code URJC

Programación Concurrente

Tema 3

Concurrencia en los lenguajes de programación

Micael Gallego
micael.gallego@urjc.es
@micael_gallego

Universidad Rey Juan Carlos

Concurrencia en los Lenguajes de Programación

- **Introducción**
- Concurrencia en Java
- Concurrencia en C/C++
- Concurrencia en JavaScript
- Conclusiones

Introducción

- Para implementar un programa informático deben seleccionarse la **plataforma de desarrollo**:
  - **Sistema Operativo**: Windows XP, Ubuntu, Windows 7, Mac OS X, iOS, Android, RedHat, …
  - **Lenguaje de programación:** C, C++, Java, Python, COBOL, ActionScript, JavaScript, Ruby…
  - **Librerías**: Las librerías disponibles para cada lenguaje en cada Sistema Operativo

Sistema Operativo

- Hay muchos tipos de **sistemas operativos**:
  - **Familias:** Windows (WinXP, Win7), Unix (linux, MacOS, Android, iOS) y otros tipos como Solaris
  - **Arquitecturas HW:** 32bits vs 64bits (x86 vs AMD64), RISC vs CISC (x86 vs ARM).
  - **Uso (Empotrados, PC Escritorio, Servidores):** iOS, Android, Windows 7, Ubuntu, Solaris, HPUX, AIX…

Lenguaje de programación

code URJC

- Hay muchos **lenguajes de programación**
- Se **diferencian** entre sí por…:
  - Sistema de tipos estático vs dinámico
  - Compilado vs Interpretado vs Máquina Virtual
  - Portables vs Centrado en una Arquitectura
  - Orientados a sistemas vs Lenguajes de script
  - Imperativo vs Declarativo
    - Imperativo: Orientado a Objetos vs Estructurado
    - Declarativo: Funcional vs Lógico

Lenguaje de programación

code URJC

- **Ejemplos**
  - **Java:** Tipos estáticos, portable, orientado a objetos, Máquina virtual
  - **C/C++:** Tipos estáticos, compilado, sistemas, imperativo, estructurado
  - **JavaScript:** Tipos dinámicos, portable, estructurado y orientado a objetos, interpretado y máquina virtual

Librerías

- Las **librerías*** son muy importantes para el desarrollo de aplicaciones
  - **Librerías del sistema operativo:** win32 (windows), POSIX (linux), librerías Android, librerías iOS…
  - **Librerías de la plataforma de desarrollo:** Java, JavaScript (navegador web), Python…
  - **Librerías de terceros:** Librerías comerciales o libres, librerías portables entre sistemas operativos, librerías genéricas vs específicas

Deberían llamarse bibliotecas al ser una traducción de "*library*"

Plataforma de desarrollo

code URJC

- Para desarrollar una aplicación hay que tener en cuenta la **plataforma de desarrollo**
  - Lenguaje de programación
  - Sistema Operativo
  - Librerías
- La programación concurrente está muy influenciada por la **plataforma de desarrollo** elegida

Plataformas de desarrollo

code URJC

- A continuación se realizará una presentación básica de la programación concurrente en las siguientes **plataformas de desarrollo:**

Java **C/C++** **JavaScript**

- Esto permite al alumno tener una visión general de las diferentes formas de utilizar la programación concurrente al desarrollar una aplicación

Concurrencia en los Lenguajes de Programación

- Introducción
- **Concurrencia en Java**
- Concurrencia en C/C++
- Concurrencia en JavaScript
- Conclusiones

Concurrencia en Java

- **Java** es una plataforma de desarrollo formada por
  - Lenguaje de programación Java
  - Máquina virtual de Java (JVM)
  - Librería estándar (API)
- La máquina virtual y la librería estándar ofrece portabilidad en **sistemas operativos** y **plataformas hardware**

Concurrencia en Java

- **Modelo de concurrencia**
  - Java ofrece un modelo de concurrencia de **memoria compartida** integrado en el **lenguaje** Java y en la **librería** estándar
  - El soporte de concurrencia ha **evolucionado mucho en Java**
  - En las últimas versiones aparecen **herramientas de alto nivel** en la librería estándar que permiten al desarrollador **abstraerse de detalles de bajo nivel**

Concurrencia en Java

- La concurrencia también está integrada en el propio **lenguaje de programación**
  - Está **definido** cómo se comparte la memoria entre diferentes hilos en el Modelo de Memoria de Java [1] (**Java Memory Model**)
  - Existen palabras reservadas en el propio lenguaje para delimitar zonas bajo exclusión mutua (**synchronized**) y para especificar que un atributo es compartido entre hilos (**volatile**)

[1] http://www.cs.umd.edu/~pugh/java/memoryModel/jsr-133-faq.html

Concurrencia en Java

- También tiene soporte en la **librería estándar**
  - Como Java es un **lenguaje orientado a objetos**, los hilos se representan como objetos de la clase **java.lang.Thread**
  - Esta clase tiene **métodos para controlar el ciclo de vida del hilo** (configurar su nombre, prioridad, iniciar, esperar a que finalice, etc…)
  - En Java los hilos se pueden **crear y destruir** en **cualquier momento** de la ejecución del programa, **no** tienen que seguir la **rígida estructura** de los programas de **SimpleConcurrent**

Concurrencia en Java

```
package ejemplo.gestionhilo;

public class Programa {
  public static void main(String[] args) {

    new Thread(()->{
       System.out.println("Soy un hilo");
    }).start();

  }
}
```

El código del hilo se codifica en una **expresión lambda**

Comienza la ejecución del hilo, el método **start()** no se espera a que finalice la ejecución del hilo.

Concurrencia en Java

- En java se pueden usar **otros modelos de concurrencia** con **librerías**
  - **Actores:**
    - **Akka:** http://akka.io/
    - **Quasar:** http://www.paralleluniverse.co/quasar/
  - **Memoria software transaccional:**
    - **Multiverse:** http://multiverse.codehaus.org
  - **Comunicando procesos secuenciales (CSP):**
    - **JCSP:** http://www.cs.kent.ac.uk/projects/ofa/jcsp/
  - **Programación funcional:**
    - **Framework Collections en Java 8:** http://java.dzone.com/articles/title-devoxx-2012-java-8
    - **RxJava:** https://github.com/Netflix/RxJava

Concurrencia en Java

- En la mayoría de lenguajes de programación **imperativos** se usan las **mismas herramientas** presentes en **Java**
- La plataforma Java se estudiará en detalle en el **Tema 5 – Programación Concurrente en Java**
- Se desarrollarán programas concurrentes **reales** utilizando herramientas **profesionales**

Concurrencia en los Lenguajes de Programación

- Introducción
- Concurrencia en Java
- **Concurrencia en C/C++**
- Concurrencia en JavaScript
- Conclusiones

Concurrencia en C/C++

- C/C++ es una **familia de lenguajes** de programación de **bajo nivel**
- Están diseñados para el **acceso al hardware** y el **rendimiento**
- Esto hace que se usen especialmente en:
  - Programación de sistemas
  - Sistemas empotrados con recursos limitados
  - Computación de altas prestaciones

Concurrencia en C/C++

- Existen diversos **estándares** de C/C++
  - C90, C99, C++03, C++11, C++14
- Existen **muchos compiladores** de C/C++ con diferentes soporte de los estándares
  - gcc: http://gcc.gnu.org/
  - clang: http://clang.llvm.org/
  - VisualC++: http://www.microsoft.com/visualstudio/
  - Intel C/C++ Compilers: http://software.intel.com/en-us/c-compilers/
  - Hay muchos más… (http://en.wikipedia.org/wiki/C99)

Concurrencia en C/C++

- Existen dos enfoques para programación concurrente en C/C++
  - **Concurrencia en el lenguaje:** Extensiones del lenguaje no estándar que "paralelizan" el código automáticamente para que algunas partes sean ejecutadas en paralelo
  - **Concurrencia con librerías:** Librerías no estándar para el desarrollo de programas concurrentes

Concurrencia en C/C++

- **Concurrencia en el lenguaje**
  - Existen **extensiones** de algunos **compiladores**
  - Amplían C/C++ con nuevas **palabras reservadas**
  - Usando esas palabras reservadas se transforma el código para que sea ejecutado en **diferentes hilos de ejecución**
  - Normalmente transforman bucles de secuencial a paralelo
  - Más importantes: **OpenMP** e **Intel Cilk Plus**

Concurrencia en C/C++

- **OpenMP**
  - Disponible en los compiladores **Intel C/C++**, **gcc**, …
  - Permite ejecutar concurrentemente **bucles**, **bloques**, etc.
  - Trabaja con un esquema **fork / join** para que un hilo principal reparta el trabajo en hilos y espere al resultado de todos ellos
  - http://openmp.org/

Concurrencia en C/C++

- **Ejemplo OpenMP**

```
double ProductoPunto(double* a, double* b, int size)
{
    double c = 0;
    #pragma omp parallel for reduction(+:c)
    for (int i = 0; i < size; ++i)
    {
        c += a[i]*b[i];
    }
    return c;
}
```

Con este ***comando*** se permite paralelizar la ejecución de las iteraciones del buble y luego "reducir" los valores de cada hilo con una operación de **suma**

Concurrencia en C/C++

- **Intel Cilk Plus**
  - Disponible en los compiladores **Intel C/C++** y **gcc**
  - Usando sólo tres **nuevas palabras clave** se pueden **paralelizar** operaciones sobre **arrays**
  - http://www.cilkplus.org/

Concurrencia en C/C++

- **Ejemplo Intel Cilk Plus**

```
int fib(int n)
{
    if (n < 2)
        return n;
    int x = cilk_spawn fib(n-1);
    int y = fib(n-2);
    cilk_sync;
    return x + y;
}
```

Con este ***comando*** se indica que fib(n-1) se puede ejecutar en un nuevo hilo en paralelo con el hilo principal que ejecutará fib(n-2)

Con este ***comando*** se indica que el hilo se bloquea hasta que han terminado los otros hilos

Concurrencia en C/C++

- **Concurrencia con Librerías**
  - La **librería estándar de C++** era muy **básica**, y en la práctica cada sistema operativo ofrecía su propio **conjunto de librerías**
  - **Win32:** Librería de la familia de sistemas operativos windows (95, 98, XP, 2000, Vista, 8..)
  - **POSIX, GTK, KDE…:** Librerías de la familia de sistemas operativos UNIX / Linux (RedHay, Ubuntu…)

Concurrencia en C/C++

- **Win32** y **POSIX** permiten la programación concurrente en un **modelo de memoria compartida**
  - Cada librería con **tipos y funciones (API) diferentes**
  - **Win32**
    - API win32 ofrece las primitivas de concurrencia.
    - http://msdn.microsoft.com/en-us/library/ms686937(v=vs.85).aspx
    - Otras librerías concurrentes para Visual C++: http://archive.msdn.microsoft.com/concrtextras
  - **POSIX**
    - Podemos referirnos a ellos como Native POSIX Threads Library (NPTL) o pthreads
    - https://computing.llnl.gov/tutorials/pthreads/

Concurrencia en C/C++

- Existen algunas librerías multiplataforma para **programación concurrente** con **memoria compartida** que se pueden usar en **Unix, Mac y Windows**
  - **Boost**
    - También tiene otro tipo de funcionalidades
    - http://www.boost.org/doc/libs/1_43_0/doc/html/thread.html
  - **Intel Threading Building Blocks**
    - También dispone de herramientas de alto nivel (estructuras de datos, gestión de hilos, …)
    - http://threadingbuildingblocks.org/

Ejemplo
POSIX pthreads

Código que se ejecuta en el hilo de forma concurrente

Creación de los hilos e inicio de la ejecución

Bloqueo hasta que terminan los hilos

```
void *thr_func(void *arg) {
  printf("hello from new thread\n");
  pthread_exit(NULL);
}

int main(int argc, char **argv) {

  pthread_t thr[NUM_THREADS];
  int i, rc;

  for (i = 0; i < NUM_THREADS; ++i) {
    rc = pthread_create
      (&thr[i], NULL, thr_func, NULL);

    if (rc) {
      fprintf(stderr, "error rc: %d\n", rc);
      return -1;
    }
  }

  for (i = 0; i < NUM_THREADS; ++i) {
    pthread_join(thr[i], NULL);
  }

  return 0;
}
```

Ejemplo
win32 Threads

Código que se ejecuta en el hilo de forma concurrente

Creación de los hilos e inicio de la ejecución

Bloqueo hasta que terminan los hilos

```
DWORD WINAPI thr_func(LPVOID args) {
  printf("hello from new thread\n");
  return 0;
}

int main(int argc, char **argv) {

  DWORD thrIds[NUM_THREADS];
  HANDLE thr[NUM_THREADS];
  int i, rc;

  for (i = 0; i < NUM_THREADS; ++i) {
    thr[i] = CreateThread
       (0,0, thr_func,0,0,&thrIds[i]);

    if (!thr[i]) {
      fprintf(stderr, "error in threads\n");
      return -1;
    }
  }

  for (i = 0; i < NUM_THREADS; ++i) {
    WaitForSingleObject(thr[i], INFINITE);
    CloseHandle(thr[i]);
  }
  return 0;
}
```

Concurrencia en C/C++

- **C++11**
  - Define una forma estándar de crear hilos

```
#include <thread>
#include <iostream>

int main(){
   std::thread t1([](){
      std::cout << "Hello from thread " << std::endl;
   });
   t1.join();
   return 0;
}
```

https://www.classes.cs.uchicago.edu/archive/2013/spring/12300-1/labs/lab6/

Concurrencia en C/C++

- **Conclusiones**
  - Las **plataformas** mayoritarias tienen librerías para la programación concurrente de bajo nivel con **memoria compartida (win32, POSIX…)**
  - El estándar **C++11** ya proporciona una forma estándar de gestionar hilos

Concurrencia en C/C++

- **Conclusiones**
  - Existen **librerías multiplataforma** que proporcionan herramientas de **más alto nivel (Boost, Intel TBB…)**
  - Se pueden usar **variantes del lenguaje C/C++** que ofrecen herramientas del **alto nivel** especialmente diseñadas para **paralelización de algoritmos (OpenMP, Intel Cilk Plus…)**

Concurrencia en los Lenguajes de Programación

- Introducción
- Concurrencia en Java
- Concurrencia en C/C++
- **Concurrencia en JavaScript**
- Conclusiones

Concurrencia en JavaScript

- **JavaScript** es un lenguaje que se diseñó para ejecutarse en el contexto de una **página web** dentro de un **navegador web**
- Los **navegadores** actuales ejecutan el código JavaScript con **máquinas virtuales*** (V8 de Chrome, IonMonkey en Firefox…)
- **JavaScript** es el nombre "popular" del **estándar ECMAScript**
- Muchos de los aspectos de la plataforma de desarrollo con **JavaScript** se definen bajo el estándar **HTML5**

* http://en.wikipedia.org/wiki/JavaScript_engine

Concurrencia en JavaScript

- El estándar **WebWorkers** dentro de HTML5 permite usar un modelo de programación concurrente de **paso de mensajes** en **JavaScript**
- A los hilos en segundo plano se les denomina **WebWorkers** (o simplemente **workers**)
- Los **workers** no usan memoria compartida, sólo se comunican mediante **mensajes** entre hilos
- Un **worker** se puede iniciar desde el script principal (***main***) o desde **otro worker**

Concurrencia en JavaScript

- En **JavaScript** el código siempre se ejecuta en respuesta a un **evento** (carga de página, evento del usuario, temporizador, llamada AJAX…)
- Un **worker** funciona de forma similar, sólo ejecuta código ante la llegada de un mensaje desde el **script principal** o desde **otro worker**
- Un **worker** también puede enviar **mensajes de vuelta**
- Tutorial: http://anders.janmyr.com/2013/02/web-workers.html

Concurrencia en JavaScript

Ejemplo WebWorkers

Código de respuesta al recibir un mensaje del WebWorker

Ejecución del hilo (WebWorker) al enviarle un mensaje

Código del hilo (WebWorker)

Script principal

```
var worker = new Worker('doWork.js');

worker.addEventListener('message',
  function(e) {
     console.log('Worker said: ', e.data);
  }, false);

// Send data to the worker.
worker.postMessage('Hello World');
```

doWork.js

```
self.addEventListener('message', function(e)
{
  //do somethin useful
  self.postMessage(e.data);
}, false);
```

Concurrencia en JavaScript

- También existen **extensiones** (no estándar) **del lenguaje JavaScript** para concurrencia
  - **Intel RiverTrail**
    - Enfocado a la implementación de **algoritmos paralelos**
    - Disponible como **plugin de Firefox**
    - Quizás se estandarice en el futuro
    - https://github.com/RiverTrail/RiverTrail/wiki
    - http://www.infoq.com/news/2011/11/webgl-webcl-multicore-rivertrail

Concurrencia en JavaScript

- Actualmente se está popularizado el uso de JavaScript **fuera del navegador**
- El más popular es el **servidor web node.js** (basado en V8 de Chrome) que utiliza librerías (módulos)
- Módulos para **WebWorkers en node.js**
  - https://npmjs.org/package/webworker-threads
- Módulos para la **gestión directa de procesos en node.js**
  - http://nodejs.org/api/child_process.html
  - https://github.com/passcod/DynWorker

Concurrencia en los Lenguajes de Programación

- Introducción
- Concurrencia en Java
- Concurrencia en C/C++
- Concurrencia en JavaScript
- **Conclusiones**

Conclusiones

- Cada **plataforma de desarrollo** (lenguaje, sistema operativo y librerías) tiene sus propios mecanismos de **programación concurrente**
- Las plataformas que soportan el modelo de **memoria compartida**:
  - Suelen ofrecer **herramientas de bajo nivel similares:**
    - Gestión de hilos y herramientas de sincronización
  - Habitualmente disponen de **librerías de alto nivel:**
    - Estructuras de datos concurrentes, herramientas avanzadas de sincronización, otros modelos de programación concurrente…)

Conclusiones

- Existen plataformas muy extendidas que tienen un **soporte de concurrencia limitado** o únicamente ofrecen modelos de alto nivel **(JavaScript, Go, ActionScript/Flash…)**
- Esto se debe a que la **programación concurrente es compleja** y se ha **restringido** en ámbitos en los que se considera que **no es imprescindible** o se puede ofrecer un modelo más **sencillo**

Conclusiones

- La **programación concurrente** no es una disciplina nueva en informática
- Pero la llegada de los **procesadores multicore** ha obligado a las plataformas de desarrollo a ofrecer herramientas para **aprovechar** (y no desperdiciar) todo esa **potencia de cómputo**
- En los próximos años **evolucionarán** y se **popularizarán las herramientas de concurrencia** que ofrecen las plataformas