Sistemas operativos avanzados

Planificación del procesador

5ª parte: Planificación de aplicaciones paralelas y distribuidas

Contenido

- Planificación de aplicaciones paralelas en multiprocesadores
- Planificación en sistemas distribuidos
- Planificación de aplicaciones paralelas en sistemas distribuidos

Planificación de aplicaciones paralelas en MP

- Aspecto vinculado con el campo del paralelismo
  - Básicamente ajeno al SO
  - Implementado normalmente por entorno de ejecución paralelo
  - Presentación sólo da una visión general
- Aplicación paralela AP formada por múltiples procesos/*threads*
  - No finaliza hasta que no termine el último
- Alto grado de paralelismo e interacción
  - Procesos/*threads* de una AP deberían ejecutar simultáneamente
- En el sistema hay un conjunto de APs con ejecución *batch*
  - Uso de un planificador a largo plazo
- 2 técnicas más frecuentes
  - *Space sharing*
  - *Gang scheduling*
- Temas abiertos: asignación de UCPs a AP en MP jerárquico

Space sharing

- Multiplexación en el espacio de las APs
- Se asigna un conjunto de procesadores a cada AP
  - Una UCP dedicada a cada proceso/*thread*
    - Uso de primitivas de afinidad estricta del SO para la reserva
  - Ejecución simultánea: interacción muy rápida
  - Sin sobrecarga por cambios de contexto
- Planificador a largo plazo controla la entrada de trabajos
  - AP declara cuántas UCPs requiere y espera entrada al sistema
  - Planificador asigna UCPs a APs siguiendo una política dada
    - FCFS, SJF (requiere estimación de tiempo), prioridad, EDF,…
  - *backfilling*: UCPs disponibles no satisfacen a AP 1º en cola
    - Se "cuela" otra AP pero garantizando no inanición de la 1ª
- Extensión para servidor paralelo: asignación dinámica
  - Servidor puede ajustar grado de paralelismo a UCPs disponibles

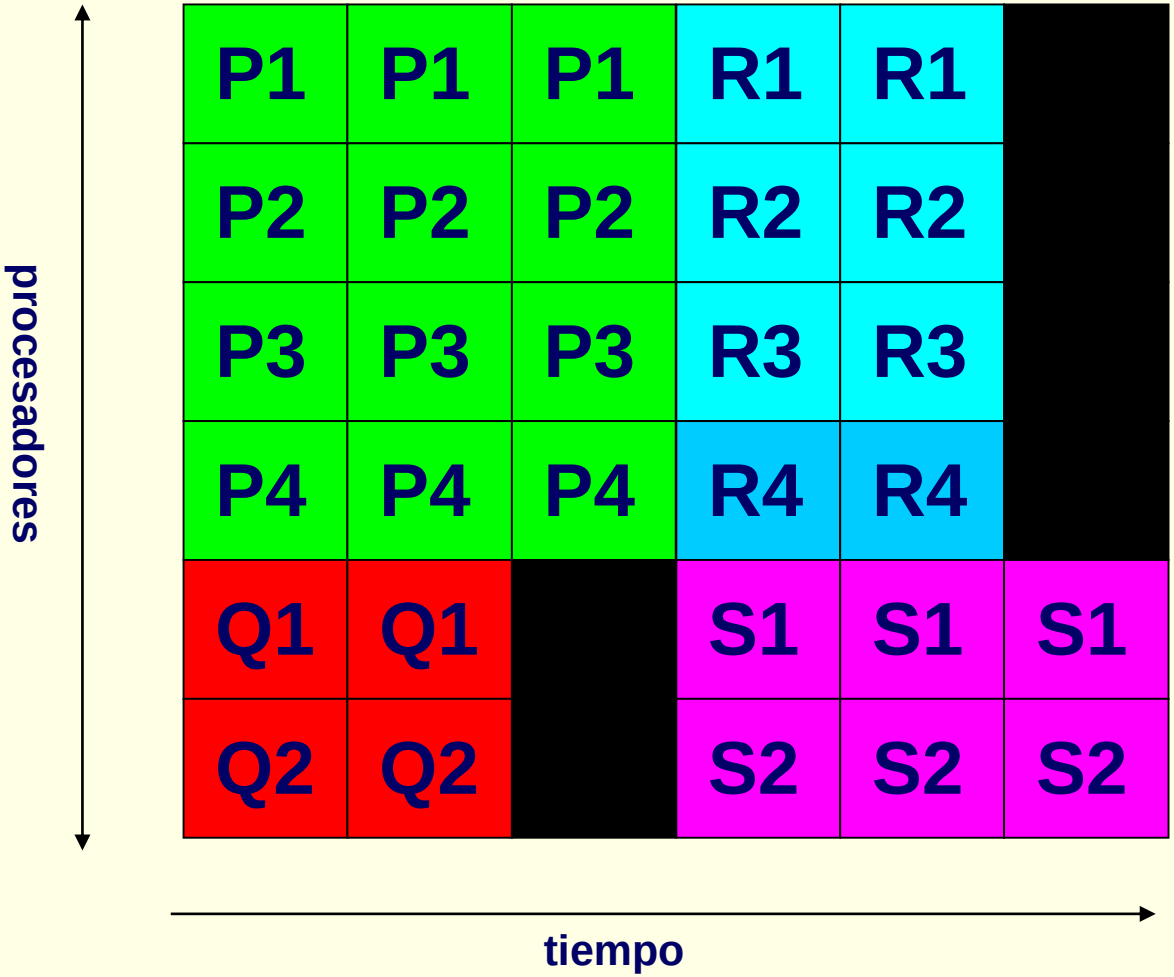
Gang scheduling

- Multiplexación en el espacio y en el tiempo de las APs
  - Procesos/*threads* de una AP ejecutan simultáneamente
  - Pero no tienen procesadores dedicados
  - En cada rodaja se ejecutan todos los proc/*threads* de varias APs
    - Dificultad/ineficiencia implementar una planificación sincronizada
  - Sobrecarga por cambios de contexto involuntarios
  - Reduce tiempo de espera de las APs para entrar al sistema
    - Aunque alarga su tiempo de ejecución
- Planificador controla la asignación de UCPs a APs
  - AP declara cuántas UCPs requiere
  - Planificador se basa en una matriz de Ousterhout
    - Rodajas X Procesadores
    - O[i,j]: qué proceso/*thread* ejecuta en UCP i durante rodaja j

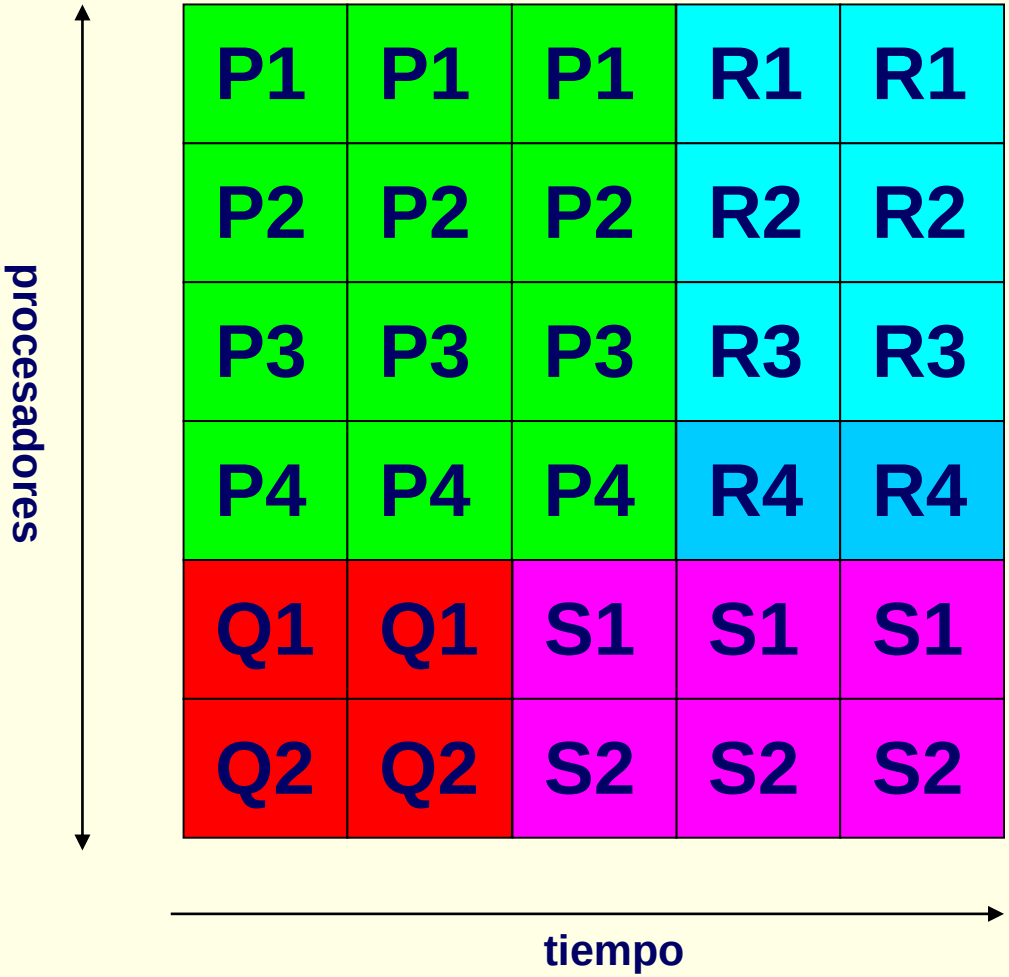
Ejemplo de planificación de APs en MP

- Sistema MP con 6 procesadores
- 4 APs por orden de prioridad (o llegada)
  - P requiere 4 UCPs y dura 3 unidades
  - Q requiere 2 UCPs y dura 2 unidades
  - R requiere 4 UCPs y dura 2 unidades
  - S requiere 2 UCPs y dura 3 unidades
- 3 estrategias:
  - *Space sharing* sin *backfilling*
  - *Space sharing* con *backfilling*
  - *Gang scheduling*

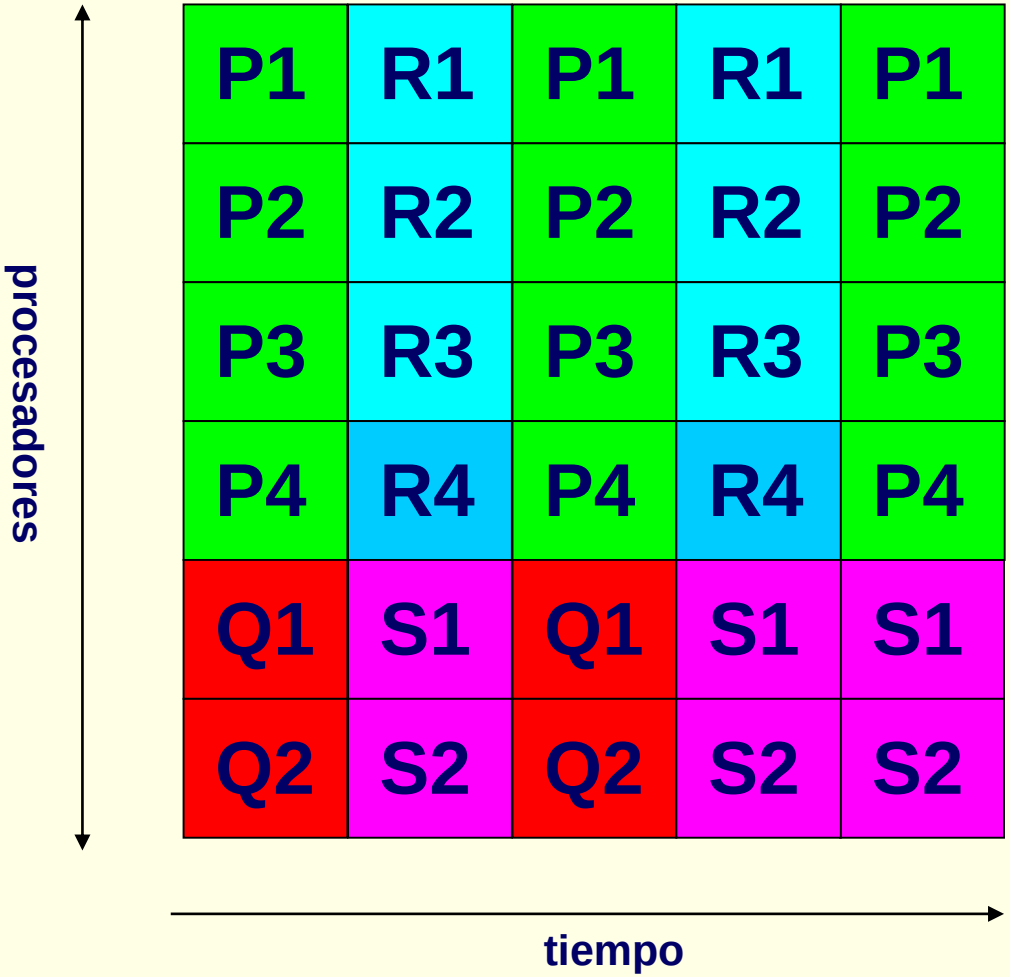
Ejemplo: *Space sharing* sin *backfilling*



Ejemplo: *Space sharing* con *backfilling*



Ejemplo: *Gang Scheduling*



Planificación en sistemas distribuidos

- Aspecto vinculado con el campo de los sistemas distribuidos
  - Básicamente ajeno al SO
  - Implementado normalmente por *middleware*
  - Presentación sólo da una visión general
- Falta de memoria compartida condiciona planificación en SS.DD.
  - Equilibrio de carga → migración (como con MP y cola/UCP)
    - Pero sin memoria compartida: costosa y técnicamente compleja
      - Incluso no factible en algunos sistemas
    - Necesario migrar mapa memoria y recursos asociados al proceso
- Asignación del procesador inicial a un proceso
  - Ejecución en la UCP donde se crea
  - Ejecución remota para reparto de carga
    - Más sencilla que migración
    - Problemas en sistemas heterogéneos

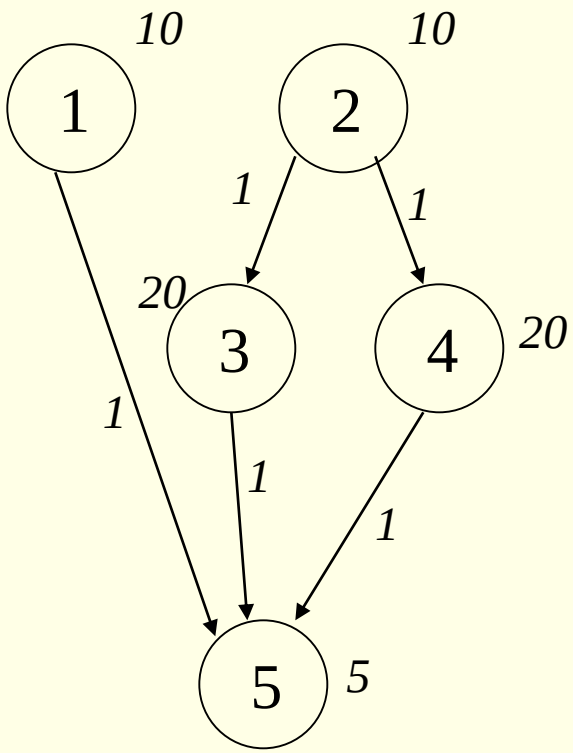
Estrategias de equilibrado de carga

- Iniciada por el emisor: emisor busca receptores
  - Nodo sobrecargado pide ayuda
    - P.e. a un conjunto aleatorio de nodos
  - Envía un proceso al nodo seleccionado (p.e. menos cargado)
    - Mejor nuevo: Ejecución remota, no requiere migración
  - Sobrecarga peticiones ayuda inútiles si sistema muy cargado
- Iniciada por el receptor: receptor solicita procesos
  - Nodo descargado ofrece ayuda
    - P.e. a un conjunto aleatorio de nodos
  - Pide un proceso al nodo seleccionado (p.e. más cargado)
    - Requiere migración
  - Sobrecarga ofertas inútiles si sistema poco cargado (- grave)
- Simétrico: combinación de las anteriores

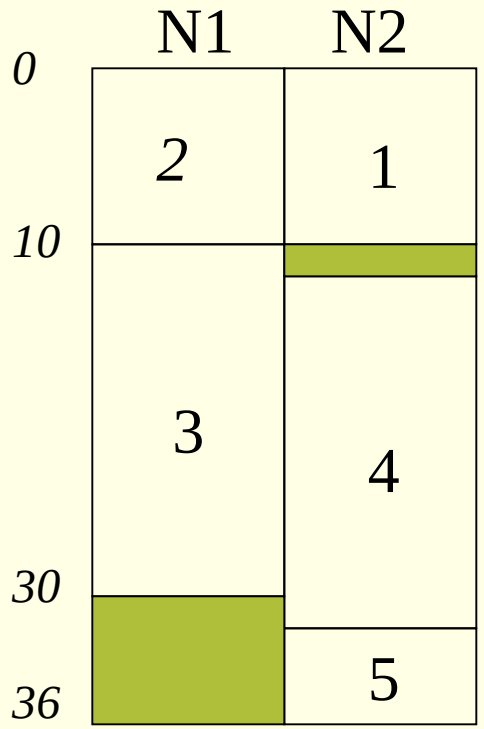
Planificación de aplicaciones paralelas en SS.DD.

- Maximizar paralelismo: Esquema con 1 proceso de la AP/nodo
  - Gestión similar a *Space sharing* en MP
    - Planificador a largo plazo controla la entrada de trabajos
    - AP declara cuántas UCPs requiere y espera entrada al sistema
    - Planificador asigna UCPs a APs siguiendo una política dada
      - FCFS, SJF (requiere estimación de tiempo), prioridad, EDF,…
    - *backfilling*: UCPs disponibles no satisfacen a AP 1º en cola
      - Se "cuela" otra AP pero garantizando no inanición de la 1ª
- Esquema con múltiples procesos de la AP/nodo
  - Asignación estática de conjunto de procesos a nodos
    - Maximizando paralelismo y minimizando comunicación
  - Depende de la arquitectura de la AP. Ejemplos:
    - Modelo basado en precedencias: grafo acíclico dirigido (DAG)
    - Modelo basado en comunicaciones: grafo no dirigido

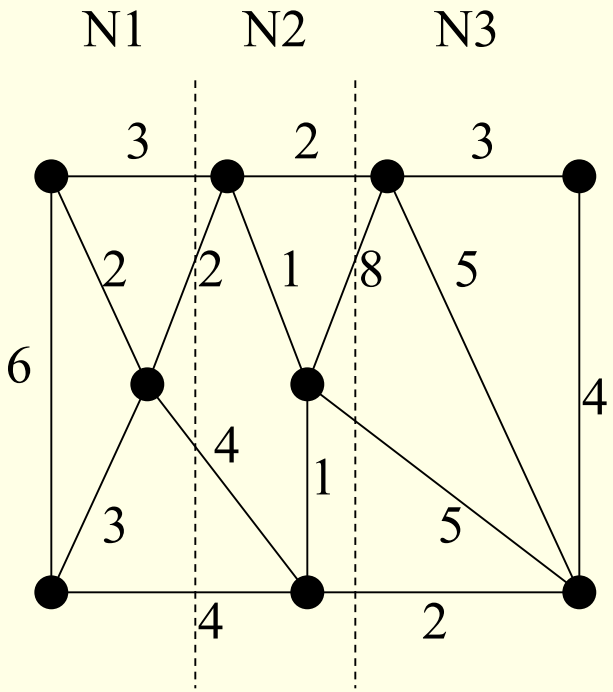
Ejemplo modelo precedencia de tareas (DAG)



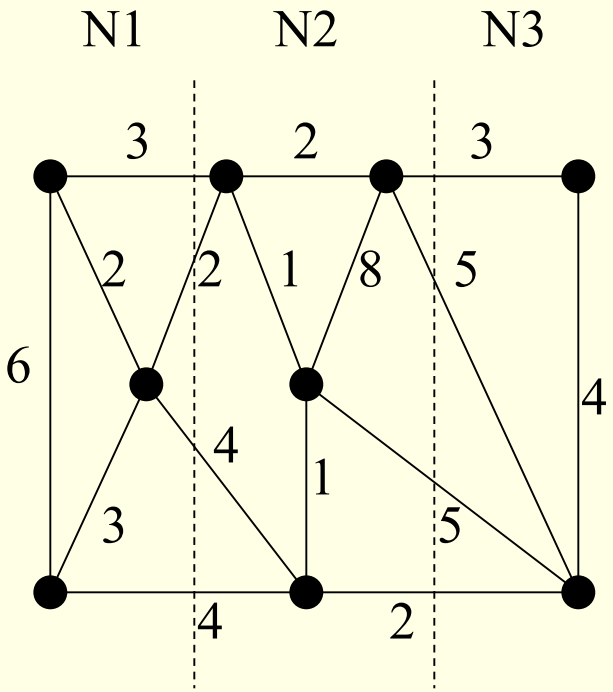
Planificador



Ejemplo modelo basado en comunicaciones



Tráfico entre nodos:
13+17=30



Tráfico entre nodos:
13+15=28

Tanenbaum. "Sistemas Operativos Distribuidos"