code
URJC

Programación Concurrente

Tema 1
Introducción

Micael Gallego
micael.gallego@urjc.es
@micael_gallego

Universidad Rey Juan Carlos

- **Bibliografía básica**
  - Palma, J.T. y otros. *Programación Concurrente*. Editorial Thomson, 2006
  - Yela, A.; Arroyo, F.; Fernandez, L. *Teoría y Práctica del Módulo de Programación Concurrente de la Asignatura de Programación II*. Departamento de Publicaciones de la E.U.I. de la Universidad Politécnica de Madrid, 1997

Programación concurrente

¿Qué es?

¿Dónde se usa?

¿Para qué se usa?

¿Cómo se usa?

PROGRAMACIÓN CONCURRENTE

Introducción a la Programación Concurrente

- **¿Qué es la programación concurrente?**
- ¿Dónde se usa la programación concurrente?
- ¿Para qué se usa la programación concurrente?
- ¿Cómo se programa concurrentemente?
- Conclusiones

¿Qué es la programación concurrente?

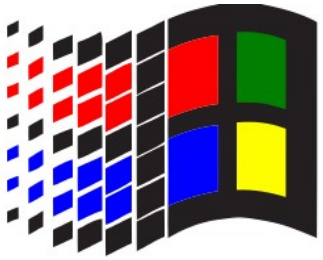
- **Concurrencia**
  - **RAE:**
    - 3. f. Coincidencia, concurso simultáneo de varias circunstancias.
  - **Wikcionario:**
    - 2. Acaecimiento o concurso de diversos sucesos o cosas en un mismo tiempo.
    - 7. Acción simultánea de dos ó mas.

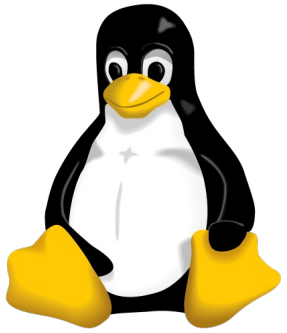
¿Qué es la programación concurrente?

- La **programación concurrente** se ocupa del estudio y desarrollo de programas que puedan realizar varias tareas "al mismo tiempo"
- La disciplina surge en los **años 60** cuando los **sistemas operativos**, gracias a mejoras hardware, comienzan a aprovechar el procesador ejecutando varios procesos de forma concurrente:
  - Ofrecer un entorno interactivo a **múltiples usuarios**
  - Aprovechar el procesador cuando un **proceso espera por una operación de entrada/salida**

¿Qué es la programación concurrente?

- Ejecución de varios programas a la vez en ordenadores personales (**multitarea**)
  - **MS-DOS** no permitía la ejecución de varios programas de forma concurrente
  - **Windows 1.0 (1985)** permitió la ejecución de varios programas a la vez pero de forma poco robusta (**mutitarea cooperativa**)
  - **Windows NT 3.1 (1993)** ofrece la primera ejecución simultánea de programas de forma segura (**multitarea preferente o *preemtive***)
  - **Linux (1991)** permite la ejecución simultánea de varios programas (como WinNT) y el acceso de varios usuarios
- Actualmente la gran mayoría de **dispositivos de computación** permiten la ejecución de procesos concurrentes





¿Qué es la programación concurrente?

- **Inicialmente** la concurrencia sólo la usaba el **sistema operativo** para la ejecución de varios programas de forma simultánea, pero internamente **los programas no podían** realizar varias tareas concurrentemente
- Los **lenguajes de programación y sus librerías** evolucionaron para que los desarrolladores pudieran implementar **programas concurrentes** (que realizan varias tareas simultáneamente)
- Algunos lenguajes de programación que **permiten el desarrollo de programas concurrentes** (casi todos los usados en la actualidad…)
  - **C/C++**, **Java, JavaScript,** Scala, Go, C#, Haskell, Erlang, OCAM…

Programa Secuencial y Proceso

- **Programa Secuencial**
  - Conjunto de declaraciones de datos e **instrucciones ejecutables**, escrito en un lenguaje de programación
  - Estas instrucciones deben ejecutarse una a continuación de otra, siguiendo una **secuencia** determinada por un algoritmo, para resolver un cierto problema
  - Un **programa** en Pascal es un programa **secuencial**

Programa Secuencial y Proceso

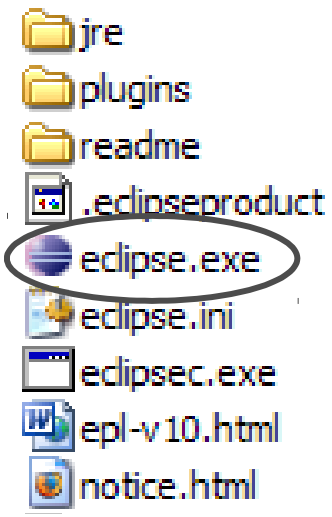
- **Proceso**
  - Es la **ejecución** de un programa secuencial en un sistema informático
  - Algunos programas pueden iniciarse por **segunda vez** antes de haber finalizado la ejecución previa
  - Esto hace que existan **varios procesos** de un **mismo programa secuencial** ejecutándose a la misma vez en un momento dado

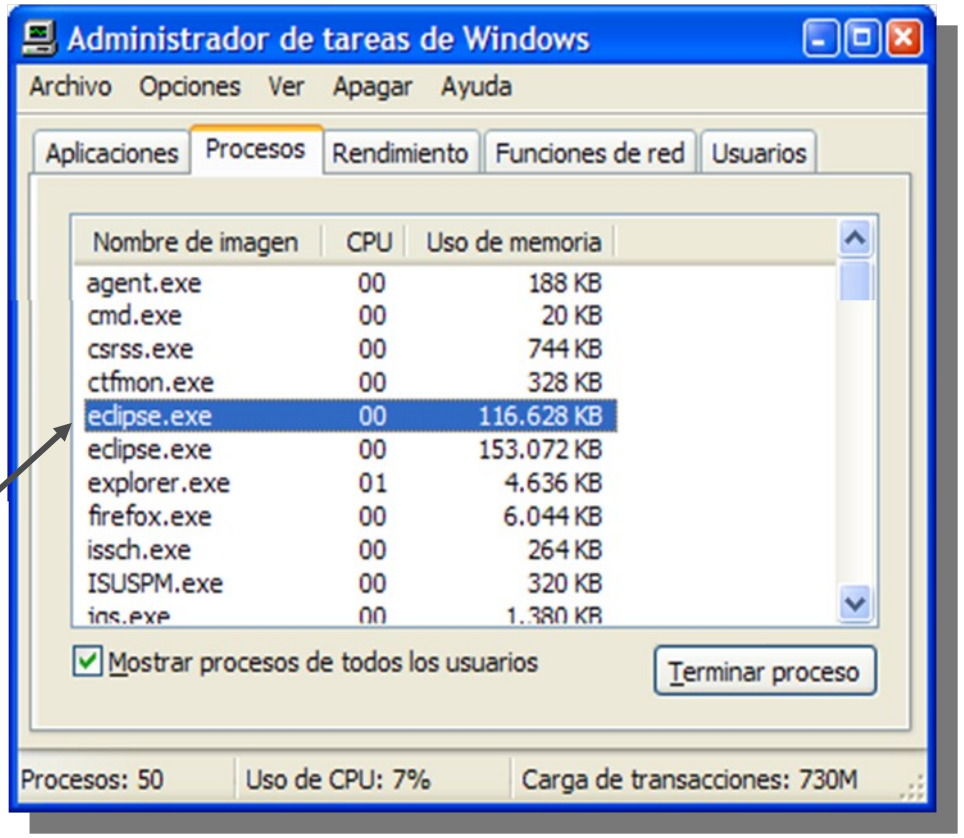
Programa Secuencial y Proceso

- **Programa Secuencial**
  - Representado en el disco duro como un fichero ejecutable (.exe en Windows)
- **Proceso**
  - Representado como un proceso en el **Administrador de tareas** (windows) o el **Monitor del sistema** (linux) cuando se arranca un fichero ejecutable

Programa Secuencial y Proceso

Programa Secuencial



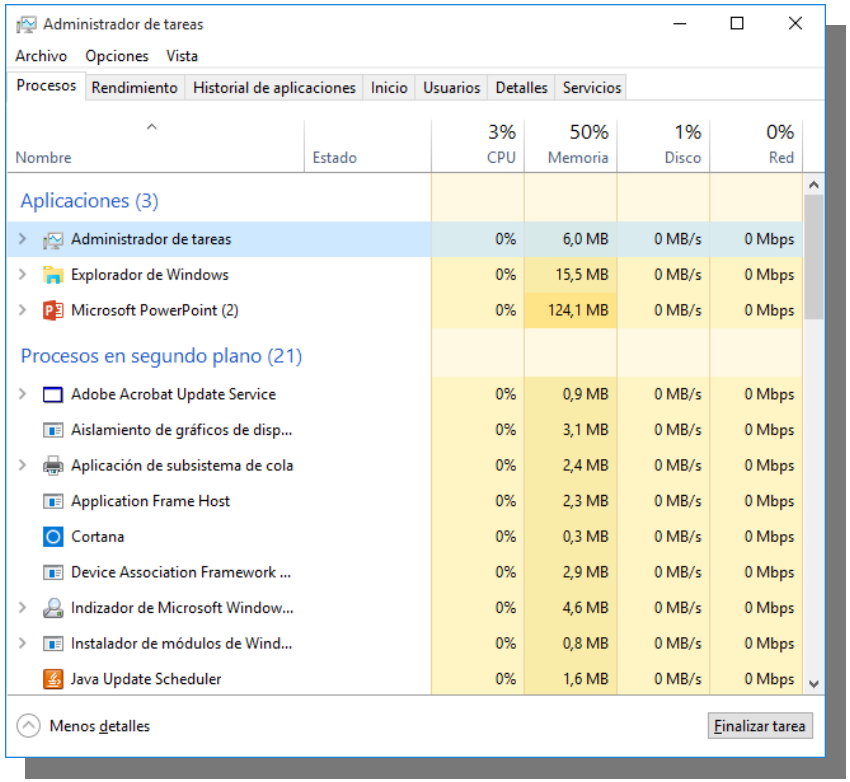


Administrador de tareas de Windows XP

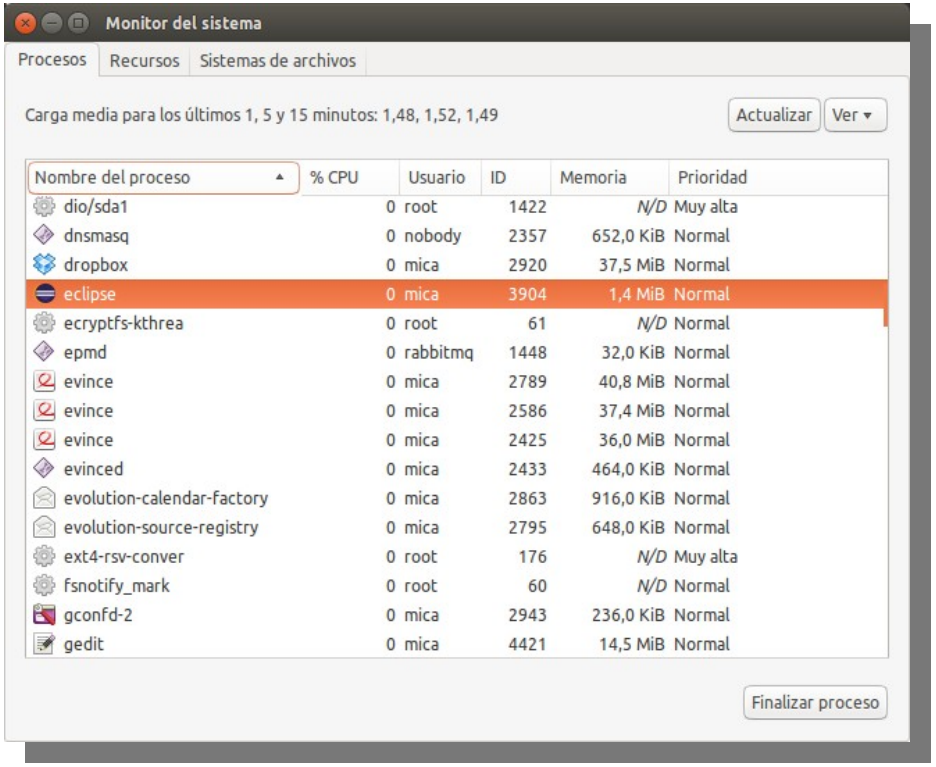
Proceso

Existen dos procesos concurrentes del programa "eclipse.exe"

Programa Secuencial y Proceso



Windows 10



Ubuntu 14.04

Programa Secuencial y Proceso

- **Procesos**
  - Cada proceso tiene su propio **espacio de memoria** y su propia **pila de ejecución** (*stack*)
  - Los procesos se pueden **comunicar entre sí**, pero son unidades **autónomas** e **independientes** (un fallo en un proceso no afecta a los demás procesos).
  - Están gestionados por el **sistema operativo**.

Programa Secuencial y Proceso

- **Programas concurrentes implementados con procesos del sistema operativo**
  - Los **primeros programas concurrentes** estaban formados por varios **procesos colaborando entre sí**
  - En **linux** es habitual que los procesos se comuniquen usando ***pipes*** (tuberías)
  - Se consideró que no era tan importante que los procesos fueran **independientes**, se prefería que **consumieran menos recursos**, que fuera más ligeros
  - Además, ciertas aplicaciones **necesitaban compartir la memoria para colaborar** más estrechamente entre sí

Programa Secuencial y Proceso

- **Hilos (*Threads*)**
  - Para el desarrollo de programas concurrentes se diseñaron unos **procesos ligeros** llamados **hilos** (***threads***)
  - Todos los **threads** de un programa se ejecutan en el mismo espacio de memoria (comparten toda la memoria)
  - Cada **thread** tiene su propia pila de ejecución (***stack***)

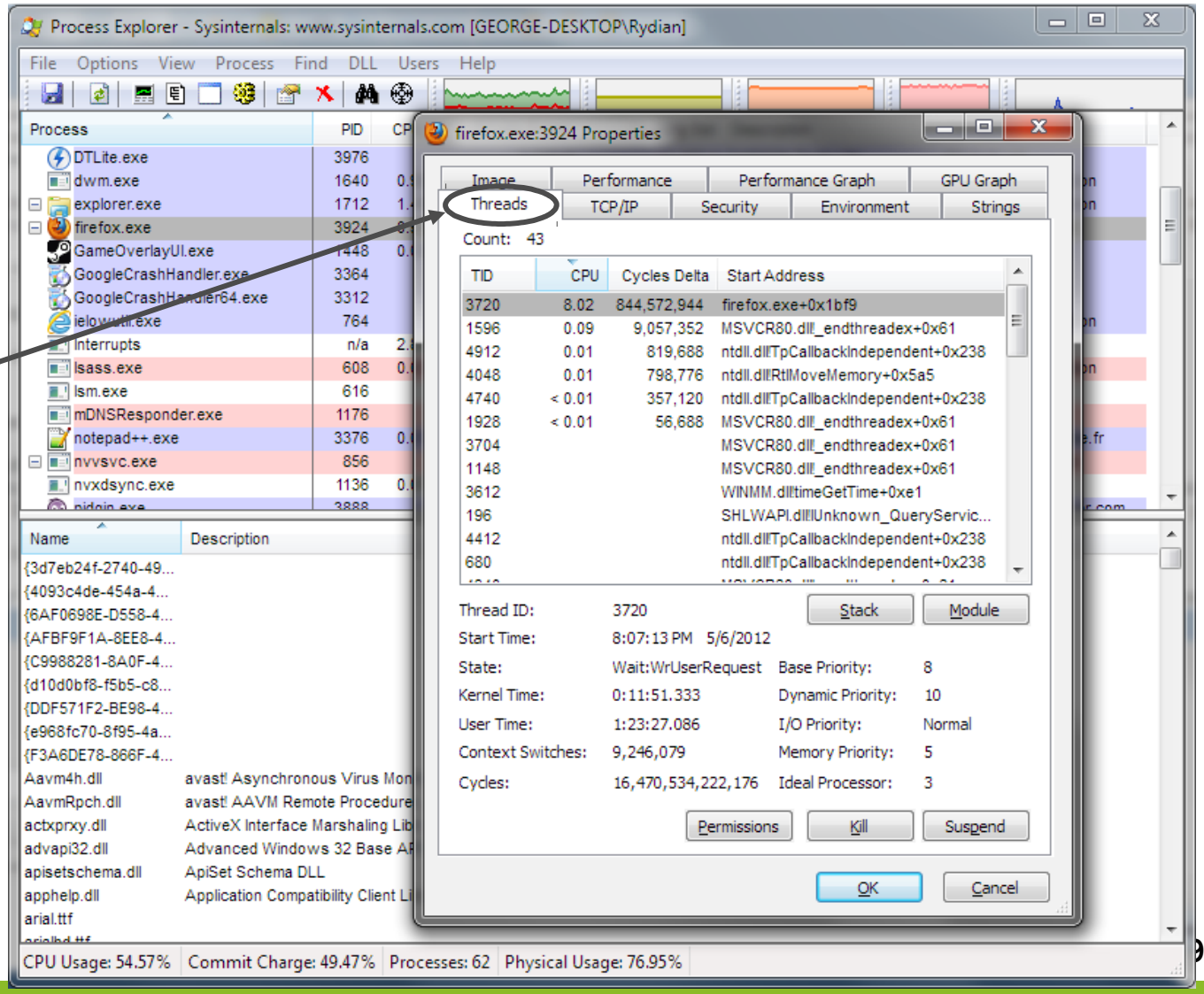
Programa Secuencial y Proceso

- **Hilos (*Threads*)**
  - Consumen **menos recursos** que los procesos "pesados" del sistema operativo (menos memoria y menor tiempo de creación)
  - Son **menos robustos** que los procesos pesados, ya que un fallo en un **thread** afecta a todo el programa y puede llegar provocar su **finalización inesperada**

Programa Secuencial y Proceso

code URJC

- **Explorador de procesos** en Windows con visualización de los **threads de cada proceso**



Programa Secuencial y Proceso

code URJC

- Puede darse concurrencia a dos niveles
  - A nivel de **Sistema Operativo**
    - **Programa Secuencial**: Fichero ejecutable
    - **Proceso**: Proceso que aparece en el sistema cuando se ejecuta un fichero ejecutable
    - **Todos los sistemas operativos actuales** permiten la concurrencia de procesos
  - A nivel de **Programa**
    - **Programa Secuencial**: Fragmento de código de un programa
    - **Proceso**: Ejecución independiente de un fragmento de código
    - La mayoría de las tecnologías de desarrollo (lenguajes+librerías) permiten la concurrencia con hilos

Programa Secuencial y Proceso

- **Concurrencia a nivel de Programa**
  - **Programa Secuencial**
    - Un conjunto de sentencias y declaración de variables
    - Normalmente está representado como las sentencias de un **método** con llamadas a otros métodos
  - **Proceso ligero o hilo de ejecución (*thread*)**
    - La **ejecución independiente** de un método (que pueden tener llamadas a otros métodos)
    - Existen programas que al ejecutarse sólo tienen **un hilo (secuencial)** y otros programas que tienen **varios hilos (concurrente)**

Programa Secuencial y Proceso

code URJC

Tipos de procesos dependiendo de su nivel

	Sistema Operativo	Programa
Programa Secuencial	Un fichero ejecutable (.exe en Windows)	Conjunto de sentencias (Método)
Proceso	Ejecución de un .exe Tiene memoria propia **Proceso pesado**	Ejecución de las sentencias de un método **Proceso ligero** **Hilo de ejecución** ***Thread***

Programa Secuencial y Proceso

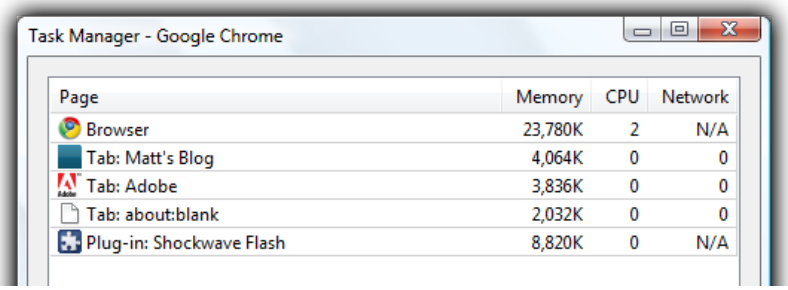
code URJC

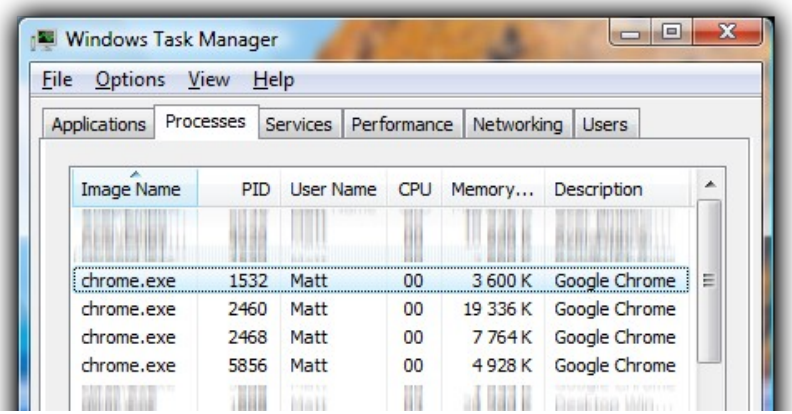
- **¿Qué es mejor: procesos o hilos?**
  - Los **procesos** consumen más **recursos** que los **hilos** (memoria y CPU)
  - Los **procesos** tardan más **tiempo en iniciarse** que los **hilos**
  - Los **procesos** son más **robustos** que los **hilos**: Si un proceso falla, el resto de procesos siguen su ejecución, pero si un hilo falla de forma no controlada, puede finalizar la ejecución del programa completo

Programa Secuencial y Proceso

- **Hilos y procesos en los navegadores web**
  - **Google Chrome:** Un proceso por cada pestaña. Si una pestaña falla, no afecta a las demás
  - **Mozilla Firefox:** Un único proceso para todas las pestañas (con varios hilos de ejecución). Si una pestaña falla, afecta a todas las demás
  - A partir de este mes se podrá usar un proceso por pestaña

https://wiki.mozilla.org/Electrolysis





Programa Secuencial y Proceso

- **Fibras**
  - Tanto los **procesos** como los **hilos** son gestionados por el **sistema operativo** con ayuda del **procesador**
  - Los **lenguajes concurrentes** que usan hilos delegan en el SO su **creación y administración**
  - Aunque los hilos son más ligeros que los procesos, **todavía siguen siendo bastante costosos**
  - Además, su número está **limitado a unos 10.000** en un servidor estándar por el consumo de memoria (aunque no estén haciendo nada)

Programa Secuencial y Proceso

- **Fibras**
  - Las fibras son **procesos ligeros** gestionados completamente por el **lenguaje de programación** (sin soporte del sistema operativo)
  - Son **mucho más ligeras que los hilos**:
    - No tardan prácticamente nada en crearse
    - Puede haber cientos de miles en un servidor estándar
  - No son **tan genéricas** como los hilos y están diseñadas para **ciertos casos específicos**

Programa Secuencial y Proceso

- **Fibras**
  - Se han puesto de moda en los últimos años debido al **lenguaje Go** diseñado por **Google** (en Go las fibras se llaman **gorutinas**)
  - **Java** no ofrece fibras de forma nativa, pero se pueden usar mediante librerías (Quasar de Parallel Universe)
  - Más información sobre fibras:

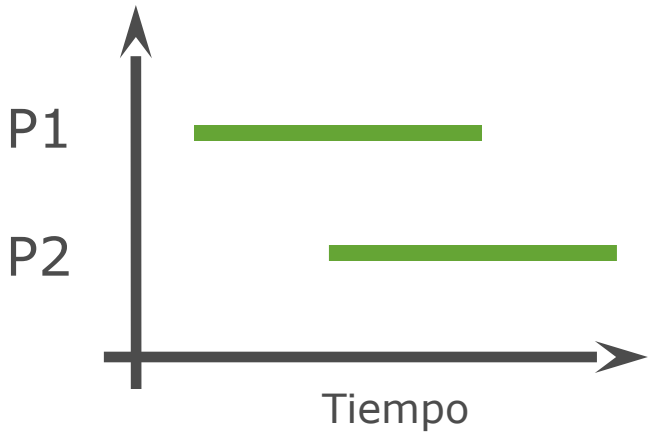
    https://en.wikipedia.org/wiki/Fiber_(computer_science)

Programa Secuencial y Proceso

- En el curso estudiaremos la **programación concurrente** a nivel de programa, usando **hilos de ejecución**
  - Es la **más usada** en el desarrollo de programas concurrentes
  - Está más **integrada** en los lenguajes de programación y las librerías
  - Es más **independiente** del Sistema Operativo, por tanto es más portable
  - Es más **sencilla** de aprender y de usar

Procesos Concurrentes

- **Procesos Concurrentes**
  - P1 y P2 se dice que son dos procesos concurrentes si la primera instrucción de uno de ellos se ejecuta entre la primera y la última instrucción del otro



Programa Concurrente

- **Programa Concurrente**
  - Conjunto de varios programas secuenciales, cuyos procesos pueden ejecutarse concurrentemente en un sistema informático
  - También llamado multi-hilo (***multi-threaded***)
  - En contraposición, a los programas que no son concurrentes se les denomina mono-hilo (***single-threaded***)

Sistema Concurrente

- **Sistema Concurrente**
  - Sistema Informático (HW+SW) en el que es posible ejecutar varios procesos concurrentemente
  - Prácticamente la **totalidad** de los sistemas informáticos actuales son **sistemas concurrentes**
  - Ejemplos: Servidores, PCs, teléfonos móviles, tabletas, dispositivos empotrados…

Introducción a la Programación Concurrente

- ¿Qué es la programación concurrente?
- **¿Dónde se usa la programación concurrente?**
- ¿Para qué se usa la programación concurrente?
- ¿Cómo se programa concurrentemente?
- Conclusiones

¿Dónde se usa la programación concurrente?

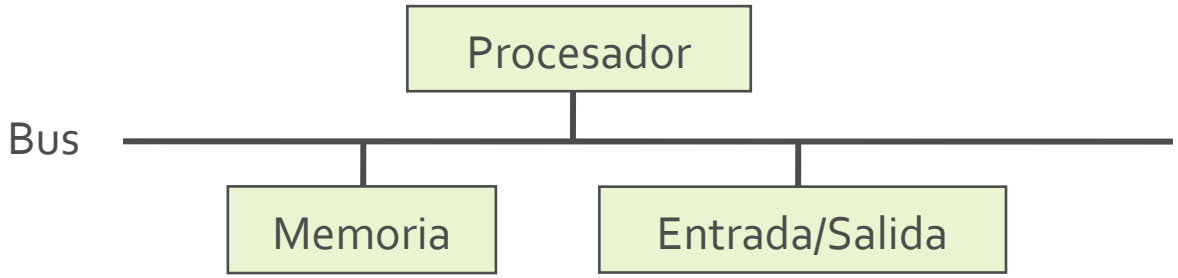
- **Arquitecturas de Sistemas Concurrentes**
  - **No** vamos a entrar en **detalles hardware** de implementación de arquitecturas
  - Pero vamos a dar una **visión general** de las diferentes **arquitecturas** de sistemas concurrentes
  - El objetivo es conocer los aspectos básicos para el **desarrollo de programas concurrentes**

¿Dónde se usa la programación concurrente?

- **Arquitecturas de Sistemas Concurrentes**
  - Sistemas monoprocesador
  - Sistemas multiprocesador
    - Muy acoplados
    - Poco acoplados
  - Sistemas híbridos

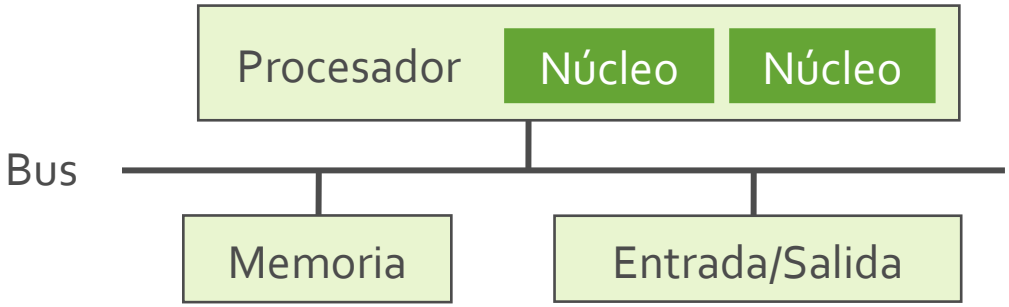
Arquitecturas Físicas

- **Sistemas monoprocesador**
  - Sistemas con un único procesador
  - La arquitectura de los PCs hace algunos años



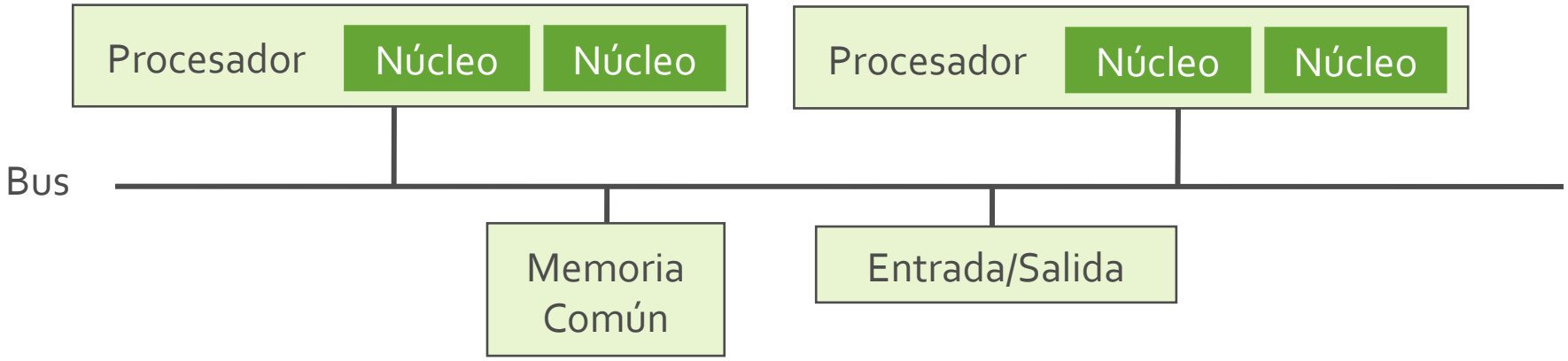
Arquitecturas Físicas

- **Sistemas multiprocesador muy acoplados**
  - Actualmente se están integrando varios procesadores en un único chip
  - A los procesadores internos se les denomina *cores* o núcleos de ejecución
  - Al chip se le denomina **procesador multicore**
  - La mayoría de los procesadores actuales en PCs, *Smartphones*, tabletas, etc. tienen esta arquitectura



Arquitecturas Físicas

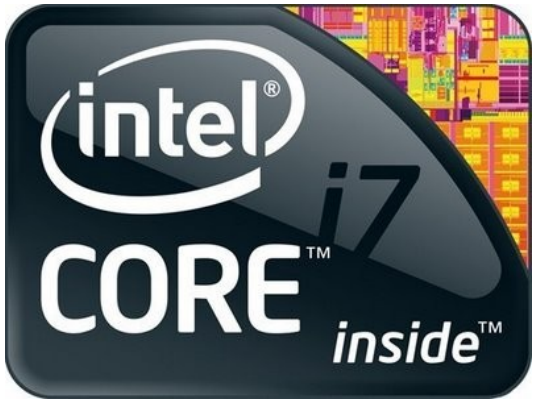
- **Sistemas multiprocesador muy acoplados**
  - Varios procesadores en la misma máquina
  - Usado en **servidores** y **estaciones de trabajo**
  - Conocidos como SMP (*Symmetric Multi-Processing*)



Arquitecturas Físicas

- **Sistemas multiprocesador muy acoplados**

Procesador Intel i7-5960x
8 cores



Titan X650 - Quad CPUs Intel Xeon E5-4600 V2 Series HPC Super Workstation up to **48 cores**

Arquitecturas Físicas

- **Sistemas multiprocesador muy acoplados**

IBM BladeCenter Express (Procesador 3.0 GHz 64-bit POWER7™ con **16 cores**)

Azul Compute Appliance Vega 3 Series 7300 con 16 procesadores Vega 3 con 54 cores cada uno. **864 cores** en total

Arquitecturas Físicas

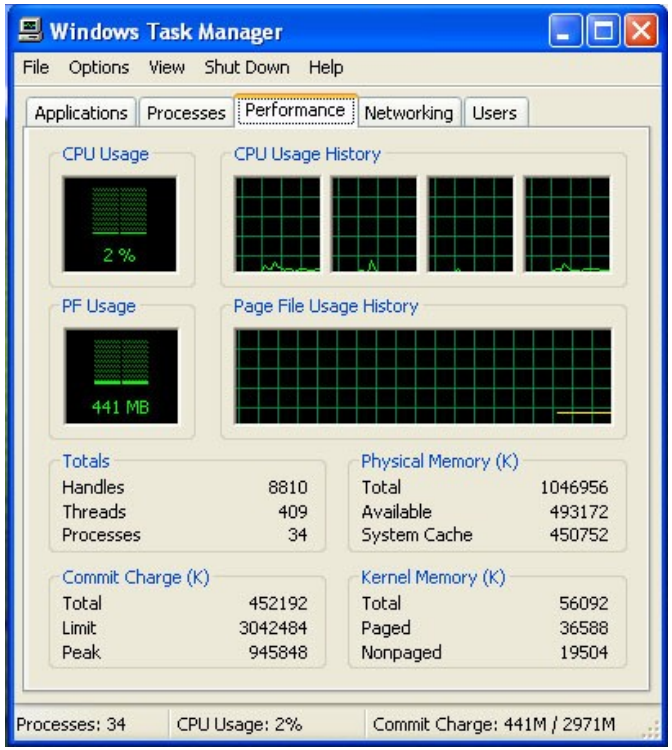
- **Sistemas multiprocesador muy acoplados**

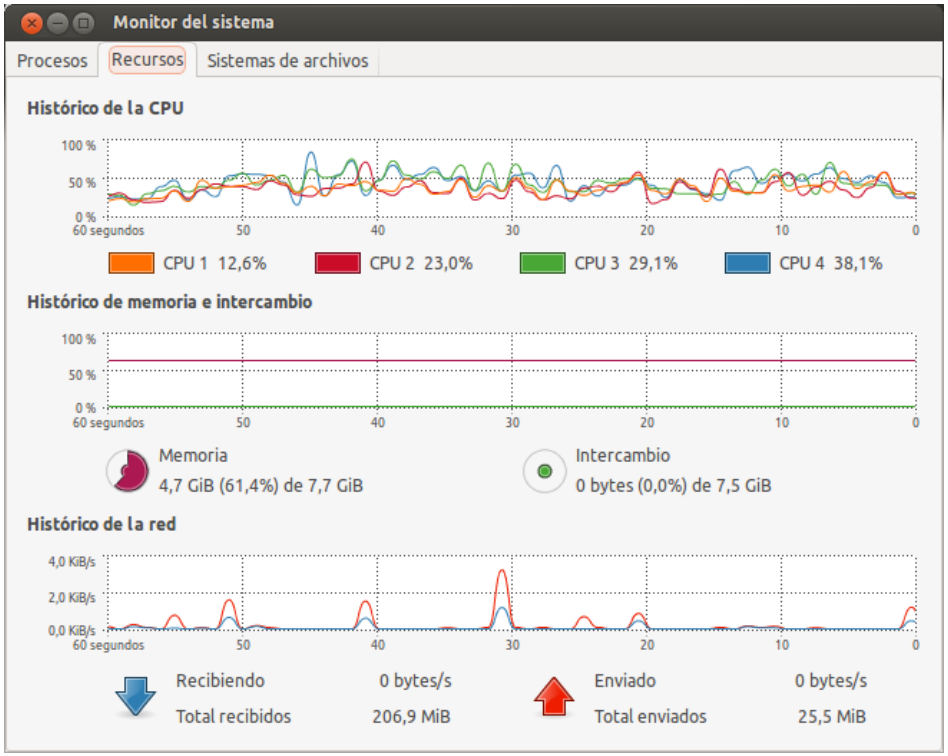
HTC ONE X 3G
Quad-core 1.5GHz (**4 cores**)

Playstation 4 con Procesador AMD Jaguar
8-core + ATI Radeon GPU con 18 cores

Arquitecturas Físicas

- **Sistemas multiprocesador muy acoplados**





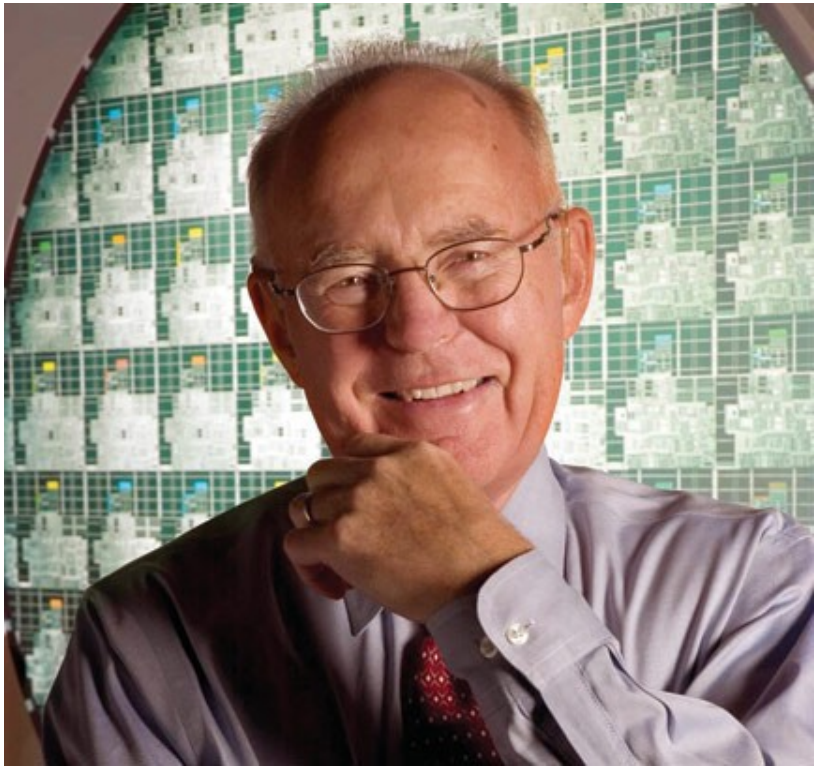
Arquitecturas Físicas

- **La Ley de Moore**

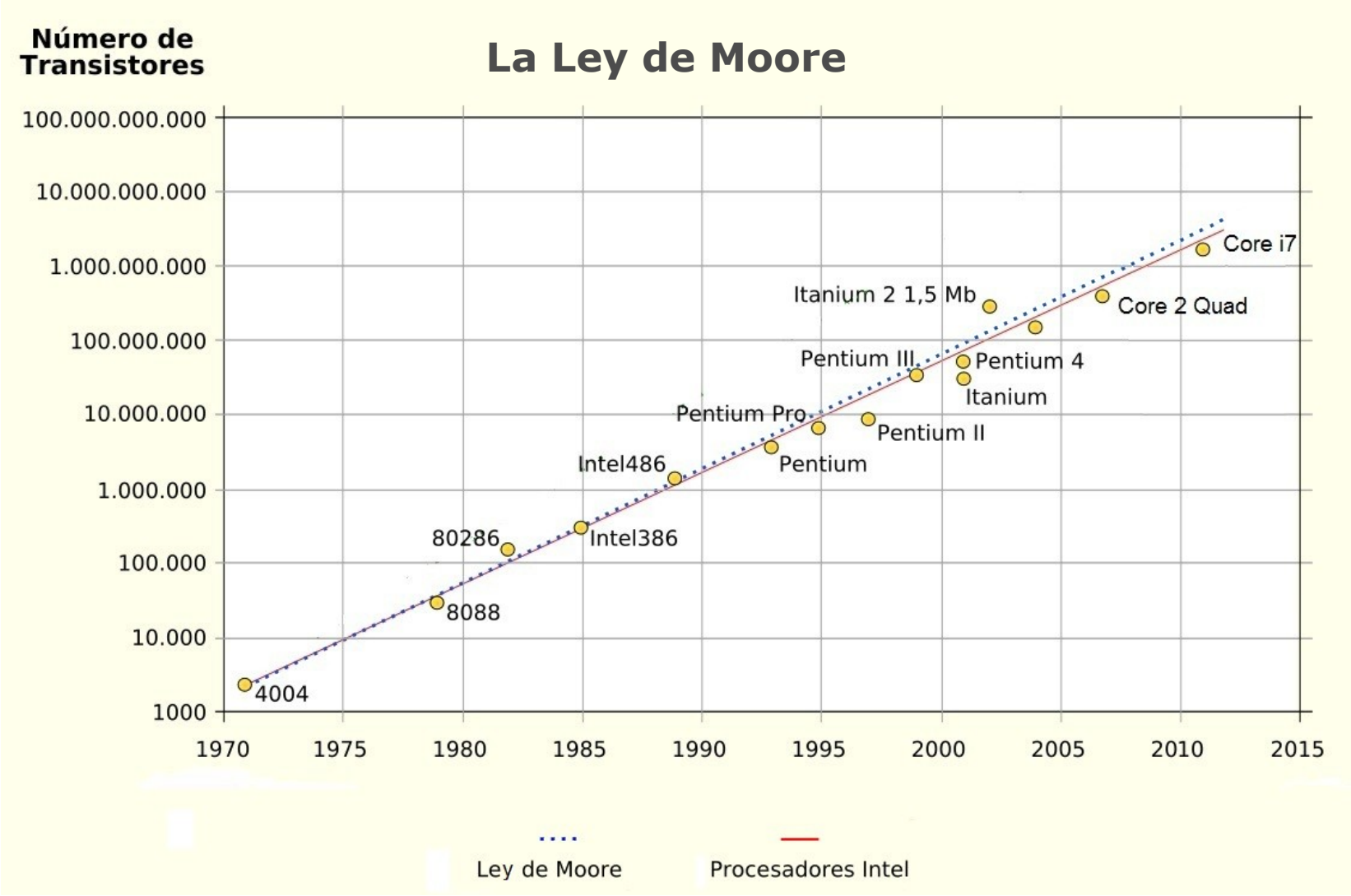
Es una ley empírica que expresa que aproximadamente cada **dos años se duplica el número de transistores** en un circuito integrado

Formulada por el cofundador de Intel, **Gordon E. Moore**, el 19 de abril de 1965.

Posteriormente fue adaptada para indicar que los **transistores se doblarían cada 18 meses**



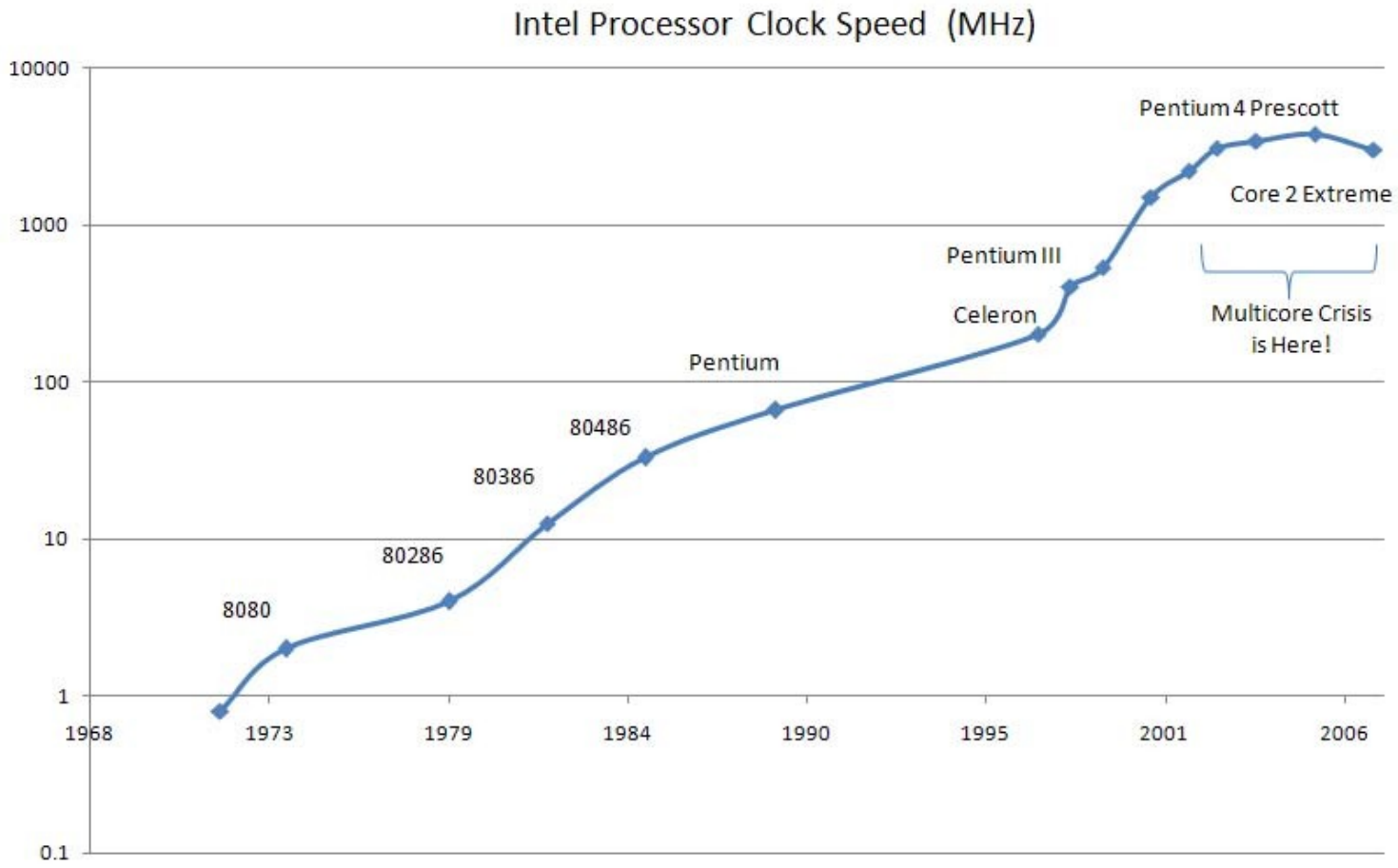
Arquitecturas Físicas



Arquitecturas Físicas

- **La Ley de Moore: el aumento de velocidad y la potencia de cómputo**
  - Durante años bastaba con **aumentar la frecuencia** de reloj de los chips para aumentar la **velocidad** de ejecución y por tanto la **potencia de cómputo**
  - Actualmente existen **impedimentos físicos** para seguir aumentando la frecuencia de reloj, principalmente debidos a la **potencia consumida** y la **disipación del calor**

Arquitecturas Físicas



Arquitecturas Físicas

- **La Ley de Moore: el aumento de velocidad y la potencia de cómputo**
  - Para aumentar la **potencia de cómputo** es necesario incluir **varias unidades de proceso o cores** en cada procesador
  - La tendencia es tener cada vez **más cores por procesador**

Free lunch is over!

http://www.gotw.ca/publications/concurrency-ddj.htm

Arquitecturas Físicas

- **Sistemas multiprocesador poco acoplados**
  - Varios dispositivos conectados en red
    - Dispositivos conectados en red para comunicarse
    - ***Clusters*** de ordenadores que actúan como un único sistema
  - También llamados sistema de **memoria distribuida**

Arquitecturas Físicas

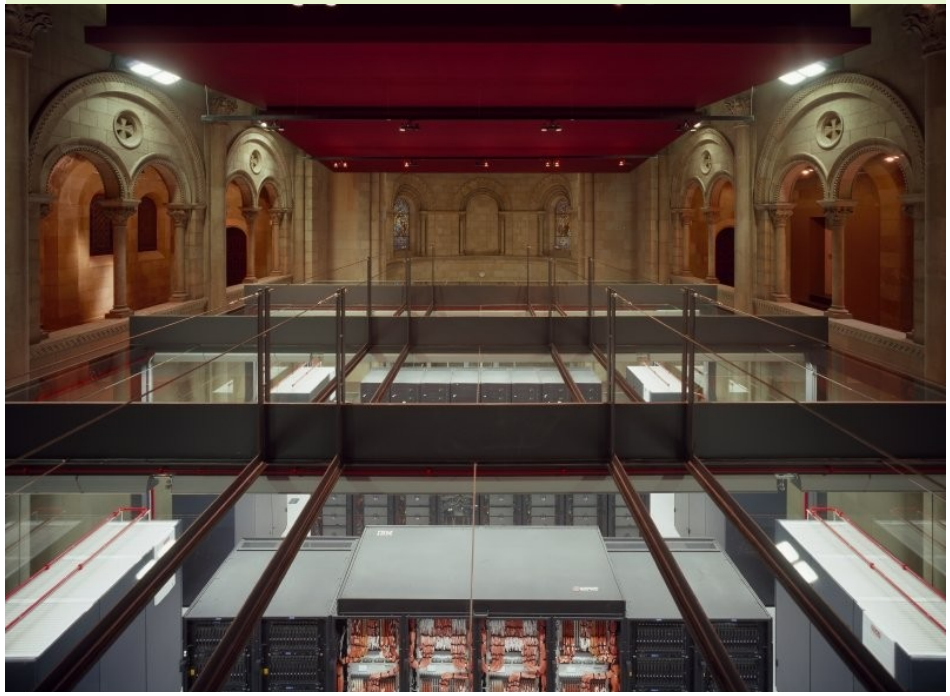
Clústers de ordenadores

Clúster homogéneo:
Varios equipos con el mismo sistema operativo y hardware

Clúster heterogéneo:
Varios equipos con distintos sistemas operativos y/o hardware

MareNostrum
Universidad Politécnica de Barcelona (UPC)

10.240 procesadores IBM PowerPC 970MP



http://es.wikipedia.org/wiki/Clúster_(informática)

Arquitecturas Físicas

- **Sistemas híbridos**
  - Dispositivos con varios procesadores, con varios ***cores*** cada uno, conectados en red formando un ***cluster***
  - Prácticamente la totalidad de los ***clusters*** actuales son **híbridos**

Arquitecturas Físicas

- **Modelos de Concurrencia**
  - Son diferentes **enfoques para el desarrollo** de programas concurrentes
  - Se dividen en **dos grandes bloques**, en función de la arquitectura física en la que se pueden usar
  - Con **arquitecturas híbridas**, se pueden combinar **varios modelos** en un mismo programa concurrente o usar el **mismo modelo**

Arquitecturas Físicas

- **Modelos de Concurrencia**
  - **Modelos de memoria compartida**
    - Los procesos pueden acceder a una **memoria común**
    - Existen variables compartidas que varios procesos pueden **leer** y **escribir**
  - **Modelos de paso de mensajes**
    - Los procesos se intercambian **mensajes** entre sí
    - Un proceso **envía mensaje** y otro proceso lo **recibe**

Arquitecturas Físicas

	Modelos de Concurrencia	
Arquitecturas de Sistemas Concurrentes	Paso de Mensajes	Memoria Compartida
Monoprocesador	√	√
Multiprocesador Muy Acoplado	√	√
Multiprocesador Poco Acoplado (Red)	√	?*

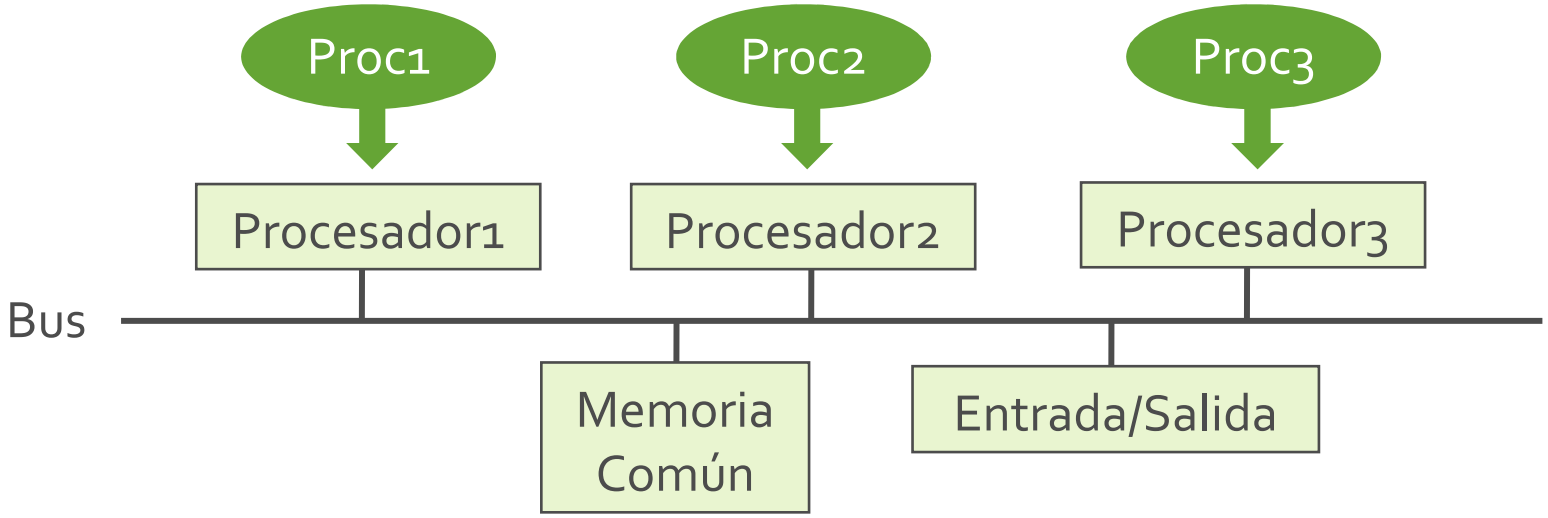
* Hay librerías que simulan la memoria compartida sobre red

Asignación de Procesos a Procesadores

- Un **procesador (core)** sólo puede ejecutar **un proceso a la vez**
- Si hay tantos **procesadores como procesos**, cada **proceso** se ejecuta en un **procesador**
- Lo más habitual es que haya **más procesos que procesadores**
- ¿Qué ocurre en ese caso?

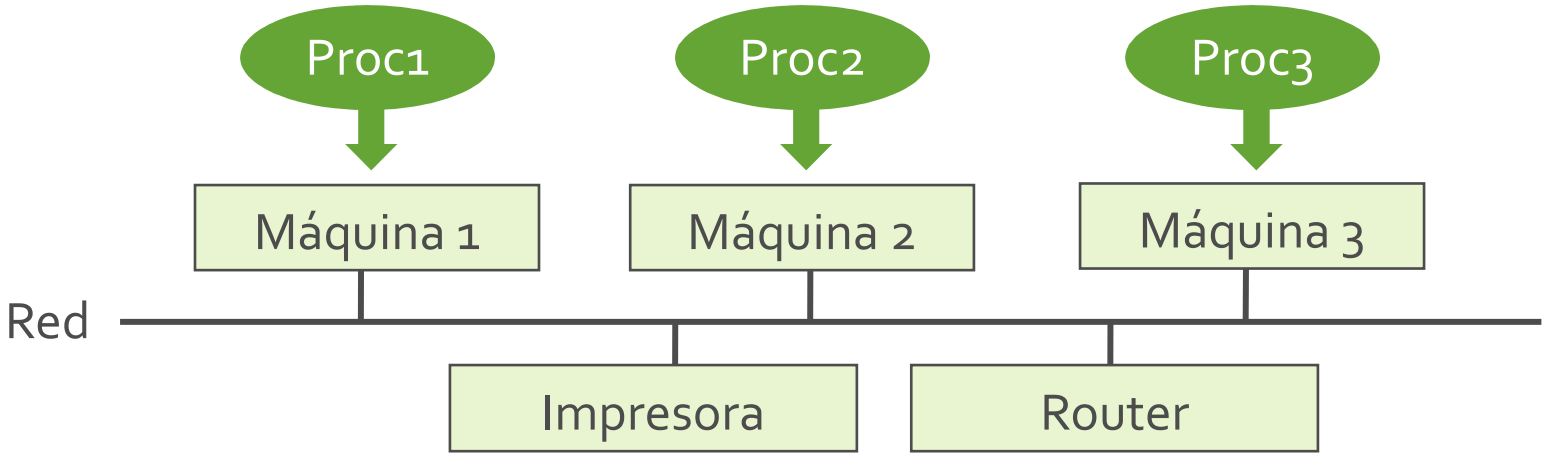
Asignación de Procesos a Procesadores

- **Multiproceso**
  - Cada proceso se ejecuta en su propio procesador en un sistema de memoria compartida



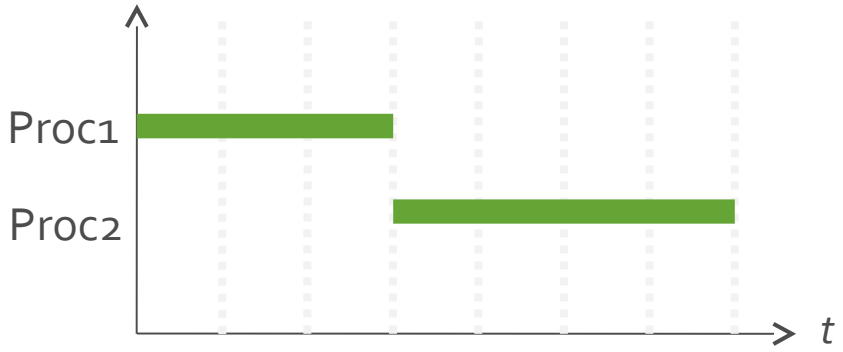
Asignación de Procesos a Procesadores

- **Procesamiento Distribuido**
  - Cada proceso se ejecuta en su propio procesador dentro de cada máquina de una red (Programa distribuido)

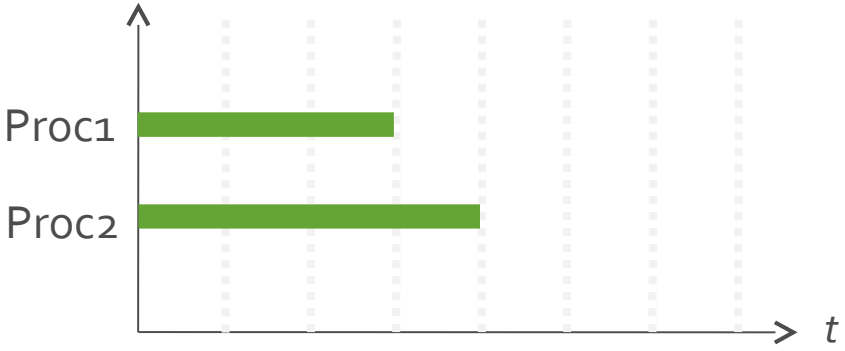


Asignación de Procesos a Procesadores

- **Paralelismo Real**
  - Se obtiene cuando hay un procesador por cada proceso (**Mutiproceso o procesamiento distribuido)**
  - Se consigue un aumento de la velocidad de ejecución del programa con respecto a la ejecución secuencial



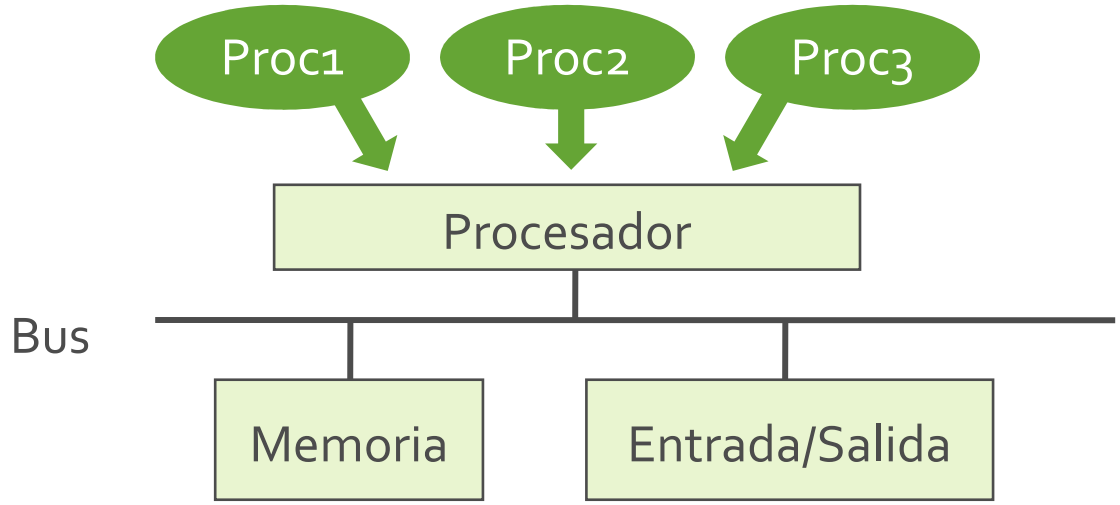
Ejecución Secuencial



Paralelismo Real

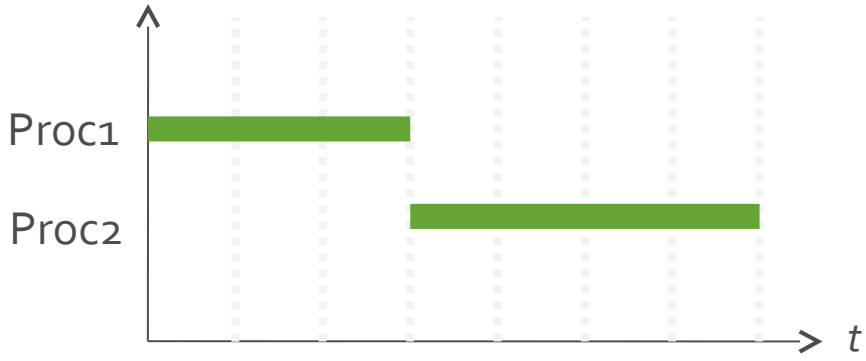
Asignación de Procesos a Procesadores

- **Multiprogramación**
  - Varios procesos se ejecutan en el **mismo** procesador
  - Cada proceso se ejecuta durante un tiempo y luego pasa a ejecutarse el siguiente proceso (**Compartición de tiempo**)

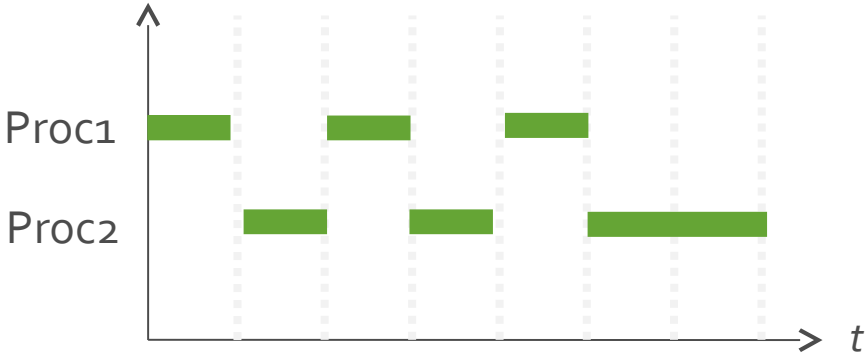


Asignación de Procesos a Procesadores

- **Paralelismo Simulado o Pseudoparalelismo**
  - Se obtiene cuando varios procesos comparten el mismo procesador (**Multiprogramación**)
  - El usuario percibe una sensación de paralelismo real
  - Aparentemente no se consigue un aumento de la velocidad de ejecución del programa con respecto a la ejecución secuencial



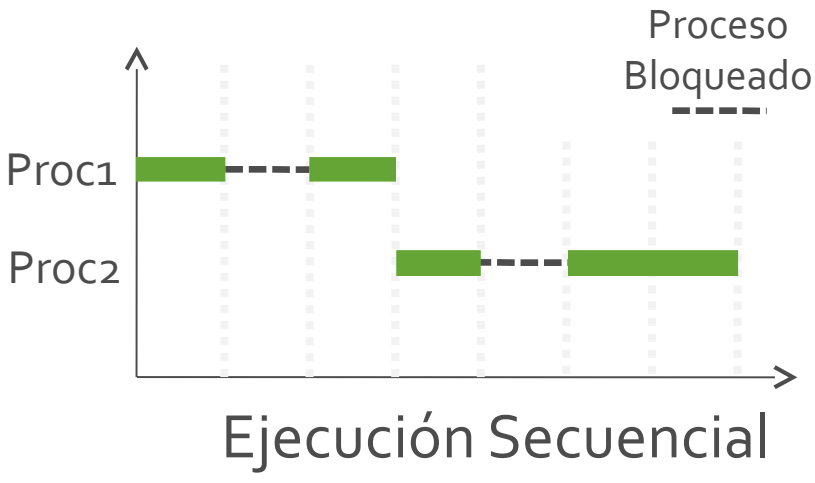
Ejecución Secuencial



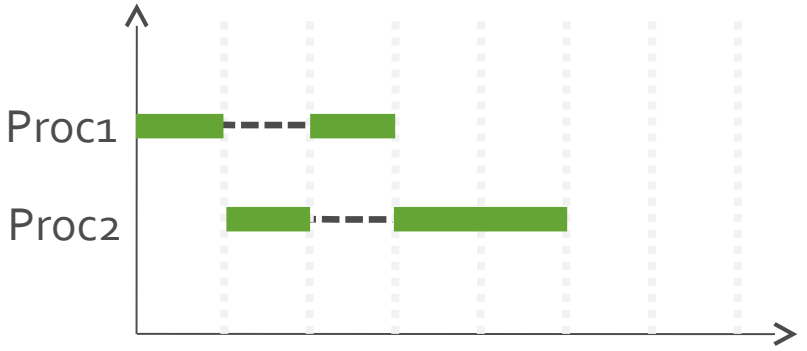
Paralelismo Simulado

Asignación de Procesos a Procesadores

- **Paralelismo Simulado o Pseudoparalelismo**
  - Cuando los procesos esperan por entrada/salida (IO), el procesador se puede aprovechar para otro proceso
  - En este caso se reduce el tiempo de ejecución del conjunto de procesos
  - Se obtiene una **reducción del tiempo de ejecución** incluso con un procesador



Ejecución Secuencial



Paralelismo Simulado

Asignación de Procesos a Procesadores

- **Asignación de Procesos a Procesadores**

Cada procesador ejecuta un proceso
- Multiproceso
- Procesamiento Distribuido

→ **Paralelismo Real**
Aumenta la velocidad de ejecución

Cada procesador ejecuta varios procesos de forma intercalada
- Multiprogramación

→ **Paralelismo Simulado**
Aumenta la velocidad de ejecución si los procesos usan Instrucciones de IO

En un sistema informático lo más habitual es que se use la **Multiprogramación** aunque se disponga de varios procesadores, porque en la mayoría de las ocasiones **hay más procesos que procesadores**

Asignación de Procesos a Procesadores

code URJC

- Hay **muchas arquitecturas** diferentes en las que se puede ejecutar un programa **concurrente**
- Para que los programas sean **portables** se diseñan e implementan sin tener en cuenta si existe **multiproceso** o **multiprogramación**
- Se usan simplificaciones o **abstracciones** que ayudan a centrar la atención en los **procesos y sus relaciones**, y nos evitan pensar en la arquitectura del sistema
- Las abstracciones que se presentan se publicaron en el libro: **Principles of Concurrent and Distributed Programming, 2nd ed. 2006 Addison-Wesley. Ben-Ari.**

Asignación de Procesos a Procesadores

code URJC

1ª Abstracción de la Programación Concurrente
Se considera que cada proceso se ejecuta en su propio procesador

- Esta abstracción permite tener en cuenta únicamente las **interacciones entre los procesos**
- No nos tenemos que preocupar de si hay **paralelismo real** o **paralelismo simulado**

Asignación de Procesos a Procesadores

code URJC

2ª Abstracción de la Programación Concurrente

Se ignoran las velocidades relativas de cada proceso, lo que posibilita considerar sólo las secuencias de instrucciones que se ejecutan

- Tenemos que pensar lo que ocurriría con nuestro programa si el procesador de cada **proceso** tuviese la **misma velocidad**
- También tenemos que pensar que pasaría si un procesador fuese **muy lento** y otro procesador fuese **muy rápido**
- Pensar **en todas las posibles situaciones** permite que nuestros programas concurrentes funcionen correctamente **en cualquier tipo de arquitectura**

Introducción a la Programación Concurrente

- ¿Qué es la programación concurrente?
- ¿Dónde se usa la programación concurrente?
- **¿Para qué se usa la programación concurrente?**
- ¿Cómo se programa concurrentemente?
- Conclusiones

¿Para qué se usa la programación concurrente?

- **¿Cuándo usar la concurrencia?**
  - Existen muchas situaciones en las que un programa concurrente es **mejor** que uno secuencial
    - Más fácil de implementar
    - Más rápido en ejecutarse
    - Consumirá menos recursos (memoria, CPU…)
  - En otros casos, es **imprescindible** que el programa sea concurrente o no se podrá implementar

¿Para qué se usa la programación concurrente?

- **¿Cuándo usar concurrencia?**
  - **Imprescindible**
    - 1) Servidores que atienden varios usuarios / peticiones simultáneamente
    - 2) Aplicaciones interactivas (que realizan tareas en segundo plano)
  - **Conveniente**
    - 3) Aumentar la velocidad cuando hay entrada y salida con un único procesador
    - 4) Aumentar la velocidad de los programas usando varios procesadores en paralelo

¿Para qué se usa la programación concurrente?

- **1) Servidores que atienden varios usuarios / peticiones simultáneamente**
  - **Servidor Web:** Atiende a cientos de peticiones simultáneamente
  - **Servidores de WhatsApp:** Atiende cientos de usuarios conectados
  - **Servidores de juegos on-line:** Múltiples partidas simultáneas

¿Para qué se usa la programación concurrente?

- **2) Aplicaciones interactivas (que realizan tareas en segundo plano)**
  - El programa realiza cálculos en segundo plano mientras que el usuario sigue trabajando
  - Ejemplos:
    - Eclipse compila en segundo plano mientras se sigue editando
    - Un navegador carga una página web mientras el usuario navega por otras

¿Para qué se usa la programación concurrente?

- **3) Aumentar la velocidad cuando hay entrada y salida en sistemas con un procesador**
  - Algunas aplicaciones que mayoritariamente **realizan operaciones en el procesador** pero no realizan entrada/salida: diseño 3D, algoritmos de optimización, etc.
  - En estas aplicaciones, **si sólo hay un único procesador**, es más eficiente que el programa sea **secuencial**
  - En las aplicaciones que tienen **entrada/salida** (acceso a disco, red), es más eficiente que sean concurrentes incluso con **un único procesador**)

¿Para qué se usa la programación concurrente?

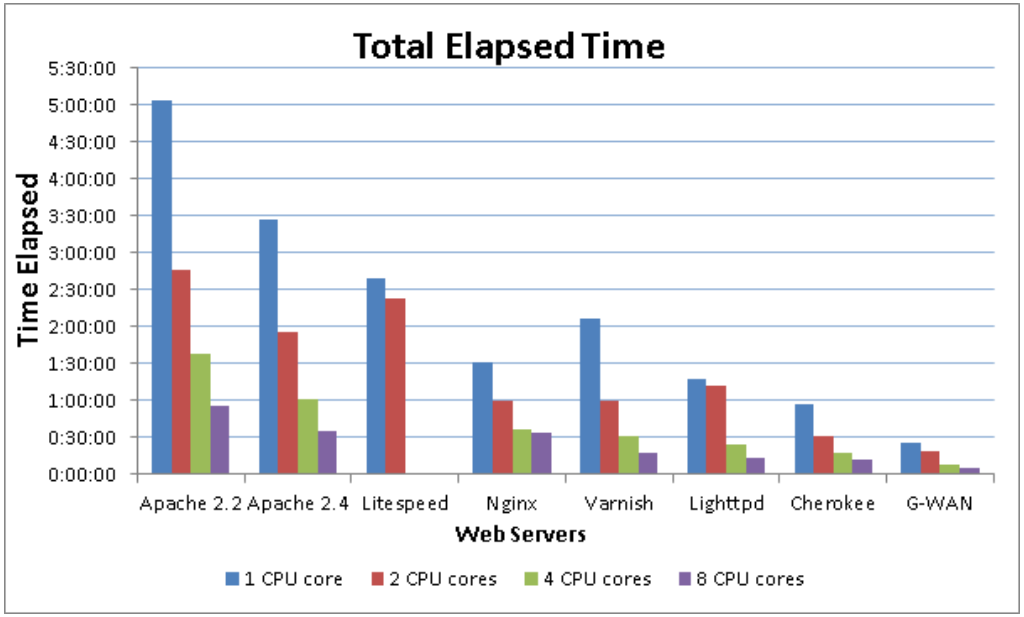
- **4) Aumentar la velocidad de los programas usando varios procesadores en paralelo**
  - Los procesadores ya **no van a aumentar de velocidad**
  - Los nuevos **procesadores** tendrán cada vez **más *cores***
  - Las aplicaciones tendrán que ser concurrentes (***multithreaded***) para **aprovechar toda la potencia** de cómputo disponible
  - A medida que pase el tiempo, cada vez será **más crítico** porque habrá más cantidad de ***cores***

¿Para qué se usa la programación concurrente?

- **4) Aumentar la velocidad de los programas usando varios procesadores en paralelo**
  - **El compilador de java oficial de Oracle** es ***single-threaded*** (**secuencial**) y por tanto no aumenta de velocidad con varios cores
    - http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6629150
  - **El compildor de java incluido en Eclipse** es ***multi-threaded*** (**concurrente**) desde 2008 y por tanto si aumenta de velocidad con varios cores
    - https://bugs.eclipse.org/bugs/show_bug.cgi?id=142126

¿Para qué se usa la programación concurrente?

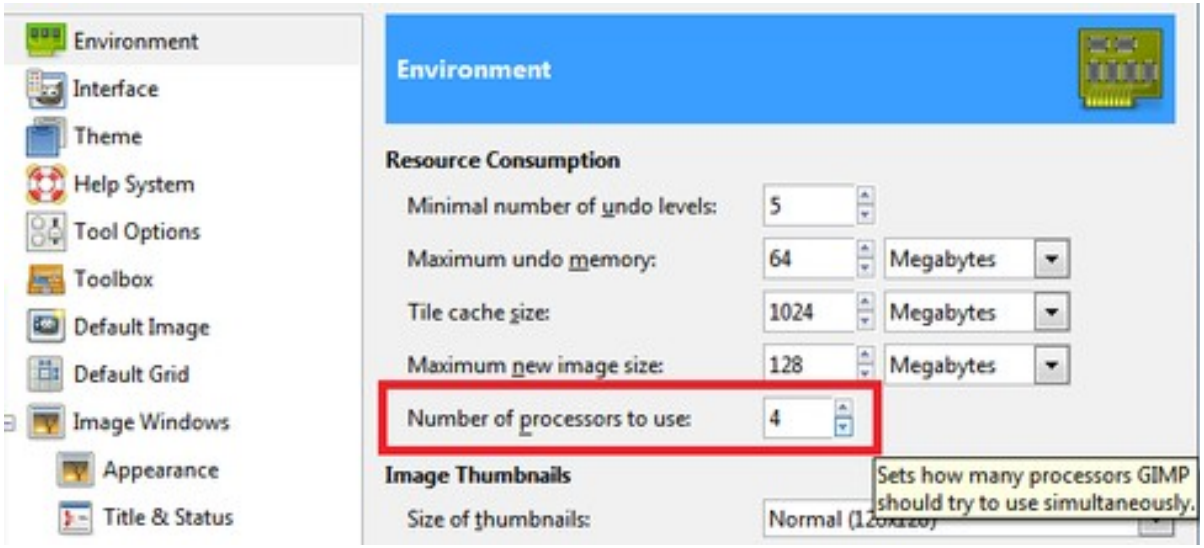
- **4) Aumentar la velocidad de los programas usando varios procesadores en paralelo**
  - Comparativa de servidores web con diferentes cores



https://www.rootusers.com/web-server-performance-benchmark/

¿Para qué se usa la programación concurrente?

- **4) Aumentar la velocidad de los programas usando varios procesadores en paralelo**
  - **Gimp:** Puede usar varios procesadores/cores a la vez para aplicar efectos a una imagen



Introducción a la Programación Concurrente

- ¿Qué es la programación concurrente?
- ¿Dónde se usa la programación concurrente?
- ¿Para qué se usa la programación concurrente?
- **¿Cómo se programa concurrentemente?**
- Conclusiones

¿Cómo se programa concurrentemente?

- Existen **muchas formas diferentes** de implementar un programa concurrente
- Depende del **lenguaje** de programación y la **librería** que estemos usando
- Existen **modelos teóricos** que luego se concretan de formas diferentes en los diferentes lenguajes y librerías
- Se les conoce como **"Modelos de concurrencia"**

¿Cómo se programa concurrentemente?

- **Modelos de Concurrencia**
  - Existen dos grandes familias: **Modelo de Memoria compartida** y **Modelo de Paso de Mensajes**
  - Hay **varios modelos diferentes** dentro de cada familia
  - No hay un consenso sobre cual es el **mejor de todos** (como ocurre con los lenguajes de programación)

¿Cómo se programa concurrentemente?

- **Niveles de abstracción de los modelos**
  - **Bajo nivel**
    - Cercanos a la funcionalidad ofrecida por el **hardware**
  - **Alto nivel**
    - Inspirados en **modelos matemáticos**
    - Normalmente cualquier modelo de **alto nivel** se puede implementar **sobre cualquier otro** modelo de más **bajo nivel**

¿Cómo se programa concurrentemente?

- **La elección de un modelo u otro está determinado por:**
  - **Tipo de programa a implementar**
    - Las aplicaciones de **red** pueden tener distintas necesidades que las aplicaciones **interactivas**
  - **Lenguaje de programación utilizado y sus librerías**
    - Hay lenguajes en los que sólo se puede usar un modelo de **alto nivel** (JavaScript)
    - Los lenguajes que ofrecen modelos de **bajo nivel** suelen permitir algunos de los modelos de alto nivel con librerías (Java)
  - **Arquitectura física**
    - Los sistemas multiprocesador **poco acoplados no** pueden usar modelos de **memoria compartida** (pero al contrario si)

¿Cómo se programa concurrentemente?

- **Modelos de concurrencia más usados**
  - Hilos y cerrojos
  - Actores
  - Comunicando procesos secuenciales (CSP)
  - Programación Funcional
  - Memoria software transaccional (STM)
  - … (hay muchos más)

Modelos de concurrencia

- **Hilos y cerrojos**
  - Modelo de **memoria compartida**
  - Es el modelo de más **bajo nivel**, el más cercano a las primitivas ofrecidas por el **hardware**
  - Suele estar disponible en **lenguajes imperativos** (estructurados y orientados a objetos)
  - Es el modelo **más usado y más potente, pero también el más complejo de programar**

Lenguajes: C/C++, Java, Ruby, Python, C#...

Modelos de concurrencia

- **Hilos y cerrojos**
  - Se le conoce con muchos nombres en la literatura:
    - **Estado mutable compartido**
    - Memoria compartida protegida por **cerrojos (*locks*)**
    - Programación con **hilos**
    - Programación basada en **cerrojos**
  - Aunque hay más modelos de memoria compartida, como es el más usado, hay veces que se le llama directamente como **Memoria compartida**

Modelos de concurrencia

- **Actores**
  - Modelo de **paso de mensajes**
  - Esta definido por un **modelo matemático**
  - Los **actores** son las primitivas concurrentes de procesamiento que se comunican enviando **mensajes** a otros actores
  - Dependiendo del **mensaje que reciba**, un actor puede:
    - Crear más actores,
    - Enviar más mensajes
    - Cambiar su estado

Lenguajes: Erlang, Occam, Oz, Scala, …
Librerías: Akka (Java), Quasar (Java), Actor-CPP (C++)…

Modelos de concurrencia

- **Comunicando procesos secuenciales (CSP)**
  - Modelo de **paso de mensajes**
  - Es un lenguaje formal basado en el **algebra o cálculo de procesos**
  - Los procesos se comunican enviándose mensajes a través de canales
  - El envío de mensajes es parecido al **modelo de actores**, pero en CSP un proceso puede tener varios canales pero en el modelo de actores sólo uno (de recepción)

Lenguajes: Go, Occam …
Librerías: Quasar (Java), JCSP (Java), C++CSP2 (C++)

Modelos de concurrencia

- **Programación Funcional**
  - Modelo de **memoria compartida**
  - Está basado en que en la programación funcional la **información no se modifica (es inmutable)**, por eso se puede compartir entre varios hilos sin problemas
  - Todo el **procesamiento** se hace en base a **funciones** que generan valores de **salida** partiendo de los parámetros de **entrada**
  - Como un programa está descrito de forma **declarativa** como una **composición de funciones**, es **paralelizarble**

Lenguajes: Parallel Haskell…
Librerías: Streams de Java 8 …

Modelos de concurrencia

- **Memoria software transaccional (STM)**
  - Modelo de **memoria compartida**
  - Aplica el concepto de **transacciones de bases de datos** a la memoria compartida entre procesos
  - Dentro de una transacción un proceso puede **leer y escribir** la **memoria compartida**. Todas las lecturas y escrituras ocurren de forma **atómica**, y otros procesos **no pueden ver estados intermedios**.

Lenguajes: Clojure, Haskell…
Librerías: Multiverse (Java), Durus (Python), TinySTM (C/C++)…

Modelos de concurrencia

code URJC

Familia	Modelo	Lenguajes	Librerías
Modelo de Memoria compartida	Hilos y Cerrojos	C/C++, Java, Ruby, Python, C#, Scala, …	
	Memoria Software Transaccional (STM)	Clojure, Haskell…	Multiverse (Java), Durus (Python), TinySTM (C/C++)…
	Programación funcional	Parallel Haskell…	Streams en Java 8
Modelo de Paso de mensajes	Actores	Erlang, Occam, Oz, Scala, …	Akka (Java), Quasar (Java), Actor-CPP (C++)…
	CSP	Go, Occam …	JCSP (Java), Quasar (Java), C++CSP2 (C++)

Modelos de concurrencia

code URJC

- En este **curso** estudiaremos los modelos más populares de cada familia:
  - Modelo de memoria compartida: **Hilos y cerrojos**
  - Modelo de paso de mensajes: **Actores**

- Para **aprender los conceptos básicos** de la programación concurrente sin entrar en detalles particulares de un lenguaje concreto se usará la librería **SimpleConcurrent** para Java (Parte I)

Modelos de concurrencia

- Posteriormente se estudiarán tecnologías profesionales (Parte 2):
  - En el **tema 4** se dará una **visión general** de la programación concurrente en los **lenguajes de programación** más utilizados en la actualidad (**Java, C/C++ y JavaScript**)
  - En el **tema 5** se abordará en **detalle** la **programación concurrente con Java** en ambos modelos

Introducción a la Programación Concurrente

code URJC

- ¿Qué es la programación concurrente?
- ¿Dónde se usa la programación concurrente?
- ¿Para qué se usa la programación concurrente?
- ¿Cómo se programa concurrentemente?
- **Conclusiones**

Conclusiones

- La **Programación Concurrente** es una técnica que todo **desarrollador debe dominar** para el desarrollo de programas
- Prácticamente cualquier **programa "real"** usa la **concurrencia** de una forma u otra
  - Aplicaciones con **interfaz gráfico de usuario** (escritorio, móviles…)
  - Aplicaciones **web** (un hilo de ejecución por petición o programación reactiva)

Conclusiones

- En los **próximos años** es previsible que los procesadores tengan cada vez **más núcleos de procesamiento**
- Esa será la única forma de **aumentar la potencia de cómputo** de los dispositivos al no poder aumentar la velocidad de ejecución
- Para aprovecharla, los programas secuenciales existentes se tienen que **reimplementar** y los nuevos se tienen que **diseñar** dividiendo el procesamiento en **tareas** que puedan **ejecutarse en paralelo**

Conclusiones

- Pese a que existen **muchos modelos** de programación concurrente, el más utilizado sigue siendo el modelo de bajo nivel de **estado compartido con cerrojos**
- Es el más **complejo** de usar correctamente, pero es el más **potente**
- Es conveniente **dominar otros modelos de concurrencia** porque pueden ser más adecuados para ciertos tipos de programas (actores, CSP, funcional, etc…)

Conclusiones

- Desarrollar programas concurrentes es **mucho más complejo** que desarrollar programas secuenciales
  - Son más **difíciles** de programar
  - Es más **difícil** comprobar que su funcionamiento es el **correcto**, porque hay que considerar cómo se comportan con **uno o varios procesadores**, con la misma o diferentes **velocidades**
  - Que **una ejecución** sea **correcta** con unos datos de entrada no implica que **todas las ejecuciones** sean **correctas** con esos mismos datos (como ocurre con los programas secuenciales).
  - Hay **ejecuciones problemáticas** que pueden aparecer sólo en **raras ocasiones** (con el sistema en producción y mucha carga)

Conclusiones

- Pese a que es un **campo maduro**, los modelos de **concurrencia** actuales siguen **evolucionando** para **facilitar** la construcción de programas concurrentes evitando sus muchas dificultades
- Eso implica que constantemente aparecer **nuevos modelos y técnicas**
- Actualmente la tendencia es usar técnicas **declarativas** propias de la **programación funcional** para definir **operaciones** que puedan ejecutarse de forma **concurrente**

Conclusiones

Concurrencia vs Paralelismo

Estos términos están muy relacionados pero no son sinónimos

Conclusiones

- **Concurrencia vs Paralelismo**
  - **Concurrencia**
    - En general se puede considerar que la **concurrencia** o la **programación concurrente** se refieren a **técnicas** de elaboración de **programas** con **varios flujos de ejecución**
  - **Paralelismo**
    - El **paralelismo** o **ejecución en paralelo** se utiliza para hablar de un tipo de **ejecución** de un programa **concurrente** sobre varios **procesadores** con procesos ejecutándose de forma **simultánea**

The Art of Concurrency. 2009. Clay Breshears. O'REALLY

Conclusiones

- **Concurrencia vs Paralelismo**
  - **Sistema concurrente**
    - Se dice que un **sistema es concurrente** si permite que dos o más acciones estén en **progreso** al mismo tiempo
  - **Sistema paralelo**
    - Se dice que un **sistema es paralelo** si permite que dos o más acciones se ejecuten de forma **simultánea**
    - Un sistema **paralelo** necesita al menos **dos procesadores**
    - El **paralelismo** es un tipo de **concurrencia**

The Art of Concurrency. 2009. Clay Breshears. O'REALLY

Conclusiones

- **Concurrencia vs Paralelismo**
  - Existen muchos casos en los que el concepto de paralelismo se utiliza en el sentido de concurrente
  - **Paralelización**
    - Se denomina **paralelización** al proceso de convertir programas **secuenciales** en programas **concurrentes**, generalmente con el objetivo de ejecutarse de forma **paralela** para aprovechar la potencia de cómputo
  - **Programación paralela**
    - En los últimos 20 años, el término **programación paralela** se ha usado para referiste a la programación de sistemas de **memoria distribuida** (en red) porque son **sistemas paralelos**

The Art of Concurrency. 2009. Clay Breshears. O'REALLY

Conclusiones

- **Concurrencia vs Paralelismo**
  - **Concurrencia**
    - Una propiedad de un sistema o programa
    - Un programa concurrente puede o no ser ejecutado en paralelo
  - **Paralelismo**
    - Comportamiento en tiempo de ejecución de ejecutar varias tareas al mismo tiempo

StackOverflow
http://stackoverflow.com/questions/1050222/concurrency-vs-parallelism-what-is-the-difference

Conclusiones

- **Concurrencia vs Paralelismo**
  - **Concurrencia**
    - Es la programación como la composición de procesos en ejecución independientes
    - Se refiere a tratar con muchas cosas a la vez. Se refiere a la estructura
    - Proporciona una forma de estructurar una solución para resolver un problema que puede ejecutarse en paralelo (aunque no es necesario).
    - La concurrencia es una forma de estructurar un programa dividiéndole en piezas que pueden ejecutarse de forma independiente
  - **Paralelismo**
    - Programación como la ejecución simultánea de computaciones (posiblemente relacionadas)
    - Se refiere a hacer muchas cosas a la vez. Se refiere a la ejecución

Rob Pike's talk: Concurrency is not Parallelism (it's better!)
http://vimeo.com/49718712
http://concur.rspace.googlecode.com/hg/talk/concur.html#landing-slide