

Apuntes-Tema-1-AC.pdf



Lisenk



Arquitectura de Computadores



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

LA RESIDENCIA PERFECTA PARA TI EN GRANADA

Amro Granada es nuestra nueva y moderna residencia para estudiantes en la histórica ciudad de Granada, a menos de cinco minutos a pie de las principales facultades.

Estancias Flexibles

PLAN AMIGO
Cheque Amazon

120€



Hay gente que no deja los apuntes porque
tiene miedo a la competencia

A NOSOTROS, NOS ENCANTA

COMPITE EN **GAMERSFY**
Y GANA **DINERO JUGANDO**
A TUS **VIDEOJUEGOS**
FAVORITOS

GAMERSFY

REGÍSTRATE CON
EL CÓDIGO:

WUOLAH

Y CONSIGUE **100**
MONEDAS GRATIS



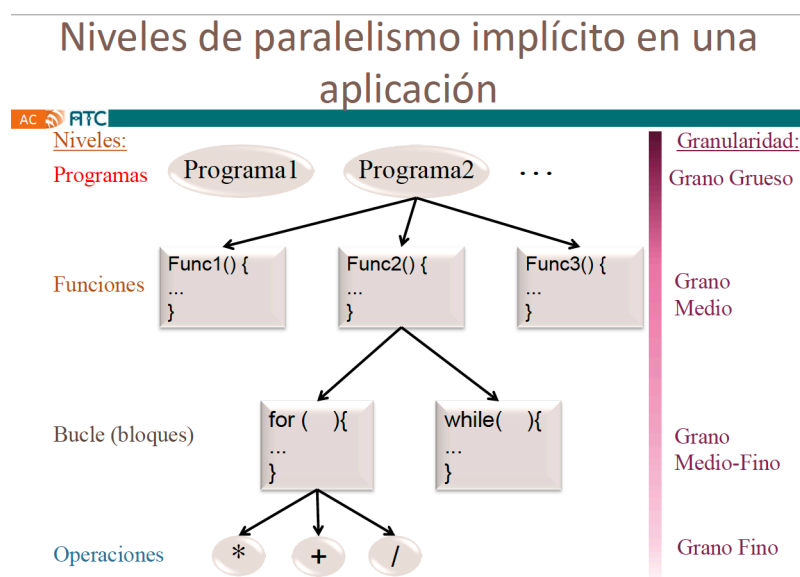
Tema 1. Arquitecturas Paralelas

1. Clasificación del paralelismo implícito en una aplicación

1.1 Criterios de clasificaciones del paralelismo implícito

- **Niveles de paralelismo implícito** en un código que resuelve la aplicación.
- **Paralelismo de tareas:** Capacidad para realizar tareas distintas a la vez (mediante distintos procesadores)
 - Se extrae de la **estructura lógica de funciones** en una aplicación
 - Relacionado con el **paralelismo a nivel de función**
- **Paralelismo de datos:** Hacemos la misma tarea, pero con datos diferentes a la vez (sin dependencia entre estos datos).
 - Se extrae de la **representación matemática** de la función
 - Relacionado con **paralelismo a nivel de bucle**.
- **Granularidad:** Es el tamaño de trabajo que tiene un paralelismo, es decir, el volumen de trabajo y tareas realizadas de cada parte del paralelismo.

1.2 Niveles de paralelismo implícito en una aplicación



1.3 Dependencias de datos

Para que se presente dependencia entre el bloque de código B2 con respecto a B1:

- Deben hacer referencia a una misma posición de memoria variable (M)
- B1 aparece en la secuencia de código antes que B2

Tipos de dependencia:

- **RAW** (Read After Write) / **Dependencia verdadera**

```
...  
a=b+c  
d=a+c  
...
```

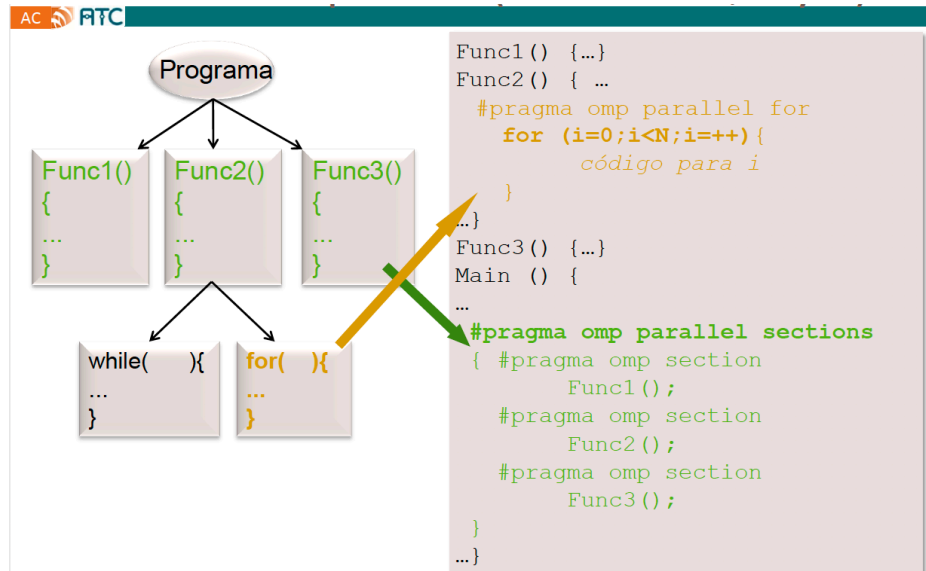
- **WAW** (Write After Write) / **Dependencia de salida**

```
...  
a=b+c  
...  
a=d+e  
...
```

- **WAR** (Write After Read) / **Antidependencia**

```
...  
b=a+1  
...  
a=d+e  
...
```

1.4 Paralelismo de datos y paralelismo de tareas en OpenMP



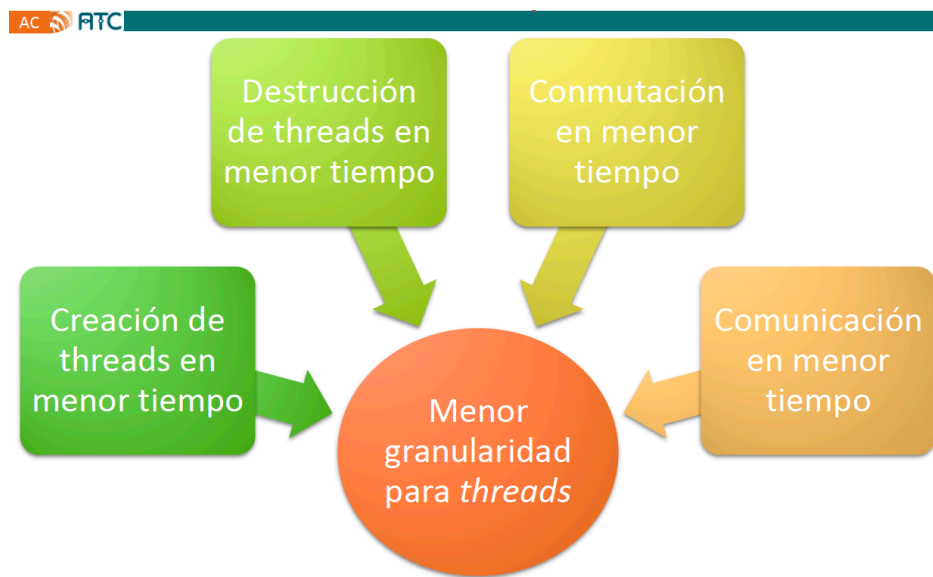
1.5 Paralelismo explícito. Unidades en ejecución en un computador

- La UC de un procesador gestiona la ejecución de instrucciones.
- **Proceso:** Mayor unidad de ejecución que gestiona el SO. Consta de múltiples flujos de control, llamados **threads**. Comprende el código del programa y lo que hace falta para su ejecución:
 - Datos en pila
 - Contenidos de los registros
 - Tabla de paginas
 - Tabla de ficheros abiertos

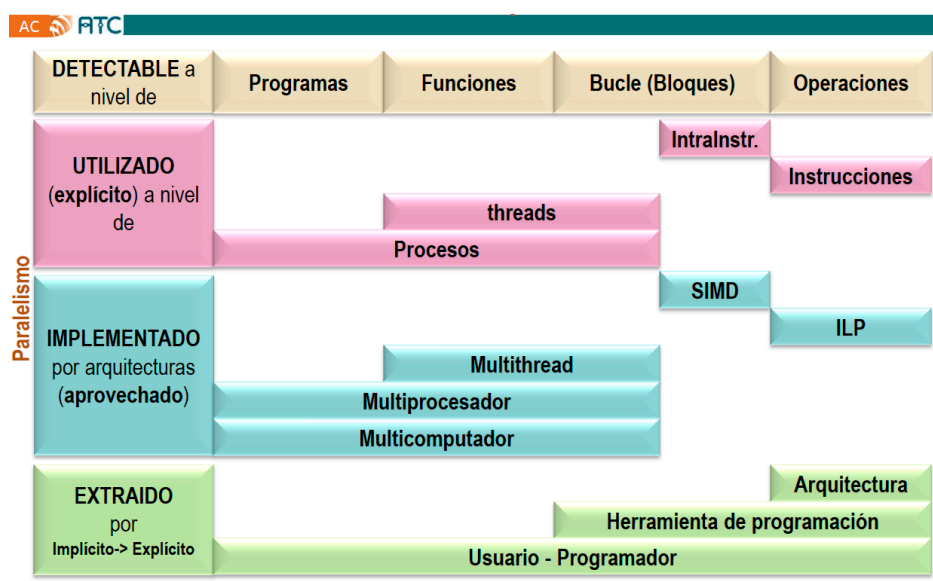
Los procesos se comunican mediante llamadas al SO

- **Thread / Light Process:** Menor unidad de ejecución que gestiona el SO. Menor secuencia de instrucciones que se pueden ejecutar en paralelo / concurrentemente (menor granularidad).

Los threads de un proceso se comunican mediante la memoria que comparten



1.6 Detección, utilización, implementación y extracción del paralelismo





Clases en
DIRECTO



Audio y vídeo
PROFESIONAL



Pizarra digital
COMPARTIDA



Máximo
10 PERSONAS

José Escribano Cobalea

lunes, 9 de marzo de 2020

2. Clasificación y prestaciones de arquitecturas paralelas

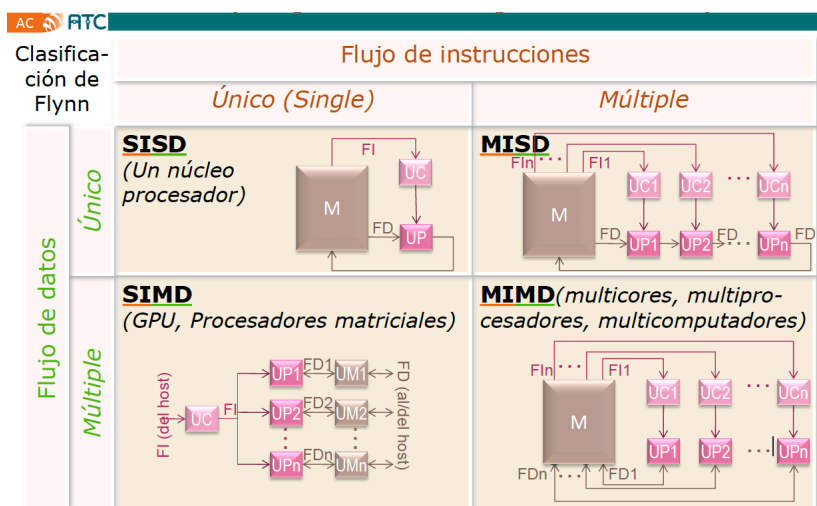
2.1 Computación paralela y computación distribuida

- **Computación paralela:** Desarrollo y ejecución de aplicaciones en un sistema de cómputo con múltiples procesadores/cores/procesadores (unidad autónoma)
- **Computación distribuida:** Desarrollo y ejecución de aplicaciones en un sistema distribuido (colección de recursos situados en distintas localizaciones físicas, conectados mediante redes)

2.2 Clasificaciones de arquitecturas y sistemas paralelos

- El uso que se le dará al computador alterará sus prestaciones y costes

Clasificación de Flynn de arquitecturas (1972)



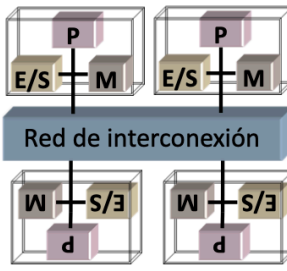
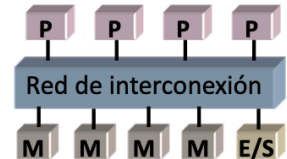
- **SISD:** Computadores uni-procesador
- **SIMD:** Aprovecha paralelismo de datos
- **MIMD:** Multiprocesadores y paralelismo funcional
- **MISD:** No existen computadores según este modelo, pero se puede conseguir programando el código de una MIMD

Sistemas de memoria

- **Multiprocesadores (SMP):** Misma red de interconexión para varios procesadores. El programador **no necesita** conocer donde están almacenados los datos
- **Multicomputadores:** Varios nodos computacionales (cada procesador con espacio de direcciones propio) conectados por una red de interconexión. El programador **necesita** conocer donde están almacenados los datos

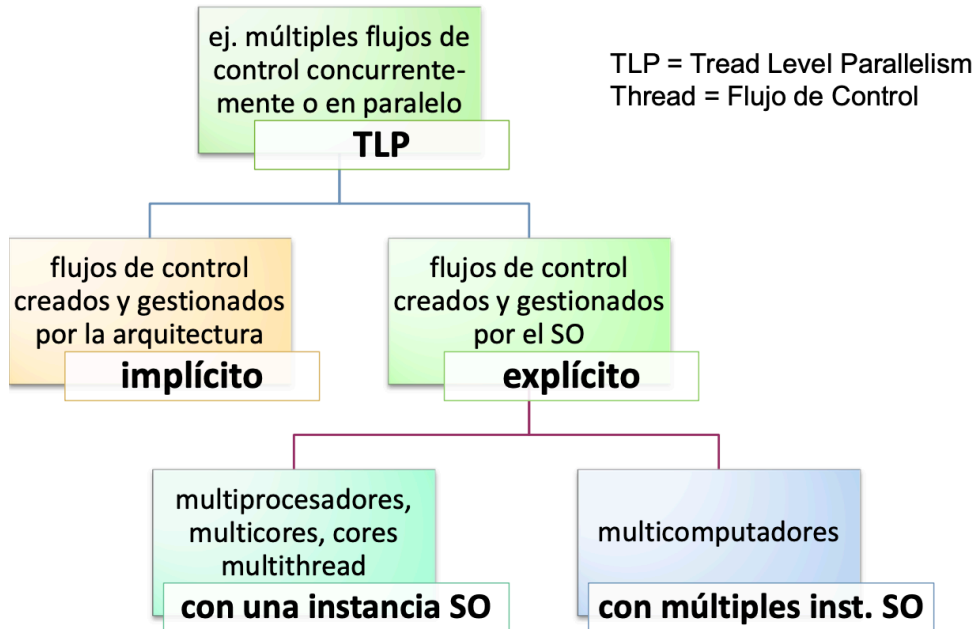
SMP	Multicomputadores
Mayor latencia - Poco escalable	Menor latencia - Más escalable
Comunicación implícita mediante variables compartidas. Datos no duplicados en memoria principal	Comunicación explícita mediante software para paso de mensajes (send/receive). Datos duplicados en memoria principal (copia)
Necesita implementar primitivas de sincronización	Sincronización mediante software de comunicación
Distribución código y datos entre procesadores no necesaria	Distribución código y datos entre procesadores necesaria
Programación mas sencilla	Programación mas difícil

- Incrementar estabilidad de multiprocesadores:
 - Aumentar cache del procesador
 - Usar redes de menor latencia y mayor ancho de banda que un bus

Multi-computadores Memoria no compartida	NORMA No Remote Memory Access	<i>ej. cluster, red de computadores</i>	Memoria físicamente distribuida	+	+
Multi-procesadores Memoria compartida Un único espacio de direcciones	NUMA Non-Uniform Memory Access	NUMA		Escalabilidad	Nivel de empaquetamiento y conexión
		CC-NUMA			
		COMA			
	UMA Uniform Memory Access	SMP Symmetric MultiProcessor	Memoria físicamente centralizada	-	-
					

- Distribuir físicamente los módulos de memoria entre los procesadores

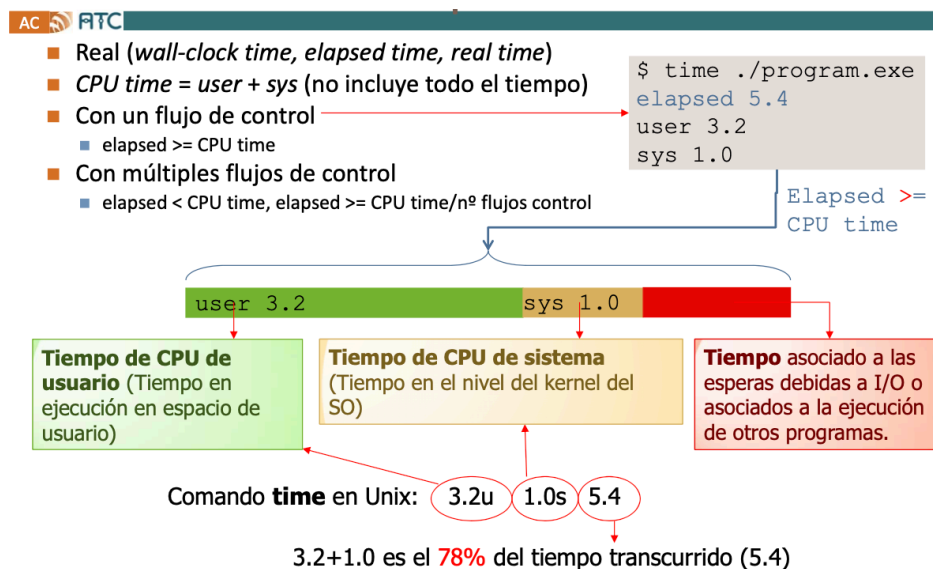
Flujos de control



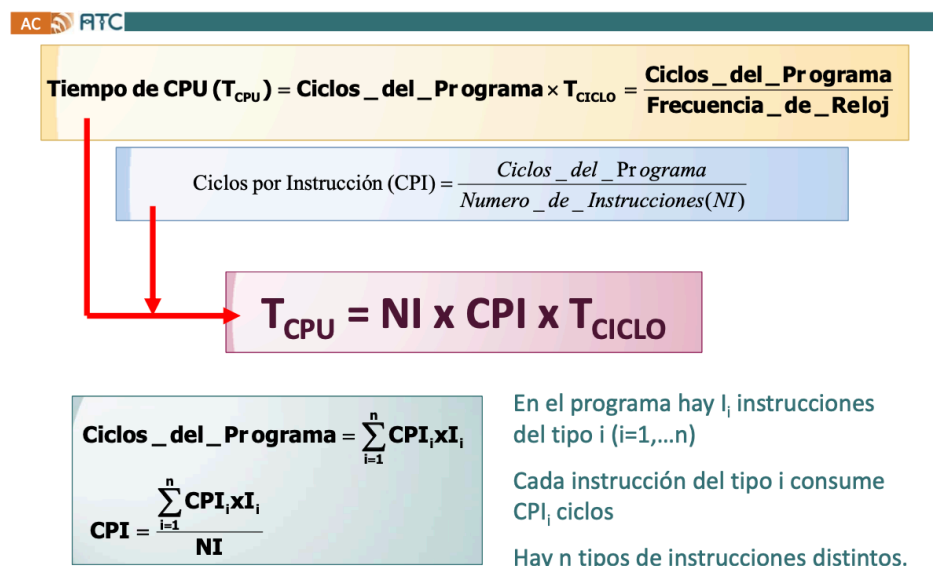
3. Evaluación de prestaciones de una arquitectura

3.1 Medidas usuales para evaluar prestaciones

3.1.1 Tiempo de respuesta



La suma de los tres tiempos de CPU conforman el tiempo de respuesta total



1 GHz = 1 nanoseg

0.5GHz = 2 nanoseg

2GHz = 0.5 nsg



Clases en
DIRECTO



Audio y vídeo
PROFESIONAL



Pizarra digital
COMPARTIDA



Máximo
10 PERSONAS

Una página más, y a por un café

Animo, tu puedes

José Escribano Cobalea

lunes, 9 de marzo de 2020

AC RTC

$$T_{CPU} = \underbrace{NI \times (CPE / IPE)}_{CPI} \times T_{ciclo}$$

Hay procesadores que pueden lanzar para que empiecen a ejecutarse (emitir) varias instrucciones al mismo tiempo.

CPE: Número mínimo de ciclos transcurridos entre los instantes en que el procesador puede emitir instrucciones

IPE: Instrucciones que pueden emitirse (para empezar su ejecución) cada vez que se produce dicha emisión.

AC RTC

$$T_{CPU} = \underbrace{(Noper / Op_instr)}_{NI} \times CPI \times T_{ciclo}$$

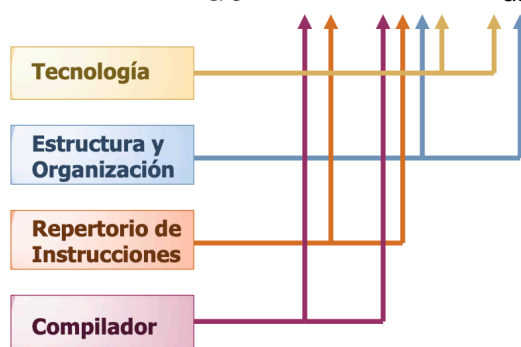
Hay procesadores que pueden codificar varias operaciones en una instrucción.

Noper: Número de operaciones que realiza el programa

Op_instr: Número de operaciones que puede codificar una instrucción.

AC RTC

$$T_{CPU} = NI \times CPI \times T_{ciclo}$$



3.1.2 Productividad

- **MIPS** (Millones de Instrucciones **P**or **S**egundo) (**M**eaningless **I**ndication of **P**rocessor **S**peed):
 - Depende del repertorio de instrucciones
 - Puede variar con el programa y inversamente con las prestaciones (mayor MIPS, peores prestaciones)

$$\text{MIPS} = \frac{\text{NI}}{T_{\text{CPU}} \times 10^6} = \frac{F(\text{frecuencia})}{\text{CPI} \times 10^6}$$

- **MFLOPS** (Millones de Operaciones en Coma Flotante **P**or **S**egundo):
 - Solo tiene en cuenta las operaciones en coma flotante (no es una medida adecuada para todos los programas),
 - El conjunto de operaciones en coma flotante no es constante en máquinas diferentes y la potencia de las operaciones en coma flotante no es igual para todas las operaciones
 - **Es necesaria una normalización de las instrucciones en coma flotante**

$$\text{MFLOPS} = \frac{\text{Operaciones_en_Coma_Flotante}}{T_{\text{CPU}} \times 10^6}$$

3.2 Conjunto de programas de prueba (Benchmark)

- Exigen fiabilidad y permitir comparar diferentes realizaciones de un sistema o diferentes sistemas
- Interesante para vendedores, fabricantes, investigadores y compradores de hardware o software
- Tipos:
 - Microbenchmark
 - Kernels
 - Sintéticos
 - Programas reales
 - Aplicaciones diseñadas

3. Ganancia en prestaciones

- Cuando en un computador se incrementan las prestaciones de un recurso, hacemos que su velocidad sea 'p' veces mayor. El incremento de velocidad respecto a anteriormente (máquina base) se expresa mediante la ganancia de velocidad (speed-up, S_p)

$$S_p = \frac{V_p}{V_1} = \frac{T_1}{T_p}$$

V_1 Velocidad de la máquina base

V_p Velocidad de la máquina mejorada (un factor p en uno de sus componentes)

T_1 Tiempo de ejecución en la máquina base

T_p Tiempo de ejecución en la máquina mejorada

3.1 Ley de Amdahl

- La mejora de velocidad, S, que se puede obtener cuando se mejora un recurso de una maquina en un factor p esta limitada por:

$$S \leq \frac{p}{1 + f(p - 1)}$$

donde f es la fracción del tiempo de ejecución en la máquina sin la mejora durante el que no se puede aplicar esa mejora

- **Hay que mejorar el caso más frecuente (lo que más se usa)**