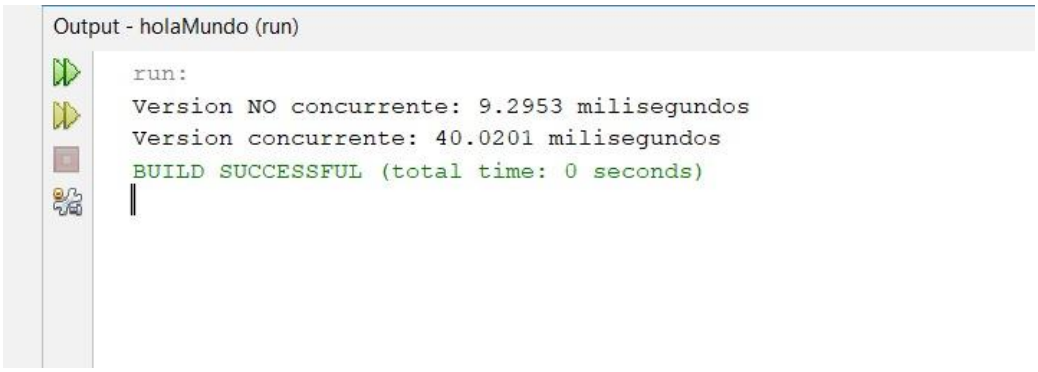
```
            vec[i] /= 10;
        }
        System.out.println("Version NO concurrente: " + ((System.nanoTime() - tiempo) / 1000000) + " milisegundos");
    }

    // INSTRUCCIONES CONCURRENTES
    private static void vec_concurrente() {
        int nproc = Runtime.getRuntime().availableProcessors();
        int inicio = 0, fin = 0;
        principal[] prin = new principal[nproc];
        double tiempo = System.nanoTime();

        for (int i = 0; i < nproc; i++) {
            inicio = fin;
            fin += vec.length / nproc;
            prin[i] = new principal(inicio, fin);
            prin[i].start();
        }

        for (int i = 0; i < nproc; i++) {
            try {
                prin[i].join();
            } catch (Exception e) {}
        }

        System.out.println("Version concurrente: " + ((System.nanoTime() - tiempo) / 1000000) + " milisegundos");
    }

    public void run() {
        for (int i = inicio; i < fin; i++) {
            vec[i] /= 10;
```

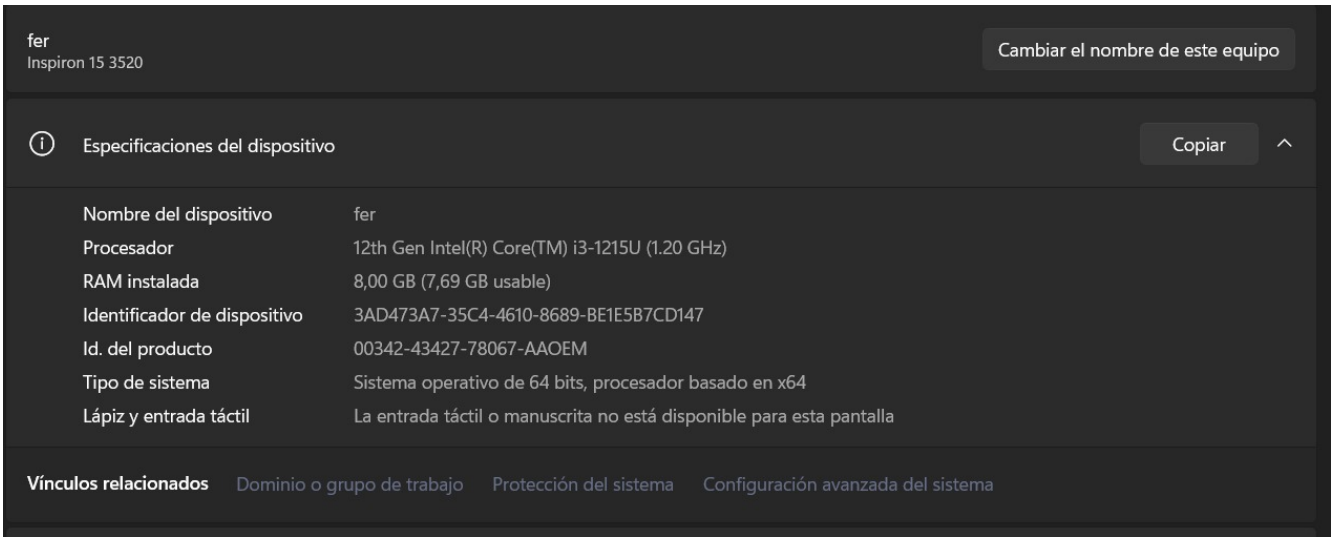
Primer equipo




Sistema › Acerca de
DESKTOP-812TQ52
HP Laptop 15-bw0xx
Cambiar el nombre de este equipo
Especificaciones del dispositivo
Copiar
Nombre del dispositivo DESKTOP-812TQ52
Procesador AMD A9-9420 RADEON R5, 5 COMPUTE CORES 2C+3G 3.00 GHz
RAM instalada 12.0 GB (11.5 GB usable)
Identificador de dispositivo 50BC1C40-8446-4A99-B489-87BF6677B4B2
Id. del producto 00329-00000-00003-AA896
Tipo de sistema Sistema operativo de 64 bits, procesador basado en x64
Lápiz y entrada táctil La entrada táctil o manuscrita no está disponible para esta pantalla
Vínculos relacionados Dominio o grupo de trabajo Protección del sistema Configuración avanzada del sistema
Especificaciones de Windows
Copiar
Edición Windows 11 Enterprise
Versión 21H2
Instalado el 31/01/2024
Versión del sistema operativo 22000.493
Experiencia Paquete de experiencia de características de Windows 1000.22000.493.0

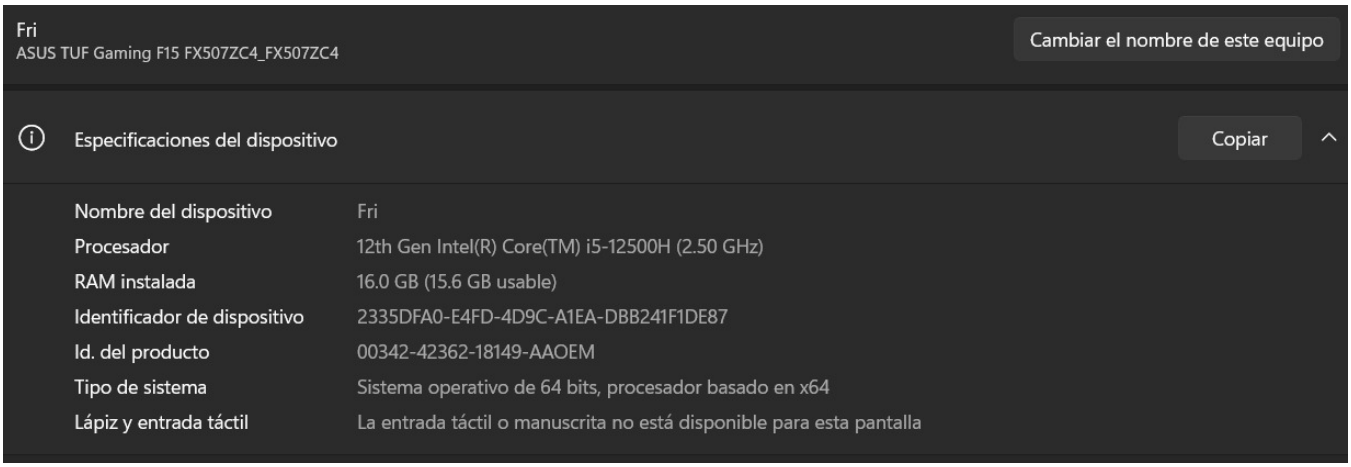
Equipo 2


Output - holaMundo (run)
run:
Version NO concurrente: 9.2953 milisegundos
Version concurrente: 40.0201 milisegundos
BUILD SUCCESSFUL (total time: 0 seconds)


fer
Inspiron 15 3520
Cambiar el nombre de este equipo
Especificaciones del dispositivo
Copiar
Nombre del dispositivo fer
Procesador 12th Gen Intel(R) Core(TM) i3-1215U (1.20 GHz)
RAM instalada 8,00 GB (7,69 GB usable)
Identificador de dispositivo 3AD473A7-35C4-4610-8689-BE1E5B7CD147
Id. del producto 00342-43427-78067-AAOEM
Tipo de sistema Sistema operativo de 64 bits, procesador basado en x64
Lápiz y entrada táctil La entrada táctil o manuscrita no está disponible para esta pantalla
Vínculos relacionados
Dominio o grupo de trabajo
Protección del sistema
Configuración avanzada del sistema

Equipo 3


Output - EjemploThread (run)
run:
Version NO concurrente: 7.7873 milisegundos
Version concurrente: 18.4665 milisegundos
BUILD SUCCESSFUL (total time: 0 seconds)



7. Discusión

En el Equipo 1, con un procesador de doble núcleo y 12 GB de memoria RAM, el programa logró ejecutar las tareas concurrentes, pero con un tiempo de respuesta más alto. Los hilos se ejecutaron de manera intercalada, sin aprovechar completamente la concurrencia, debido a la limitación de núcleos. El resultado fue correcto, pero el proceso tardó más en completarse

En el Equipo 2, con un procesador de doble núcleo y 8 GB de memoria RAM, el programa logró ejecutar las tareas concurrentes, pero con un tiempo de respuesta más alto. Aunque fue más bajo que el equipo 1, aun así quedo por debajo del equipo 3 debido a la arquitectura de esta.

Equipo 3 (ASUS TUF Gaming F15 – tu equipo): Procesador Intel Core i5-12500H (12 núcleos híbridos) y 16 GB de RAM.

Este equipo mostró el mejor desempeño. Gracias a su arquitectura moderna con múltiples núcleos de alto rendimiento, el programa ejecutó las tareas concurrentes casi en paralelo, aprovechando eficientemente la memoria y el procesador. El tiempo total de ejecución fue el más bajo de los tres

8. Cuestionario

1.- ¿Cuáles son las ventajas de utilizar la programación concurrente?

1. Mejor aprovechamiento del procesador**:** mientras una tarea espera (por ejemplo, una lectura de archivo), otra puede avanzar.
2. Mayor eficiencia**:** el sistema puede realizar más trabajo en menos tiempo.
3. Mejor experiencia del usuario: los programas no se "congelan" mientras realizan tareas pesadas (por ejemplo, al descargar y mostrar datos al mismo tiempo).

4. Permite dividir tareas complejas**:** facilita organizar un programa en partes que trabajan de forma independiente.

2.- ¿Cuáles son las desventajas de utilizar la programación concurrente?

1. Mayor complejidad en el diseño y la programación.
2. Errores difíciles de detectar, como los de sincronización o acceso simultáneo a recursos (llamados *condiciones de carrera*).
3. Dificultad para depurar (debuggear) y probar los programas.
4. Puede consumir más recursos (memoria y CPU) si no se maneja correctamente.

3.- ¿Son iguales la programación paralela y la concurrente? Explica ¿por qué?

No, no son iguales.

1. Programación concurrente: varias tareas avanzan durante el mismo periodo de tiempo, pero no necesariamente se ejecutan al mismo instante. Ejemplo: una computadora con un solo procesador que alterna rápidamente entre tareas (como escribir y escuchar música).
2. Programación paralela: varias tareas se ejecutan exactamente al mismo tiempo, cada una en un procesador o núcleo diferente. Ejemplo: en una computadora con varios núcleos, un núcleo reproduce música mientras otro procesa un video.

9. Conclusiones

En conclusión, aunque el resultado lógico del programa fue el mismo en los tres casos (las operaciones se realizaron correctamente), el tiempo de ejecución y la eficiencia variaron considerablemente según las propiedades del equipo. Esto evidencia que el rendimiento de la programación concurrente depende tanto del software como del hardware, especialmente del número de núcleos, la velocidad del procesador y la capacidad de la memoria

10. Referencias Bibliográficas

Referencias

mindware. (26 de enero de 2015). *mindware*. Obtenido de https://mindwaresrl.com/2015/01/26/programacion-estructurada-java-parte-1-de-2