Teoria 3 - Planificacion de Procesos

Definiciones

- Mecanismo: técnica para lograr un resultado.
- Política: uso particular de mecanismos que satisfaga ciertos criterios. Ej: prioridades.
- Productividad/Rendimiento (throughput): número de trabajos por unidad de tiempo.
- Tiempo de paso/retorno (turnaround): tiempo entre que se lanza un proceso y termina.
- Capacidad de ejecución: mantener la CPU ocupada de forma constante.
- Justicia (fairness): que un procesos obtenga un tiempo de CPU "justo" o razonable.
- Equilibrio: que todas las partes del sistema estén ocupadas.
- Tiempo de respuesta: tiempo desde que el usuario da una orden y se obtiene respuesta a la misma.
- Proporcionalidad: que las tareas sencillas demanden poco tiempo de respuesta.
- Fiabilidad: no perder datos, reaccionar en un tiempo límite, etc.

Planificación

En un sistema multiprogramado generalmente existirán varios procesos al mismo tiempo requiriendo el procesador. El **planificado** (scheduler) decide *qué* proceso ejecutar a continuación y *cuándo* cambiar el proceso en ejecución.

El **despachador** (dispatcher) es el módulo del SO que delega el control de la CPU al proceso seleccionado por el planificador. Esto, como ya vimos, implica un cambio de contexto. Hay que tener en cuenta que este cambio de contexto consume tiempo de CPU que no se le dedicará a los programas del usuario, por lo que *cuanto más se llame al dispatcher, menos tiempo se dedicará a los programas de usuario*.

La planificación (scheduling) es el **conjunto de políticas y mecanismos del SO que gobiernan la forma en que los procesos llegan a ejecutarse**.
Existen tres niveles de planificación:

- Planificación a largo plazo: planificador de trabajo. Decide qué procesos entran al sistema y cuáles deben esperar.
- Planificación a mediano plazo: planificador de swapping. Controla la cantidad de procesos en memoria principal (RAM). A nivel estado, los procesos que maneja son los que están en Listo o Bloqueado, sacandolos o enviándolos a la memoria principal
- Planificación a corto plazo: planificador de CPU. Controla qué procesos (en estado Listo) ejecuta el CPU. A nivel proceso, maneja la transición entre Listo y Ejecutándose.

Nos centraremos en la planificación a corto plazo.

Criterios de planificación

Aunque el objetivo de la planificación es mejorar el rendimiento general del sistema y dar buen servicio a todos los procesos, hay ciertos criterios que pueden orientarse a dar un mejor servicio al usuario y otros que se orientan al sistema. Los primeros se relacionan con el comportamiento del sistema como lo percibe el usuario o un proceso, mientras que los orientados al sistema se centran en el uso efectivo y eficiente del procesador.

- Orientados al usuario, por prestaciones
  - Tiempo de estancia/ejecución (turnaround time): tiempo total entre que se lanza un proceso y finaliza.
  - Tiempo de respuesta (responce time): tiempo desde la petición hasta que se empieza a recibir respuesta.
  - Fecha tope (deadline): cuando se puede especificar un plazo límite para un proceso. El planificador debe maximizar el porcentaje de estas para cumplir con los plazos establecidos.
- Otros orientados al usuario
  - Previsibilidad: un proceso debería ejecutarse con el mismo tiempo y coste a pesar de la carga del sistema.
- Orientados al sistema, por prestaciones
  - Rendimiento (productividad): se debe maximizar el número de procesos completados por unidad de tiempo.
  - Utilización de CPU (eficiencia): porcentaje de tiempo que se ocupa el procesador (cuanto mayor, mejor).
- Otros orientados al sistema
  - Equidad: si no hay prioridad por el usuario o el sistema, todos los procesos deben ser tratados igual.
  - Imposición de prioridades: deben ejecutarse más rápidamente los procesos de mayor prioridad (en caso de establecerse prioridades).
  - Equilibrio del uso de recursos: se debe obtener un mayor rendimiento en el uso de recurso al estar estos ocupados de manera equitativa el mayor tiempo posible (se debe favorecer a los procesos que hagan poco uso de recursos sobreutilizados).

Medidas de tiempo

- Tiempo de **respuesta** (latency / response time): tiempo en que el proceso *comienza* a responder, **no** el tiempo que tarda en enviar toda la respuesta.
- Tiempo de **servicio** o ejecución o ráfaga de CPU: tiempo que consumiría si fuese el único proceso existente, sin contar el tiempo de carga/espera.

- Tiempo de **paso**, de **estancia** o de **retorno** (turnaround): tiempo total transcurrido desde que se incia hasta que se finaliza. Incluye el tiempo de carga en memoria, tiempo en cola de listos, tiempo de ejecución en CPU y tiempo de operaciones de E/S (bloqueado).
- Tiempo de **espera**: tiempo de retorno quitando CPU y E/S.
- Índice de servicio: relación entre el tiempo de servicio y el tiempo de retorno.
- Justicia (Fairness): relación entre el tiempo de espera y el tiempo en CPU.

CPU y I/O Burst

Se denomina CPU Burst (ráfagas de CPU) a los interválos en los que "trabaja" la CPU en un proceso, mientras que los I/O Burst (ráfagas de E/S) son aquellos en los que se envían y/o traen datos. Los procesos alternan entre estos dos estados durante su ciclo de vida, empezando por un CPU Burst.
Un proceso que invierte la mayor parte del tiempo se dice que es **CPU-bound**, mientras que, por el contrarios, los que invierten más tiempo en operaciones de E/S son I/O-bound. La prioridad de un proceso será inversamente proporcional al uso que haga del CPU.

Planificación apropiativa y no-apropiativa

Existen 4 circunstancias en las que es necesario tomar decisiones sobre qué proceso debe continuar su ejecución:

1. Cuando un proceso cambia de Ejecución a Bloqueado (por ejemplo, por una solicitud de E/S).
2. Cuando un proceso cambia de Ejecución a Listo (por ejemplo, por una interrupción o finalización de quantum de tiempo).
3. Cuando un proceso cambia de Bloqueado a Listo.
4. Cuando un proceso termina.

En **1** y **4**, la única opción es seleccionar un nuevo proceso (si existe alguno en la cola de listos).
En **2** y **3** tenemos la opción de planificar un nuevo proceso o no. En caso de que el esquema contemple la planificación de un nuevo proceso, se dice que es **apropiativo**

Planificación No-Apropiativa

También llamada multitasking o sin desalojo. En este esquema una vez que se le otorga la CPU a un proceso, no se le puede retirar hasta que termina o se bloquea (por solicitudes de E/S).

Esto trae como consecuencia que

- Los trabajos largos hacen esperar a los trabajos cortos.
- Se trata a los procesos con más equidad.
- Los tiempos de respuesta son más predecibles, ya que un proceso nuevo no puede desplazar a los procesos en espera de menor prioridad.

Suele utilizarse en procesos por lotes (batch) como inventarios, cálculos de nóminas, impresión de listados, etc.

Planificación Apropiativa

También llamada "planificación por torneo". En este esquema un proceso en ejecución puede ser interrumpido y pasado a la cola de listos. Esta decisión puede deberse a la llegada de un nuevo proceso, la aparición de una interrupción o de forma periódica.

Características

- Los procesos de mayor prioridad tendrán atención más rápida.
- Mejores tiempos de respuesta en sistemas de tiempo compartido.
- Tiene un costo en recursos. Requiere intercambios más frecuentes de contexto y mantener muchos procesos en almacenamiento principal esperando la CPU, lo que implica sobrecarga.

Algorítmos de Planificación

A la hora de decidir qué proceso de la cola de listos debe asignarse a la CPU, existen varios algorítmos:

- FIFO (First-In First-Out) o FCFS (First-Come First-Served): asigna de acuerdo al orden de llegada.
  - El el más simple.
  - Es no-apropiativo.
  - El tiempo de espera promedio es alto.
- SJF (Shortest Job First): asocia a cada proceso el largo de su CPU-burst, para así asignar a CPU el proceso con menor CPU-burst. En caso de empate, se usa el orden de llegada.
  - Es no-apropiativo.
  - Es un caso particular de planificación por prioridades.
  - Se deben conocer de antemano los tiempos de ejecución.
  - El tiempo medio de espera es el menor posible para un conjunto de procesos dado.
- SRTF (Shortest Remaining Time First): variante apropiativa de SJF. Si aparece un proceso con menor CPU burst, reemplaza al que se está ejecutando.
  - Es apropiativo
- Prioridades: las prioridades pueden definirse de forma interna (limite de tiempo, requisitos de memoria, archivos abiertos, etc) o externa al SO (importancia, costo, etc). Por general, suele ser también apropiativa.
  - El principal problema es la muerte por inanición o bloqueo indefinido de los procesos con baja prioridad. Una solución podría ser un mecanismo de envejecimiento que aumente gradualmente la prioridad de los procesos que esperan.

- Round Robin (RR) o Planificación por Turnos: asigna de acuerdo al orden de llegada, como FCFS (FIFO), pero asigna además un intervalo de tiempo (time quantum) límite. Una vez cumplido el intervalo, el proceso es desalojado y puesto al final de la cola.
    - Es apropiativo.
    - Es simple de implementar.
    - Es el más justo, ya que todos los procesos reciben una parte de CPU.
    - La elección de quantum es importante, uno pequeño generará muchos cambios de contexto y afecterá el rendimiento, mientras que con uno muy grande se comportará como un FCFS.
- Colas múltiples o multinivel (multilevel queue): divide la lista de procesos listos en varias colas, de acuerdo al tipo, por lo que a la hora de asignar un proceso a CPU se elije primero una cola y luego un proceso dentro de la misma. Se aplican dos niveles de planificación, por un lado dentro de cada cola, y por otro lado entre las colas.
    - Es apropiativa.
    - Los procesos no pueden cambiar de cola.
    - Por lo general la planificación entre colas se hace con prioridad y quantum de tiempo, que se distribuirá entre los procesos de la misma.
- Colas multinivel con Retroalimentación (multilevel feedback queue): una variante de las colas múltiples donde los procesos pueden moverse entre colas, es una forma de implementar el envejecimiento. Un proceso que está mucho tiempo en espera pasa a otra de mayor prioridad.
    - Es apropiativa.
    - Es el algorítmo más usado en los SO.
    - Logra equidad, tiempo de respuesta y baja sobrecarga.
    - Es complejo de implementar.