Sistemas Operativos

Planificación de uso de CPU

Viktor Andrés Tapia Vásquez

Segundo Semestre 2021

Departamento de Informática, Campus SSJJ.

Índice de Contenidos

1. Conceptos Básicos

2. Algoritmos de Itineración

3. Multiprocesamiento

Conceptos Básicos

Conceptos Básicos

- Cuando hay muchas tareas por hacer, ¿Cuál ejecutamos primero?
- El criterio de selección definirá la naturaleza del SO.
- Objetivo principal: Maximizar el uso de la CPU.
- La idea es simple: Si la ejecución de un proceso es interrumpida (sin importar la razón), otro debe tomar su lugar.
- Toda política de itineración presenta un complejo conjunto de decisiones entre varias propiedades deseables.

Conceptos Básicos

- El itinerador elige un proceso desde la cola ready y le asigna la CPU.
- Las decisiones de itineración se toman cuando los procesos pasan:
  - ✓ Desde el estado running a waiting.
  - ✓ Desde el estado running a ready.
  - ✓ Desde el estado waiting a ready.
  - ✓ Fin de un proceso.

Conceptos Básicos

- El despachador entrega el control de la CPU al proceso seleccionado por le itinerador de corto plazo.
- Esta asignación produce:
  - ✓ Cambio de contexto.
  - ✓ Cambio a modo usuario.
  - ✓ Saltar a la dirección correcta dentro del programa y reiniciarlo.
- El despachador debe ser lo más rápido posible.
- El tiempo que utiliza en detener un proceso e iniciar la ejecución de otro se conoce como tiempo de latencia del despachador.

Conceptos Básicos

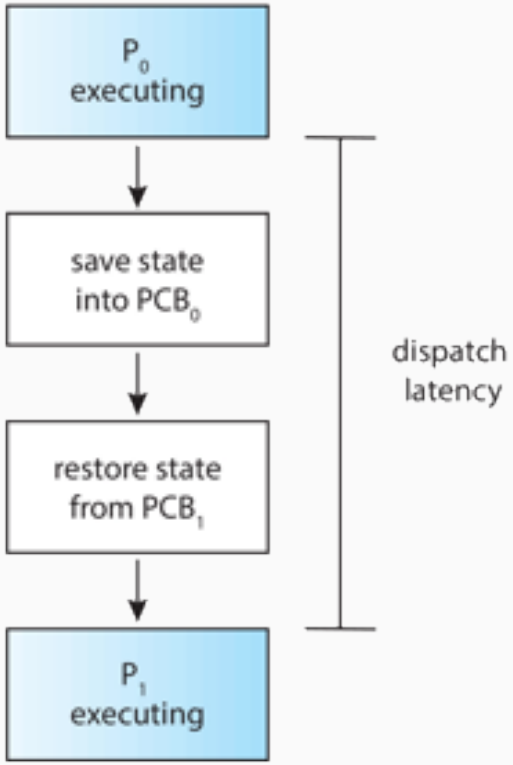


Figure 1: Latencia del despachador

Conceptos Básicos - Definiciones

- **Task:** Job o tarea. Cualquier requerimiento de usuario. Una hebra o un proceso podrían ser responsables de múltiples tareas.
- **Workload:** Conjunto de tareas a realizar por el sistema. Dada una carga de trabajo, el itinerador decide la asignación de CPU.
- **Equidad:** Igualdad en los tiempos y recursos entregados.
- **Overhead:** Tiempo en pasar de una tarea a otra.

Conceptos Básicos - Definiciones

- **Inanición:** Una tarea no avanza debido a que los recursos están asignados a tareas de mayor prioridad.
- **Scheduling Preemtive:** Los recursos se pueden quitar.
- **Conservación de trabajo:** Un algoritmo de itineración es conservador de trabajo si nunca deja el procesador ocioso cuando hay trabajo que hacer.

Algoritmos de Itineración

Algoritmos de Itineración - Criterios

Al momento de decidir que algoritmo utilizamos debemos considerar:

- **Uso de CPU:** Mantenerla siempre ocupada.
- **Tasa Procesamiento:** Tareas realizadas por unidad de tiempo.
- **Tiempo ejecución:** Desde que se ordena la ejecución hasta su fin.
- **Tiempo espera:** Suma de intervalos en la cola ready.
- **Tiempo respuesta:** Intervalo desde la llegada a la cola ready hasta la primera asignación de CPU.

Algoritmos de Itineración - Criterios

Objetivo Principal

- ✓ Maximizar el uso de CPU.
- ✓ Maximizar la tasa de procesamiento.
- ✓ Minimizar el tiempo de ejecución.
- ✓ Minimizar el tiempo de espera.
- ✓ Minimizar el tiempo de respuesta.

Formula Palta

T. Ejecución = T. Respuesta + T. Espera + E/S + Uso CPU

Algoritmos de Itineración - FCFS

- Es el algoritmo más sencillo.
- La CPU es asignada a la tarea que la solicite primero.
- Se gestiona a través de una cola FIFO.
- El tiempo promedio de espera es alto.
- Consideremos la siguiente carga de trabajo:

Process	Burst Time
P_1	24
P_2	3
P_3	3

Algoritmos de Itineración - FCFS

Supongamos que llegan en el orden P1, P2 y P3:

- Tiempo de espera: P1 = 0, P2 = 24 y P3 = 27.
- Tiempo de espera promedio: 17

Algoritmos de Itineración - FCFS

Consideremos ahora que el orden es P2, P3 y P1:

- Tiempo de espera: P1 = 6, P2 = 0 y P3 = 3.
- Tiempo de espera promedio: 3
- La disminución es considerable.
- FCFS no optimiza el tiempo de espera.
- FCFS sufre el efecto convoy: Tareas cortas detrás de una larga.

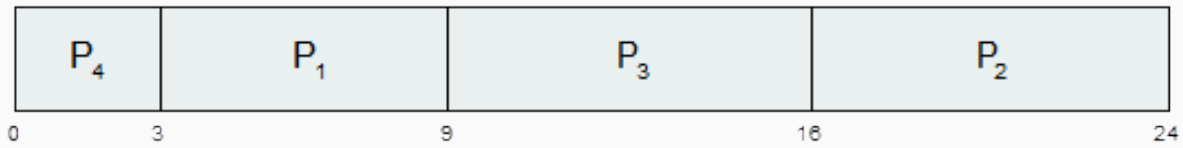
Algoritmos de Itineración - SJF

- A cada tarea se le asocia el tiempo de uso de CPU.
- La tarea con la ráfaga más corta tiene mayor prioridad.
- Si existe empeta, se resuelve por FCFS.
- SJF es óptimo. Entrega el **mínimo** tiempo de espera.
- La dificultad está en conocer el tiempo de uso de CPU.

Algoritmos de Itineración - SJF

Ejemplo: Todos los procesos llegan al mismo tiempo.

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



- Tiempo de espera promedio: (3+16+9+0) / 4 = 7

Algoritmos de Itineración - SJF

- Sufre de inanición.
- Las tareas cortas se ven favorecidas.
- La ráfaga de uso de CPU se debe estimar.
- Se usa como mecanismo a largo plazo.
- Puede ser:
  - ✓ **Cooperativo:** Permite que la tarea termine.
  - ✓ **Apropiativo:** Detiene la tarea en ejecución.

Algoritmos de Itineración - SJF

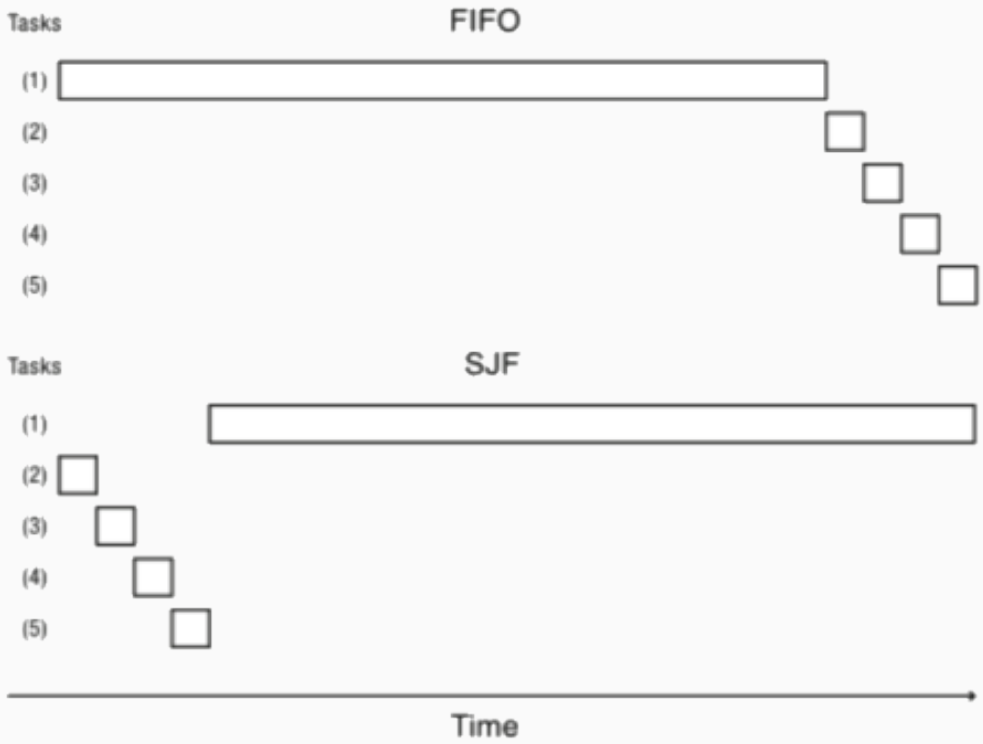
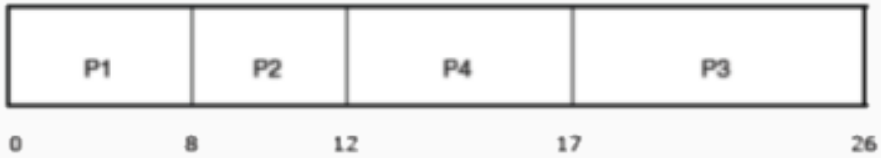


Figure 2: FIFO v/s SJF

Algoritmos de Itineración - SJF

Ejemplo: Cooperativo.

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

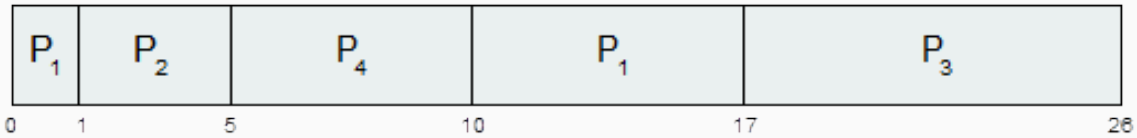


- Tiempo de espera promedio: (0+(8-1)+(17-2)+(12-3))/ 4 = 7,75

Algoritmos de Itineración - SJF

Ejemplo: Apropiativo.

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



- Tiempo de espera promedio: ((10-1)+(1-1)+(17-2)+(5-3))/ 4 = 6,5

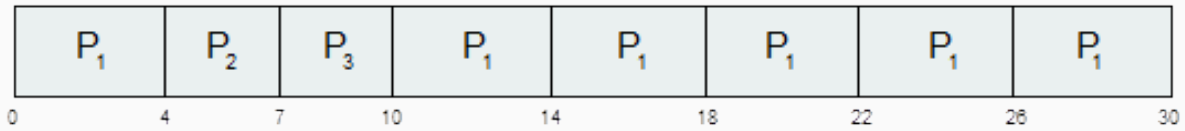
Algoritmos de Itineración - Round Robin

- A cada tarea se le asigna un **quantum de tiempo** (q).
- Usualmente q está entre 10 y 100 milisegundos.
- Al terminar q, si la tarea no ha finalizado, se interrumpe.
- La tarea interrumpida se agrega al final de la cola ready.
- Desempeño:
  - ✓ Si q es grande se transforma en FCFS.
  - ✓ Si q es pequeño hay que tener ojo con el overhead.

Algoritmos de Itineración - Round Robin

Ejemplo: quantum = 4.

Process	Burst Time
P_1	24
P_2	3
P_3	3



- Tiempo de respuesta mejor que SJF.
- Tiempo promedio de espera generalmente largo.

Algoritmos de Itineración - Round Robin

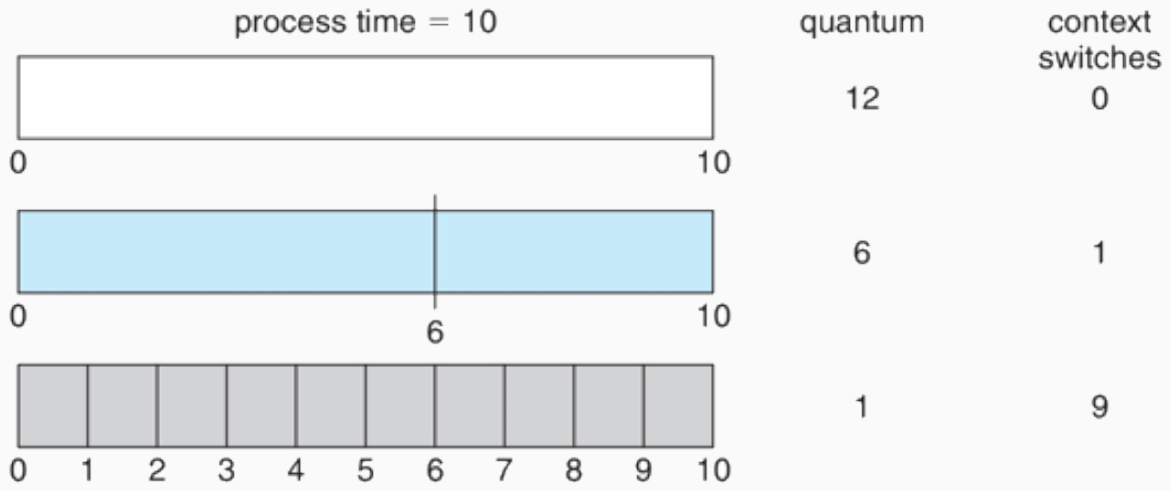


Figure 3: Cambios de contexto en round robin

Algoritmos de Itineración - Round Robin

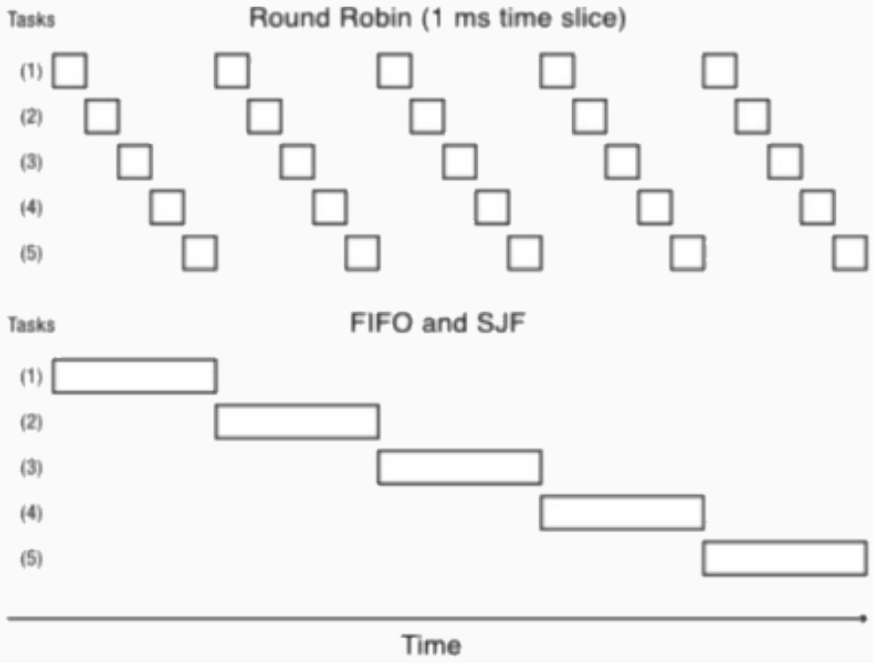


Figure 4: RR v/s FIFO v/s SJF

Algoritmos de Itineración - Prioridades

- Cada tarea tiene un número entero como prioridad.
- La CPU se asigna a la tarea con la prioridad más alta.
- SJF es un caso especial de prioridades.
- Puede ser:
  - ✓ **Cooperativo:** Permite que la tarea termine.
  - ✓ **Apropiativo:** Detiene la tarea en ejecución.
- Sufre de inanición:
  - ✓ Se soluciona con la ténica del *envejecimiento*.
  - ✓ A medida que pasa el tiempo aumenta la prioridad.

Algoritmos de Itineración - Prioridades

Ejemplo: Todos llegan al mismo tiempo.

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Tiempo promedio de espera: $(6 + 0 + 16 + 18 + 1)/5 = 8{,}2$

Algoritmos de Itineración - Prioridades

Ejemplo: Procesos con misma prioridad utilizan RR con q = 2.

Process	Burst Time	Priority
P_1	4	3
P_2	5	2
P_3	8	2
P_4	7	1
P_5	3	3

- La asignación de P2 y P3 puede ser por oden, aleatoria, etc.

Algoritmos de Itineración - Varios Niveles

- Las tareas se clasifican según su función.
- La cola ready es separada en distintas colas.
- Cada cola puede tener un algortimo distinto.
- Los procesos se asignan **permanentemente** a cada cola.
- La planificación entre colas es apropiativa y de prioridad fija.

Algoritmos de Itineración - Varios Niveles

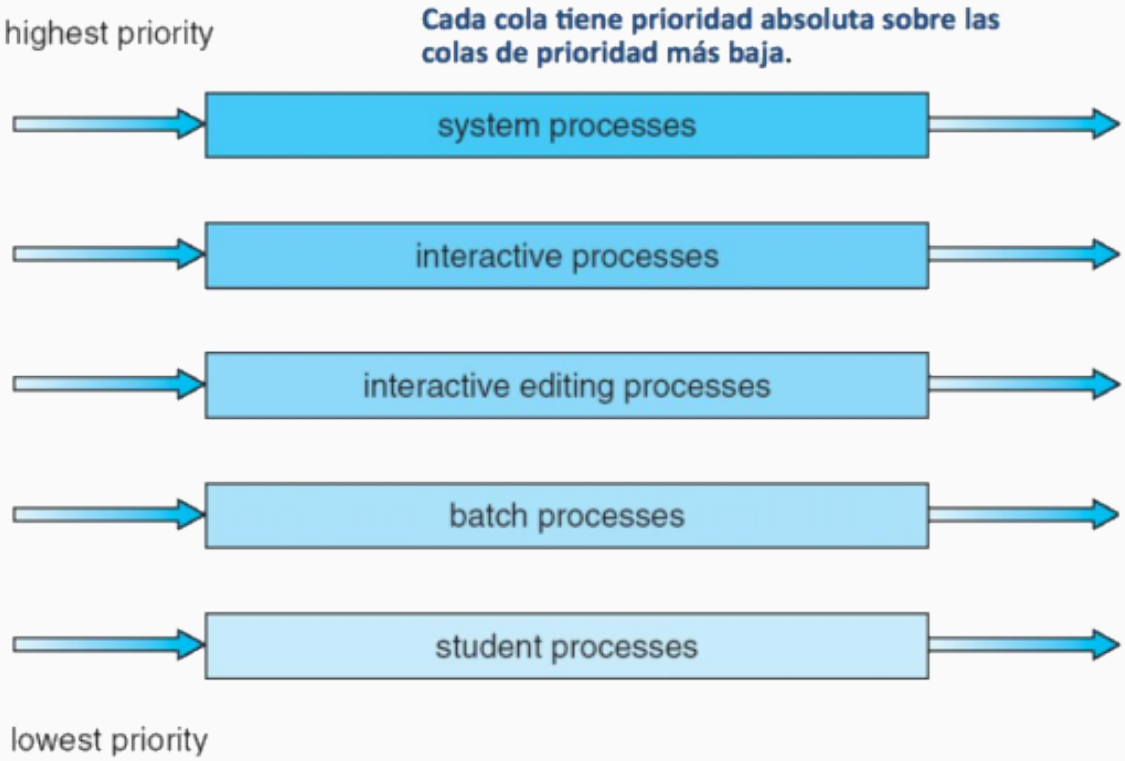


Figure 5: Varios niveles

Algoritmos de Itineración - MFQ

- SO's modernos utilizan MFQ. Es una extensión de RR.
- Se tienen múltiples colas, cada una con diferentes algortimos.
- Colas de alta prioridad utilizan RR con q's pequeños.
- La planificación entre colas es apropiativa.
- Las tareas descienden a medida que no alcanzan a terminar.
- El último nivel puede ser FIFO o RR.
- Si en medio de un quantum se produce E/S, la tarea puede volver al mismo nivel o al superior.

Algoritmos de Itineración - MFQ

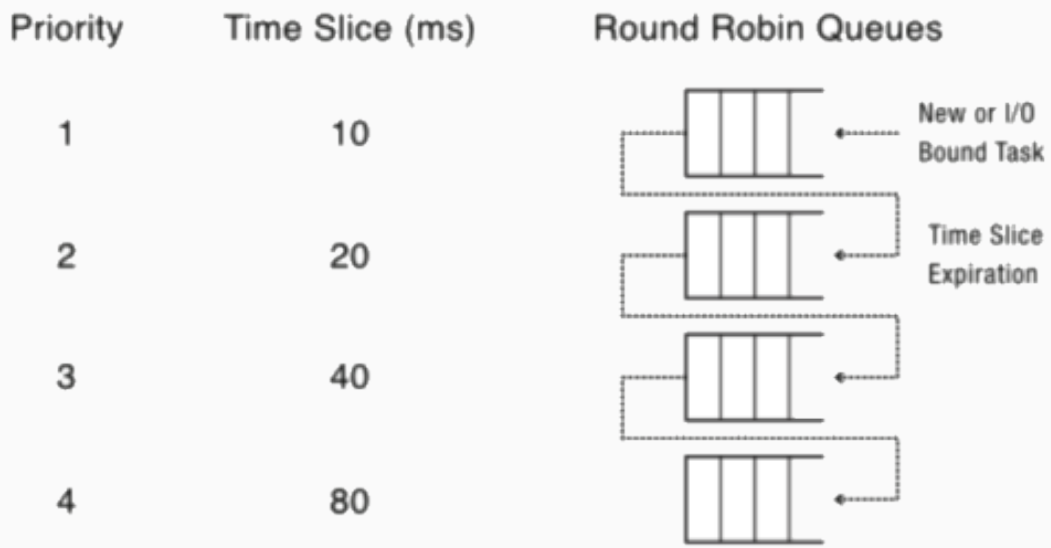


Figure 6: Implementación MFQ

Algoritmos de Itineración - MFQ

Ejemplo: Una MFQ dispone de 3 colas (Q1, Q2 y Q3, priorizadas en este orden) en donde la itineración corresponde a RR = 4, RR = 7 y FCFS respectivamente. Considere la siguiente carga de trabajo:

Process	CPU, I/O, CPU	Arrival Time
P0	5,6,7	0
P1	4,2,3	3
P2	2,3,4	4
P3	5,2,7	7
P4	3,2,4	14

Si inicialmente los procesos llegan a Q1, se pide construir la planificación y calcular los tiempos de espera, respuesta y ejecución promedio. Si un proceso hace E/S vuelve a la misma cola donde estaba.

Multiprocesamiento

Algoritmos de Itineración - Multiprocesamiento

- Computadores modernos cuentan con más de un procesador.
- La itineración de tareas se vuelve más compleja en este caso.
- Desafios:
  - ✓ ¿Cómo utilizar múltiples procesadores?
  - ✓ ¿Cómo utilizar los algoritmos de itineración?
- Arquitecturas para multiprocesamiento:
  - ✓ CPU's Multicore.
  - ✓ Core's multithreaded.

Algoritmos de Itineración - Multiprocesamiento

- Multiprocesamiento simétrico: Cada procesador itinera.
- Las hebras pueden estar en una cola ready común.
- Cada procesador puede tener sus colas privadas.

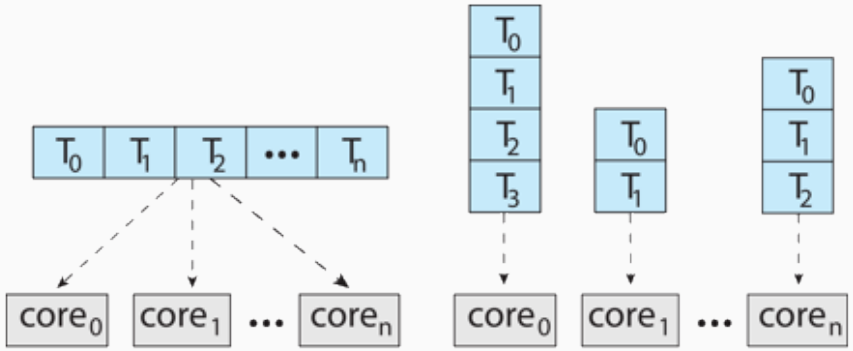


Figure 7: Cola ready común v/s Colas privadas por core

Algoritmos de Itineración - Multiprocesamiento

- SO's comerciales utilizan MFQ's por procesador.

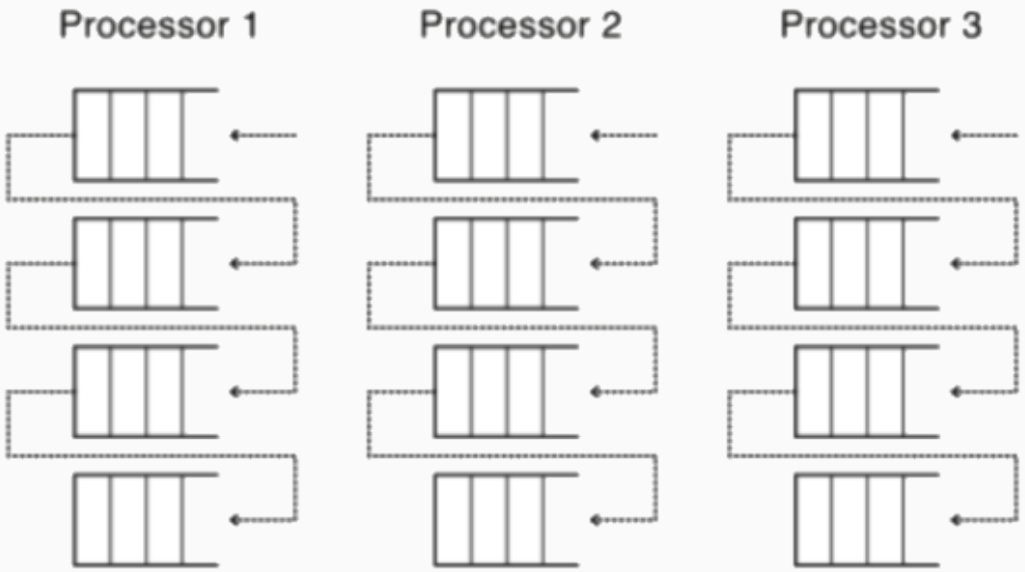


Figure 8: MFQ por procesador

Algoritmos de Itineración - Multiprocesamiento

- Tendencias tecnológicas incorporan múltiples núcleos por procesador.
- Más rápido y consume menos.
- Múltiples hebras por núcleo.
- **Hyperthreading**.
  - ✓ Un sistema *quad-core* con dos hardware threads por núcleo permite al SO ver 8 procesadores lógicos.

Algoritmos de Itineración - Multiprocesamiento

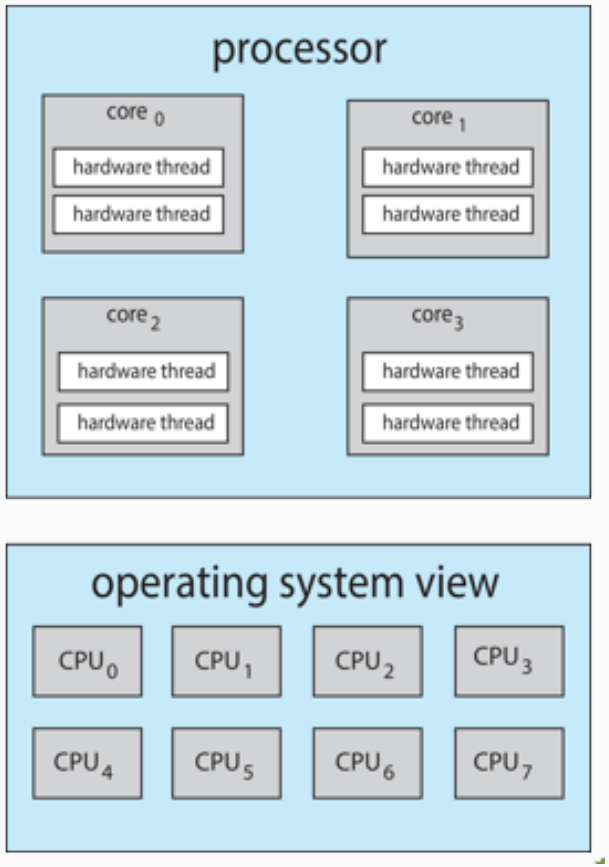


Figure 9: Multithreaded Multicore

Algoritmos de Itineración - Multiprocesamiento

- Para hebras, dos niveles de itineración.
  - ✓ El SO decidiendo a qué hebra asigna una CPU lógica.
  - ✓ Cada núcleo decide qué hardware thread correr en el chip físico.

Algoritmos de Itineración - Multiprocesamiento

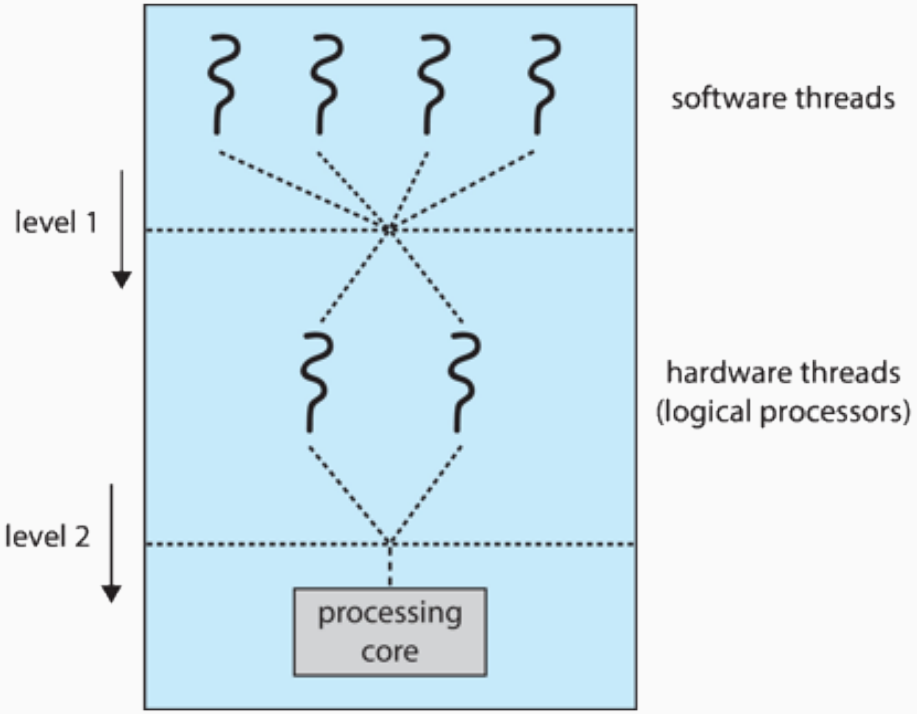


Figure 10: Multithreaded Multicore

Algoritmos de Itineración - Multiprocesamiento

- Objetivo: Mantener todas las CPU's ocupadas.
- Carga balanceada de tareas:
  - ✓ **Push:** Periodicamente buscar procesadores sobrecargados de tareas y migrar tareas a procesadores menos cargados o libres.
  - ✓ **Pull:** Procesadores disponibles pueden solicitar tareas a procesadores ocupados.

Algoritmos de Itineración - Multiprocesamiento

- Si una hebra se ejecuta, su información queda en la caché.
- **Afinidad:** Si una hebra es itinerada, volverá al mismo procesador.
  - ✓ **Soft:** El SO se preocupa de lograrlo, pero no lo garantiza.
  - ✓ **Hard:** Los procesos se ejecutan en ciertos procesadores.
- Esto mejora el desempeño.
- La carga balanceada puede afectar la afinidad de una hebra.

Sistemas Operativos

Planificación de uso de CPU

Viktor Andrés Tapia Vásquez

Segundo Semestre 2021

Departamento de Informática, Campus SSJJ.