Programación Sobre Redes

T2: Sistemas multiprocesador y paralelismo

Nicolás Mastropasqua

April 2, 2020

Instituto Industrial Luis A. Huergo

Contenidos

1. Repaso
2. Sistemas Multiprocesador
3. Paralelización
4. Scheduling con multiprocesador

Repaso

Repaso

Previamente, en Programación sobre redes...

- Concurrencia vs Paralelismo

Repaso

Previamente, en Programación sobre redes...

- Concurrencia vs Paralelismo
- Algoritmos de scheduling

Repaso

Previamente, en Programación sobre redes...

- Concurrencia vs Paralelismo
- Algoritmos de scheduling
- Ejercicios con diagramas de GANTT

Repaso

Previamente, en Programación sobre redes...

- Concurrencia vs Paralelismo
- Algoritmos de scheduling
- Ejercicios con diagramas de GANTT

Sistemas Multiprocesador

Sistema de procesador único

Concepto

Existe un único procesador de **propósito general**.

Sistema de procesador único

Concepto

Existe un único procesador de **propósito general**.

Pregunta: Dado un sistema con único procesador de propósito, sigue siéndolo si tiene...

- Procesador dedicado en el teclado

Sistema de procesador único

Concepto

Existe un único procesador de **propósito general.**

Pregunta: Dado un sistema con único procesador de propósito, sigue siéndolo si tiene...

- Procesador dedicado en el teclado
- Procesador dedicado para gráficos (GPU)

Sistema de procesador único

Concepto

Existe un único procesador de **propósito general**.

Pregunta: Dado un sistema con único procesador de propósito, sigue siéndolo si tiene...

- Procesador dedicado en el teclado
- Procesador dedicado para gráficos (GPU)
- Procesador dedicado para gestionar E/S

Sistema de procesador único

Concepto

Existe un único procesador de **propósito general**.

Pregunta: Dado un sistema con único procesador de propósito, sigue siéndolo si tiene...

- Procesador dedicado en el teclado
- Procesador dedicado para gráficos (GPU)
- Procesador dedicado para gestionar E/S

Sistema Multiprocesador

Concepto

Existen dos o más procesadores de propósito general que comparten distintos recursos del sistema, como la memoria, los buses, periféricos, etc.

Sistema Multiprocesador

Concepto

Existen dos o más procesadores de propósito general que comparten distintos recursos del sistema, como la memoria, los buses, periféricos, etc.

Ventajas y desventajas

- Mayor throughput (!)

Sistema Multiprocesador

Concepto

Existen dos o más procesadores de propósito general que comparten distintos recursos del sistema, como la memoria, los buses, periféricos, etc.

Ventajas y desventajas

- Mayor throughput (!)
- Mayor fiabilidad

Sistema Multiprocesador

Concepto

Existen dos o más procesadores de propósito general que comparten distintos recursos del sistema, como la memoria, los buses, periféricos, etc.

Ventajas y desventajas

- Mayor throughput (!)
- Mayor fiabilidad
- Mayor complejidad

Sistema Multiprocesador

Concepto

Existen dos o más procesadores de propósito general que comparten distintos recursos del sistema, como la memoria, los buses, periféricos, etc.

Ventajas y desventajas

- Mayor throughput (!)
- Mayor fiabilidad
- Mayor complejidad

Pregunta: ¿Qué otras ventajas/desventajas encuentran?

Multicore

Pregunta: ¿Qué tipo de sistema representa cada uno de los siguientes esquemas?

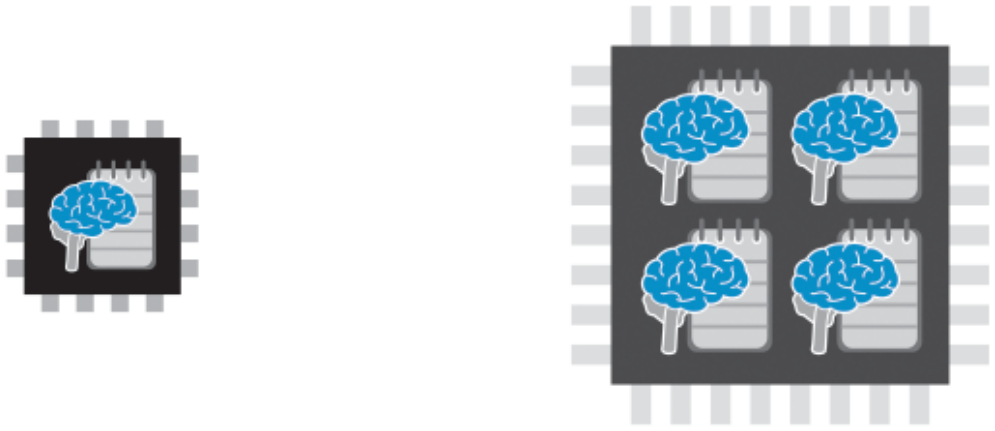


Figure 1: Procesador multicore

Multicore

Pregunta: ¿Qué tipo de sistema representa cada uno de los siguientes esquemas?

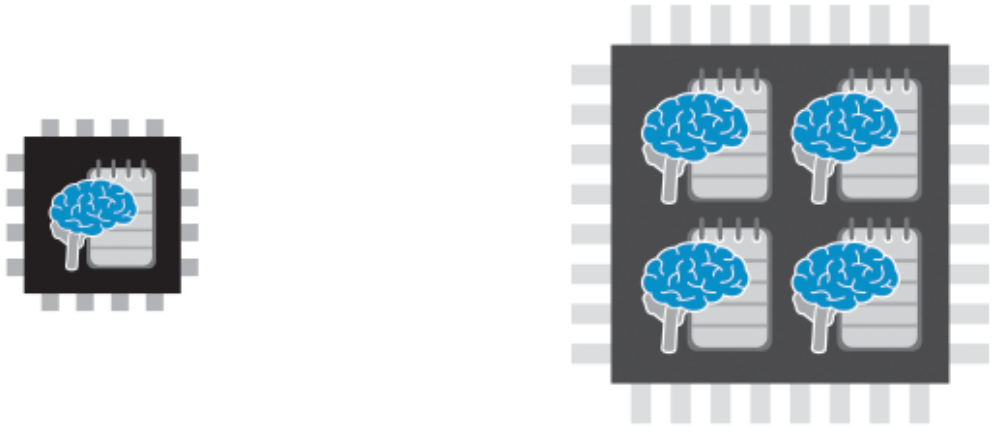


Figure 1: Procesador multicore

Multicores embebidos en un único chip

Para el sistema operativo, salvo algunos detalles como la memoria Cache, son tratados de forma equivalente a un sistema multiprocesador de N procesadores físicos

Tipos de multriprocesamiento

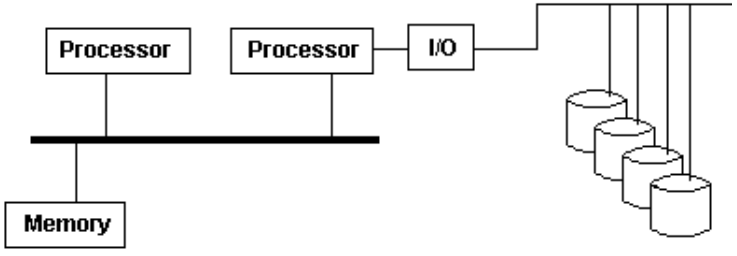


Figure 2: Ejemplo con procesador general y otro para gestionar E/S

Asimétrico

- Primera ”aproximación” al uso de múltiples procesadores
- No todos los procesadores hacen el mismo tipo de tarea (pero todos son de ”propósito general”).

Tipos de multriprocesamiento

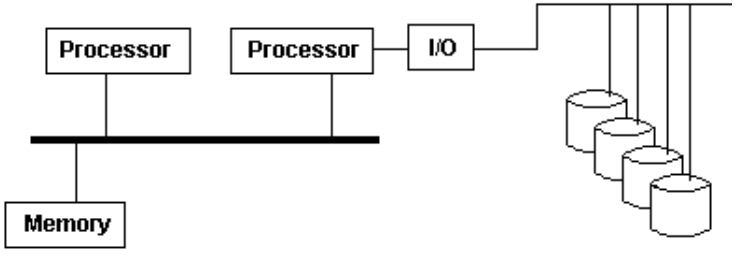


Figure 2: Ejemplo con procesador general y otro para gestionar E/S

Asimétrico

- Primera ”aproximación” al uso de múltiples procesadores
- No todos los procesadores hacen el mismo tipo de tarea (pero todos son de ”propósito general”).
- Existe un procesador **master** que controla el sistema, gestionando las tareas y los recursos en el resto de los procesadores **slaves**

Tipos de multriprocesamiento

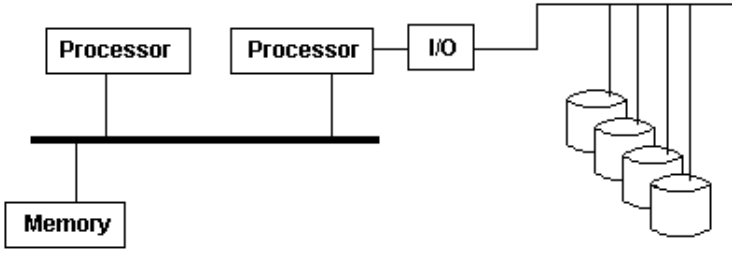


Figure 2: Ejemplo con procesador general y otro para gestionar E/S

Asimétrico

- Primera ”aproximación” al uso de múltiples procesadores
- No todos los procesadores hacen el mismo tipo de tarea (pero todos son de ”propósito general”).
- Existe un procesador **master** que controla el sistema, gestionando las tareas y los recursos en el resto de los procesadores **slaves**
- Los procesadores no necesitan comunicarse entre ellos.

Tipos de multriprocesamiento

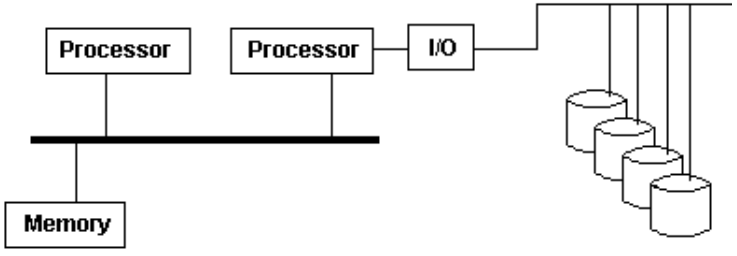


Figure 2: Ejemplo con procesador general y otro para gestionar E/S

Asimétrico

- Primera ”aproximación” al uso de múltiples procesadores
- No todos los procesadores hacen el mismo tipo de tarea (pero todos son de ”propósito general”).
- Existe un procesador **master** que controla el sistema, gestionando las tareas y los recursos en el resto de los procesadores **slaves**
- Los procesadores no necesitan comunicarse entre ellos.
- Si el Master falla, lo releva un slave. Si un slave falla, se puede realojar su tarea en otro.

Tipos de multriprocesamiento

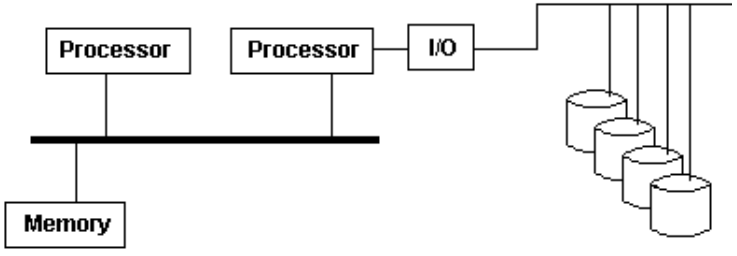


Figure 2: Ejemplo con procesador general y otro para gestionar E/S

Asimétrico

- Primera ”aproximación” al uso de múltiples procesadores
- No todos los procesadores hacen el mismo tipo de tarea (pero todos son de ”propósito general”).
- Existe un procesador **master** que controla el sistema, gestionando las tareas y los recursos en el resto de los procesadores **slaves**
- Los procesadores no necesitan comunicarse entre ellos.
- Si el Master falla, lo releva un slave. Si un slave falla, se puede realojar su tarea en otro.

Tipos de multriprocesamiento

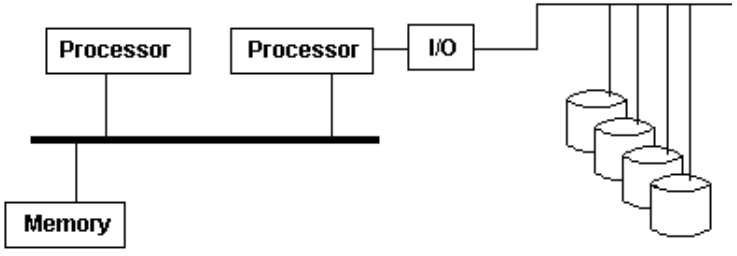


Figure 2: Ejemplo con procesador general y otro para gestionar E/S

Asimétrico

- Primera ”aproximación” al uso de múltiples procesadores
- No todos los procesadores hacen el mismo tipo de tarea (pero todos son de ”propósito general”).
- Existe un procesador **master** que controla el sistema, gestionando las tareas y los recursos en el resto de los procesadores **slaves**
- Los procesadores no necesitan comunicarse entre ellos.
- Si el Master falla, lo releva un slave. Si un slave falla, se puede realojar su tarea en otro.

Simétrico

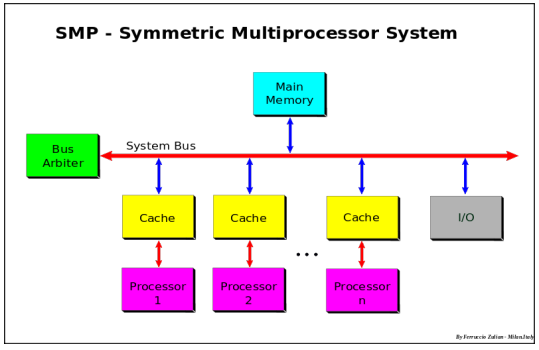


Figure 3: Esquema multiprocesador simétrico

Simétrico

- Todos los procesadores son **peers**. Cada uno puede retirar un trabajo disponible de forma independiente.

Simétrico

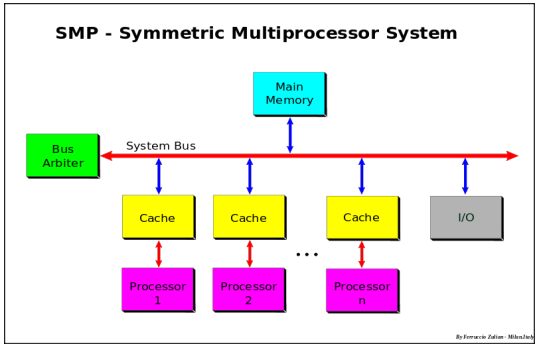


Figure 3: Esquema multiprocesador simétrico

Simétrico

- Todos los procesadores son **peers**. Cada uno puede retirar un trabajo disponible de forma independiente.
- Es más común.

Simétrico

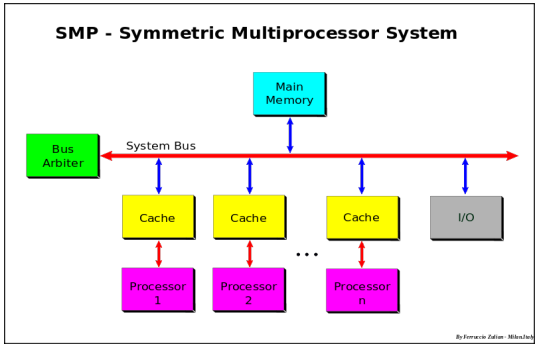


Figure 3: Esquema multiprocesador simétrico

Simétrico

- Todos los procesadores son **peers**. Cada uno puede retirar un trabajo disponible de forma independiente.
- Es más común.
- Se pueden ejecutar N tareas sin demasiado **overhead**.

Simétrico

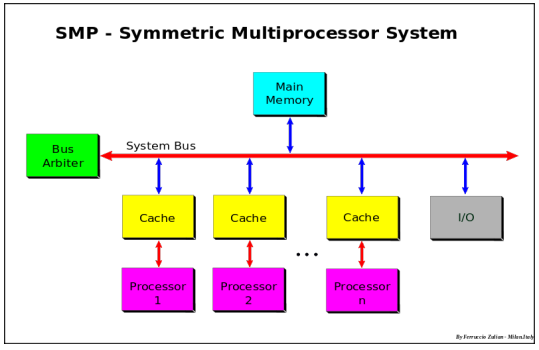


Figure 3: Esquema multiprocesador simétrico

Simétrico

- Todos los procesadores son **peers**. Cada uno puede retirar un trabajo disponible de forma independiente.
- Es más común.
- Se pueden ejecutar N tareas sin demasiado **overhead**.
- Los recursos (memorias, buses,etc) pueden ser compartidos de forma dinámica.

Simétrico

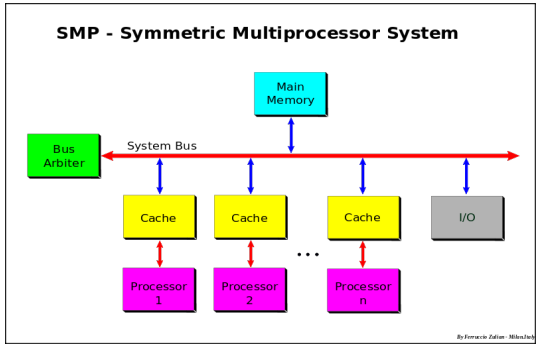


Figure 3: Esquema multiprocesador simétrico

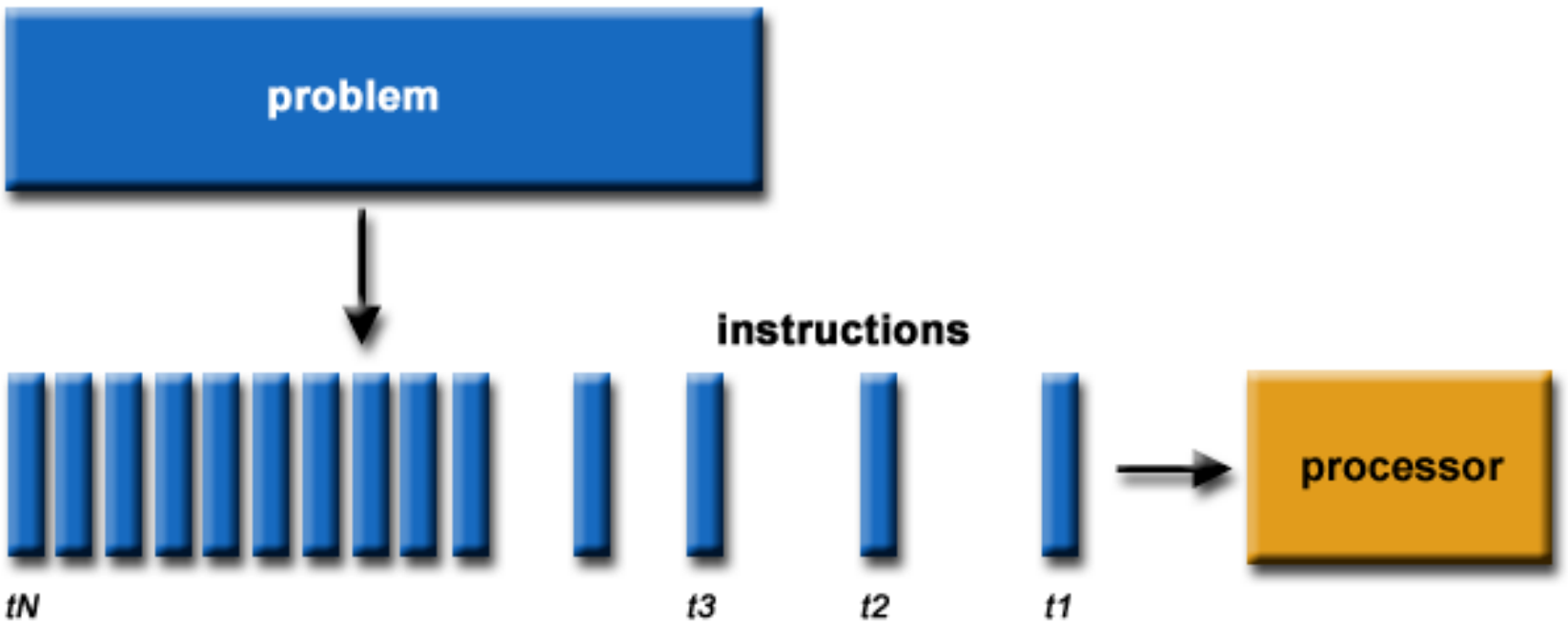
Simétrico

- Todos los procesadores son **peers**. Cada uno puede retirar un trabajo disponible de forma independiente.
- Es más común.
- Se pueden ejecutar N tareas sin demasiado **overhead**.
- Los recursos (memorias, buses,etc) pueden ser compartidos de forma dinámica.
- Aparecen los problemas de contención de recursos.

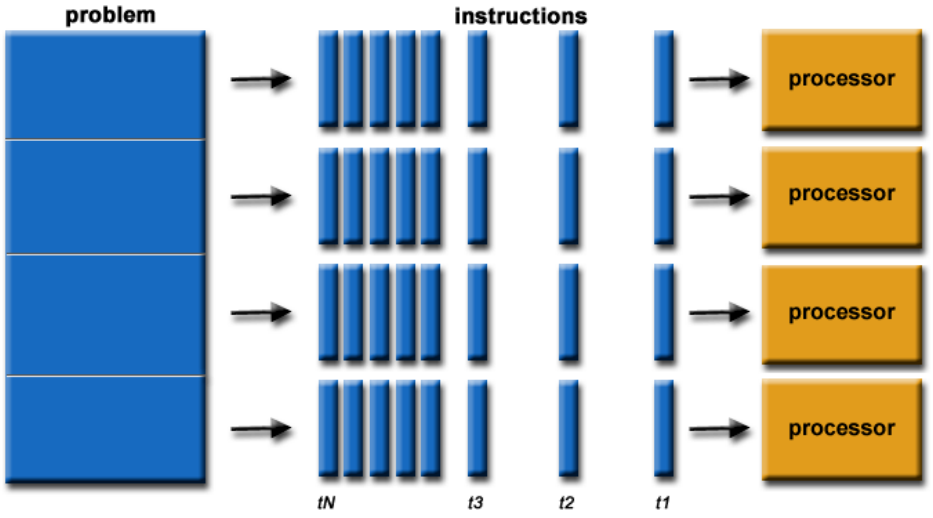
Paralelización

Computación serial

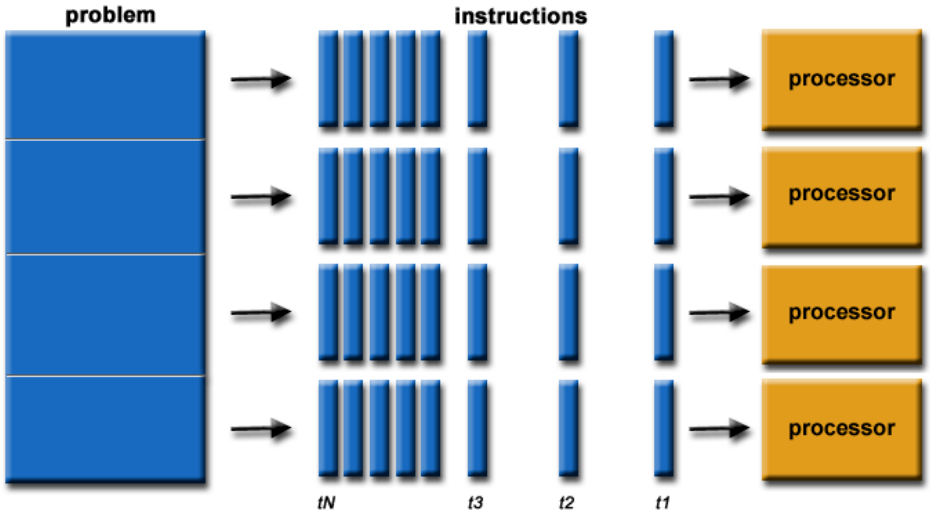
Hasta el momento, nuestro approach para resolver problemas se veía así:



Computación paralela



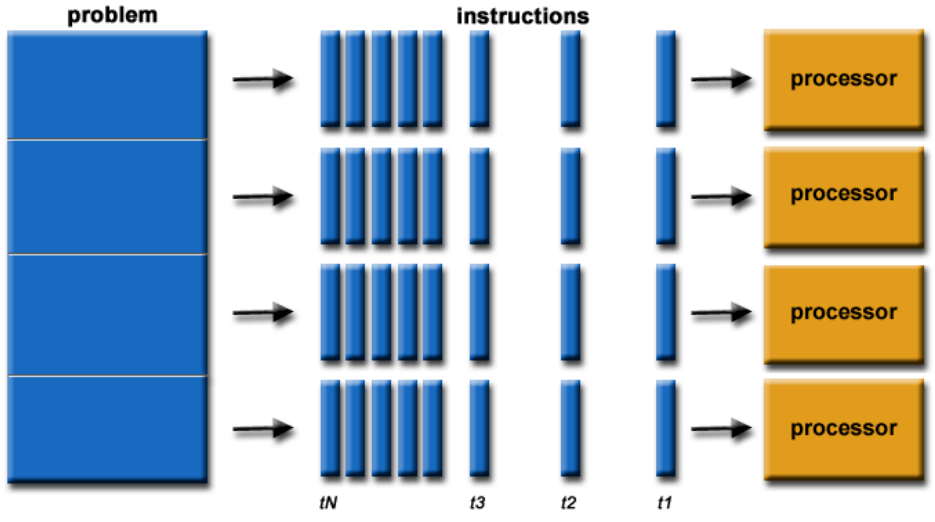
Computación paralela



Idea

- Bajo esta arquitectura,si queremos hacer que una tarea particular corra más rápida, podemos desarmarla en subtareas y ejecutarlas en paralelo

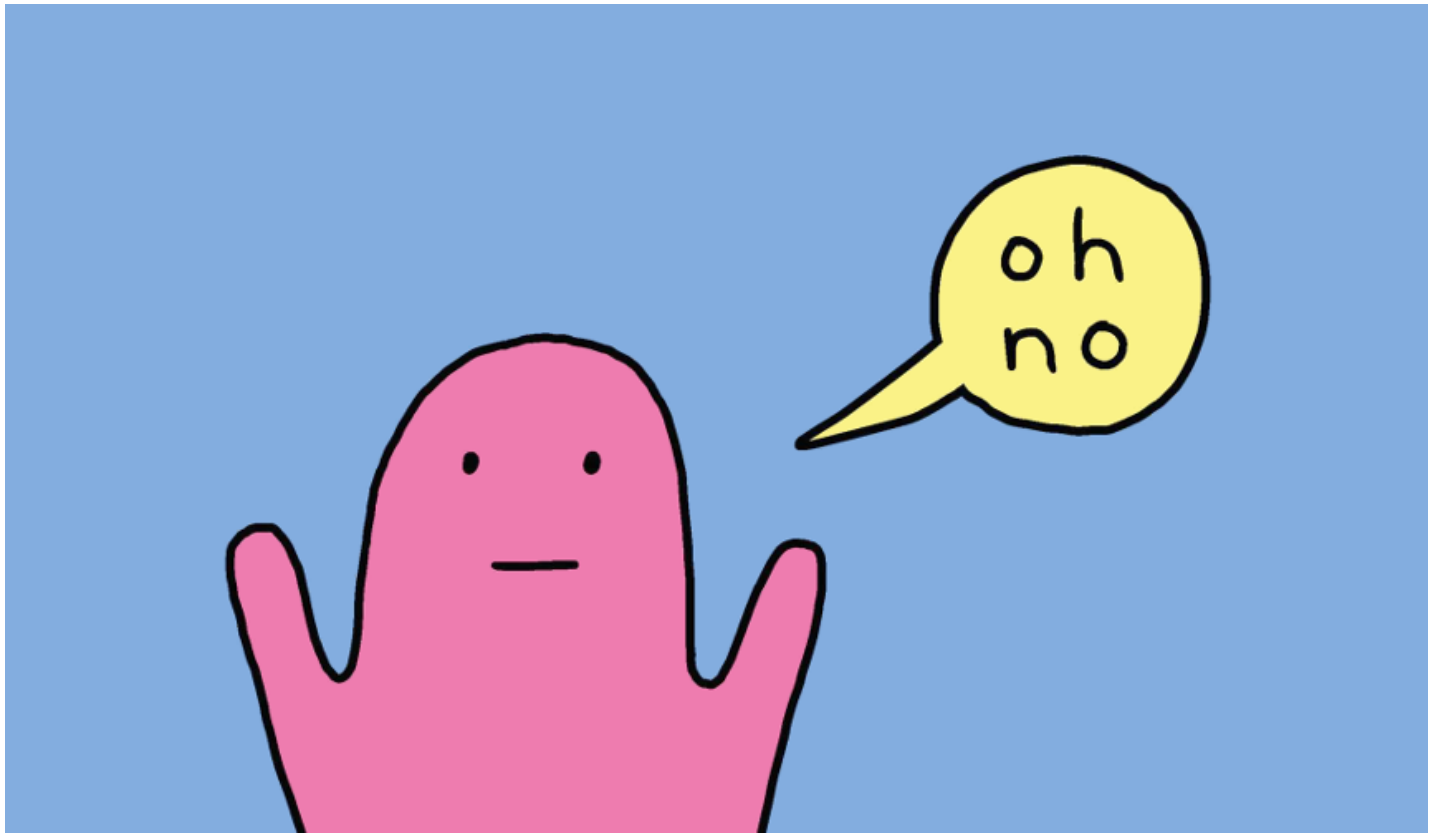
Computación paralela



Idea

- Bajo esta arquitectura,si queremos hacer que una tarea particular corra más rápida, podemos desarmarla en subtareas y ejecutarlas en paralelo
- Idealmente, con *n* CPUs tendríamos que la velocidad aumenta también con un factor *n*...
- Ojo! (El orden de) La cantidad pasos de la tarea son ”prácticamente” los mismos!

Oh no



La dura realidad del paralelismo

Problema de los pintores

Consideremos 5 amigos que deciden pintar una casa con 5 habitaciones.

La dura realidad del paralelismo

Problema de los pintores

Consideremos 5 amigos que deciden pintar una casa con 5 habitaciones.

- Pregunta: Si todas tuviesen el mismo tamaño, asignando uno a cada habitación, y asumiendo misma velocidad de trabajo. ¿Cuál es el factor de ganancia de velocidad?

La dura realidad del paralelismo

Problema de los pintores

Consideremos 5 amigos que deciden pintar una casa con 5 habitaciones.

- Pregunta: Si todas tuviesen el mismo tamaño, asignando uno a cada habitación, y asumiendo misma velocidad de trabajo. ¿Cuál es el factor de ganancia de velocidad?
- 5 veces con respecto tener un único pintor!

La dura realidad del paralelismo

Problema de los pintores

Consideremos 5 amigos que deciden pintar una casa con 5 habitaciones.

- Pregunta: Si todas tuviesen el mismo tamaño, asignando uno a cada habitación, y asumiendo misma velocidad de trabajo. ¿Cuál es el factor de ganancia de velocidad?
- 5 veces con respecto tener un único pintor!
- Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

La dura realidad del paralelismo

Problema de los pintores

Consideremos 5 amigos que deciden pintar una casa con 5 habitaciones.

- Pregunta: Si todas tuviesen el mismo tamaño, asignando uno a cada habitación, y asumiendo misma velocidad de trabajo. ¿Cuál es el factor de ganancia de velocidad?
- 5 veces con respecto tener un único pintor!
- Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

Pintores con habitaciones distintas

Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

Pintores con habitaciones distintas

Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

- El tiempo total va a estar dominado por lo que tome esta última

Pintores con habitaciones distintas

Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

- El tiempo total va a estar dominado por lo que tome esta última
- Intuitivamente, vemos que la ganancia es **solo 3 veces superior** a la de un solo pintor

Pintores con habitaciones distintas

Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

- El tiempo total va a estar dominado por lo que tome esta última
- Intuitivamente, vemos que la ganancia es **solo 3 veces superior** a la de un solo pintor
- La **fracción** de la tarea que es paralelizable determinará la **ganancia** que obtendremos

Pintores con habitaciones distintas

Pregunta: ¿Qué pasa si una de las habitaciones tiene el **doble de tamaño?**

- El tiempo total va a estar dominado por lo que tome esta última
- Intuitivamente, vemos que la ganancia es **solo 3 veces superior** a la de un solo pintor
- La **fracción** de la tarea que es paralelizable determinará la **ganancia** que obtendremos
- No siempre es posible paralelizar!

Ejercicio

Consideremos el algortmo conocido como ”La criba de Eratóstenes”, que dado un entero *n* obtiene todos los primos menores a *n*. ¿Podría paralelizarse? ¿Cómo?

```
Unmarked = {2,3,...,n}
for all a from 2 to sqrt(n):
    if a is in Unmarked then:
        for all m in muliplesLessThan(a,n):
            remove(Unmarked,m)
return unmarked
```

Scheduling con multiprocesador

Scheduling con multiprocesador

- Hasta ahora vimos distintas formas de hacer scheduling de procesos con **único procesador**

Scheduling con multiprocesador

- Hasta ahora vimos distintas formas de hacer scheduling de procesos con **único procesador**
- ¿Cómo cambia la situación en el caso de tener **múltiples procesadores** y un esquema simétrico?

Scheduling con multiprocesador

- Hasta ahora vimos distintas formas de hacer scheduling de procesos con **único procesador**
- ¿Cómo cambia la situación en el caso de tener **múltiples procesadores** y un esquema simétrico?
- Dos alternativas:
  - Todos los procesos están en única cola de listos
  - Cada procesador tiene su cola privada. En este caso, puede tener sentido implementar políticas de balanceo de carga
    - Push migration: Se monitorea la carga de cada CPU y en caso de desbalance, se redistribuyen las tareas
    - Pull migration: Si un CPU está IDLE, puede robar una tarea de otro ocupado

Afinidad

Memoria Cache

En general, se emplea esta memoria, más rápida que la RAM, para tratar de tener ”más a mano” las porciones de datos que ”más usa” un programa.

- Cuando desalojamos una tarea por una nueva, la Cache tiene que volver a llenarse según el contexto de la nueva

Afinidad

Memoria Cache

En general, se emplea esta memoria, más rápida que la RAM, para tratar de tener ”más a mano” las porciones de datos que ”más usa” un programa.

- Cuando desalojamos una tarea por una nueva, la Cache tiene que volver a llenarse según el contexto de la nueva
- Si hacemos esto muy seguido, estamos atacando la performance de cada tarea

Afinidad

Memoria Cache

En general, se emplea esta memoria, más rápida que la RAM, para tratar de tener ”más a mano” las porciones de datos que ”más usa” un programa.

- Cuando desalojamos una tarea por una nueva, la Cache tiene que volver a llenarse según el contexto de la nueva
- Si hacemos esto muy seguido, estamos atacando la performance de cada tarea
- Por eso, el sistema operativo puede tener políticas de afinidad, más o menos duras, para evitar la migración de tareas entre procesadores

Ejercicio

Dados los siguientes procesos

Tarea	Duracion	Tiempo de llegada	E/S
T0	7	0	
T1	6	1	
T2	9	2	2-5

Dibujar el diagrama de GANTT para un scheduler RR con quantum = 2 y permitiendo la migración entre los tres núcleos (afinidad blanda)

Bibliografía

- *Abraham Silberschatz, Peter Baer Galvin, Greg Gagne.* ***Operating System Concepts Eigth Edition****, 5.5-5.5.4*
- Video de Computerphile: Hasta 8:30 cubre varios conceptos vistos en esta clase

¿Dudas?

TO DO: CAMBIAR MEME BABY YODA!!