code URJC

Programación Concurrente

Tema 6

Aplicaciones Concurrentes

Micael Gallego
micael.gallego@urjc.es
@micael_gallego

Universidad Rey Juan Carlos

Aplicaciones Concurrentes

code URJC

- **Introducción**
- Aplicaciones con interfaz de usuario
- Aplicaciones Web
- Bases de datos
- Conclusiones

Introducción

- **Cualquier aplicación** se puede implementar con técnicas **concurrentes**
- Dependiendo del **tipo de aplicación**, se obtiene una **ventaja** u otra:
  - Ofrecer un servicio **interactivo** a varios clientes/tareas
  - **Aprovechar** la potencia de computación
    - En sistemas multi-core
    - Con tareas que esperan por entrada/salida

Introducción

- La gran mayoría de las aplicaciones que usamos a diario son concurrentes
  - **Aplicaciones con interfaz de usuario:** Las librerías de interfaz de usuario siguen mayoritariamente la misma estructura de separación en hilos (servicio interactivo y facilidad de desarrollo)
  - **Aplicaciones web:** Cada petición a un servidor web se procesa de forma concurrente (servicio interactivo)
  - **Bases de datos:** Un sistema gestor de base de datos permite la ejecución de consultas de forma concurrente (servicio interactivo)

Aplicaciones Concurrentes

code URJC

- Introducción
- **Aplicaciones con interfaz de usuario**
- Aplicaciones Web
- Bases de datos
- Conclusiones

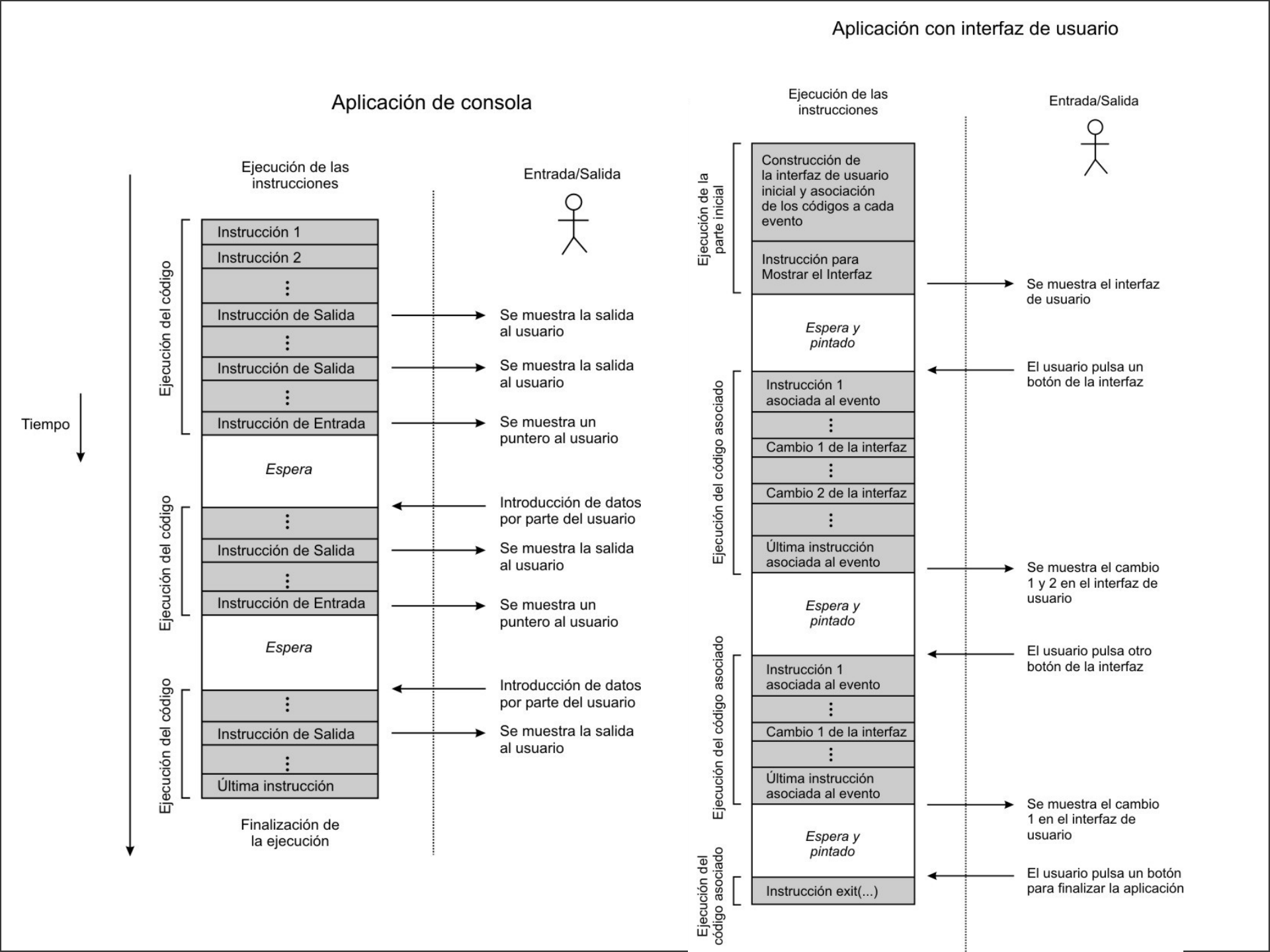
Aplicaciones con interfaz de usuario

code URJC

- La mayoría de las librerías de interfaz de usuario se ejecutan en **un solo hilo**
- A este hilo se le denomina "Hilo de despacho de eventos" (***Event dispatch thread***)
- Todas las operaciones que utilicen elementos del interfaz (**primer plano**) deben realizarse en ese hilo
- Las operaciones en **segundo plano** deben realizarse en **otro hilo** para no bloquear la interfaz

Aplicaciones con interfaz de usuario

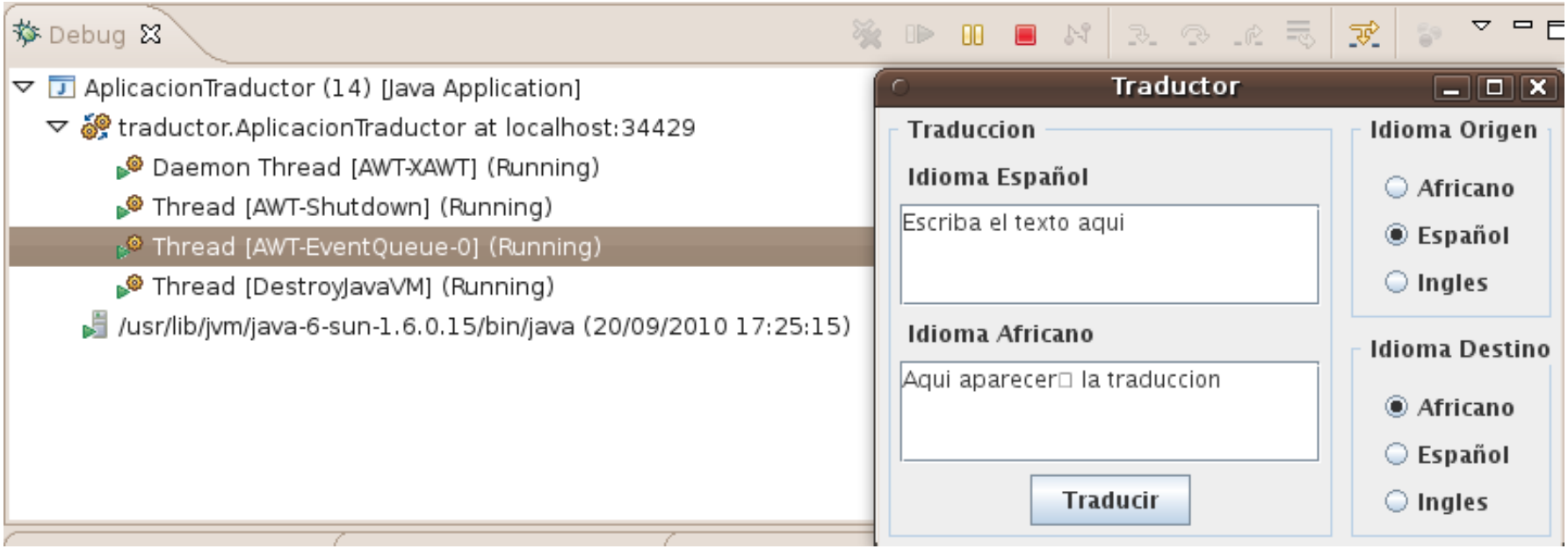
- Es habitual estudiar la programación dirigida por eventos en contraposición a la programación secuencial
  - En la **programación secuencial** el programador define el flujo del programa de principio a fin
  - En la **programación dirigida por eventos** el usuario dirige el flujo del programa interactuando con los elementos del interfaz


Aplicación de consola
Ejecución de las instrucciones
Entrada/Salida
Tiempo
Ejecución del código
Instrucción 1
Instrucción 2
Instrucción de Salida
Instrucción de Salida
Instrucción de Entrada
Espera
Ejecución del código
Instrucción de Salida
Instrucción de Entrada
Espera
Ejecución del código
Instrucción de Salida
Última instrucción
Se muestra la salida al usuario
Se muestra la salida al usuario
Se muestra un puntero al usuario
Introducción de datos por parte del usuario
Se muestra la salida al usuario
Se muestra un puntero al usuario
Introducción de datos por parte del usuario
Se muestra la salida al usuario
Finalización de la ejecución
Aplicación con interfaz de usuario
Ejecución de las instrucciones
Entrada/Salida
Ejecución de la parte inicial
Construcción de la interfaz de usuario inicial y asociación de los códigos a cada evento
Instrucción para Mostrar el Interfaz
Espera y pintado
Ejecución del código asociado
Instrucción 1 asociada al evento
Cambio 1 de la interfaz
Cambio 2 de la interfaz
Última instrucción asociada al evento
Espera y pintado
Ejecución del código asociado
Instrucción 1 asociada al evento
Cambio 1 de la interfaz
Última instrucción asociada al evento
Espera y pintado
Ejecución del código asociado
Instrucción exit(...)
Se muestra el interfaz de usuario
El usuario pulsa un botón de la interfaz
Se muestra el cambio 1 y 2 en el interfaz de usuario
El usuario pulsa otro botón de la interfaz
Se muestra el cambio 1 en el interfaz de usuario
El usuario pulsa un botón para finalizar la aplicación

Aplicaciones con interfaz de usuario

- La pantalla "congelada" o la pantalla en gris aparece cuando una tarea en el **hilo de despacho de eventos** dura mucho tiempo
- Las librerías de interfaz de usuario ofrecen mecanismos para ejecutar tareas en **segundo plano** que tardan cierto tiempo
- Estas tareas de segundo plano pueden **actualizar el interfaz** según progresan

Aplicaciones con interfaz de usuario



Aplicaciones con interfaz de usuario

- **Páginas de interés:**
  - Concurrencia en Swing:
    - http://docs.oracle.com/javase/tutorial/uiswing/concurrency/
    - http://java.sun.com/products/jfc/tsc/articles/threads/threads2.html
  - Hilo de despacho de eventos:
    - http://en.wikipedia.org/wiki/Event_dispatching_thread
  - Por qué Swing no es thread-safe:
    - https://weblogs.java.net/blog/kgh/archive/2004/10/multithreaded_t.html

Aplicaciones Concurrentes

code URJC

- Introducción
- Aplicaciones con interfaz de usuario
- **Aplicaciones Web**
- Bases de datos
- Conclusiones

Aplicaciones web

- Históricamente, en las aplicaciones web cada petición se procesa por un **hilo diferente**
- Esto permite ofrecer un **servicio interactivo** a cada cliente
- La **calidad** del servicio se **degrada** de forma paulatina
- El **número total de hilos** (peticiones concurrentes) se configura en el servidor web

Aplicaciones web

- Tener un hilo por cada petición presenta **problemas de escalabilidad**
- Si para procesar una petición se hace una consulta a la BBDD o a otro sistema, el hilo quedará a la **espera la mayor parte del tiempo**
- El número de hilos de una aplicación está limitado por diversos factores y **no suele ser mayor de 10.000.**
- http://en.wikipedia.org/wiki/C10k_problem

Aplicaciones web

- Para **evitar la limitación** y los problemas de **escalabilidad**, actualmente algunos servidores web **no** utilizan un hilo para **procesar "toda" la petición**
- Cuando se **atiende una petición** al servidor y se quiere hacer una operación de entrada/salida (por ejemplo una **consulta a la BBDD**), en vez de que el hilo quede bloqueado, define el código que se ejecutará cuando el resultado esté disponible (***callback***)
- En ese momento el hilo queda **libre** para atender a una **nueva petición** sin haber procesado completamente la anterior.

Aplicaciones web

- Este modelo de programación de servidores es muy popular en la actualidad con servidores como:
  - **Nginx:** Servidor web escrito en C que dispone de módulos para PHP. Se usa sustituyendo a Apache
  - **Vert.x:** Servidor programado en Java
  - **Node.js:** Plataforma programada en JavaScript

Aplicaciones web

- Independientemente del tipo de servidor web, cuando se procesa una petición se puede acceder a varias zonas de memoria:
  - **ServletContext:** Compartida entre las peticiones de todos los usuarios de la web
  - **HttpSession:** Compartida entre las peticiones de un único usuario
- La información compartida tiene que ser **thread-safe**, porque varios hilos pueden acceder a su información concurrentemente

Aplicaciones Concurrentes

code URJC

- Introducción
- Aplicaciones con interfaz de usuario
- Aplicaciones Web
- **Bases de datos**
- Conclusiones

Bases de datos

- Las **bases de datos** son aplicaciones que están diseñadas para que varios clientes puedan realizar **consultas de forma concurrente.**
- Conceptualmente se puede considerar que cada **consulta se ejecuta en un hilo**

Bases de datos

- La implementación más sencilla sería poner toda la **información bajo exclusión mutua**, de forma que no hubiera interferencia entre hilos
- Pero esa implementación **no sería escalable** y tendría un **rendimiento muy deficiente** con muchas peticiones concurrentes

Bases de datos

- Las bases de datos se han optimizado para ofrecer el mejor rendimiento con la siguientes estrategias
  - **Bloqueos:** El usuario indica el nivel de "exclusión mutua" de la base de datos: pesimista de toda una tabla, pesimista de una fila, optimista, etc.
  - **Transacciones:** El usuario indica cuando comienza y cuando terminan las operaciones que deben ejecutarse de forma atómica. Si una operación no se puede realizar, se aborta (*rollback*) y se reintenta.

Bases de datos

- Existen varios **tipos de bloqueos**:
  - **Bloqueo pesimista:** Los datos se bloquean para que no se puedan usar por una transacción mientras se están usando por otra transacción. Esta solución es como la exclusión mutua.
  - **Bloqueo optimista:** Los datos no se bloquean. Cuando una transacción está a punto de terminar, se verifica si los datos que se han leído han cambiado durante la transacción. En ese caso, se reintenta la transacción desde el principio.

https://es.wikipedia.org/wiki/Control_de_concurrencia_optimista

Bases de datos

- Ventajas y desventajas de los **tipos de bloqueo:**
  - **Bloqueo pesimista:** Puede limitar el rendimiento de la BBDD innecesariamente (si no hay muchas transacciones simultáneas o si no suelen usar los mismos datos)
  - **Bloqueo optimista:** Ofrece mejor rendimiento si no hay muchas transacciones que usan los mismos datos, pero si eso ocurre, puede ser contraproducente.

https://es.wikipedia.org/wiki/Control_de_concurrencia_optimista

Aplicaciones Concurrentes

code URJC

- Introducción
- Aplicaciones con interfaz de usuario
- Aplicaciones Web
- Bases de datos
- **Conclusiones**

Conclusiones

- Cada tipo de aplicación tiene una arquitectura diferente que es muy relevante para la **concurrencia**
- Siempre que se desarrolle hay que tener en cuenta los **aspectos concurrentes de la aplicación**, aunque nosotros no estemos implementando nuestro código de forma concurrente