PLANIFICACIÓN DEL PROCESADOR

Planificación de múltiples procesadores.

El problema de planificación se vuelve más complejo. Se han ensayado muchas posibilidades, tales como:

- Tener una cola aparte para cada procesador; sin embargo, un procesador podría estar ocioso, con su cola vacía, mientras otro está muy ocupado.
- Tener una cola común para los procesos listos. Los procesos ingresan en una cola y se les asigna cualquier procesador que esté disponible. Existen 2 estrategias para este caso:
    - Cada procesador se auto - planifica. El procesador examina la cola común de procesos listos y escoge el proceso que ejecutará. Como varios procesadores tratan de acceder a una estructura de datos común y actualizarla es preciso programar cada procesador con mucho cuidado.

    - Para evitar los problemas de la estrategia anterior se nombra a un procesador como planificador para los demás procesadores, lo que da lugar a una estructura maestro - esclavo. Una variante es que el procesador maestro tome todas las decisiones de planificación y se encargue del procesamiento de E/S y otras actividades del sistema. Los demás procesadores ejecutan código de usuario. Este se conoce como multiprocesamiento asimétrico, porque sólo un procesador accede a las estructuras de datos del sistema y reduce la necesidad de compartir recursos.

Planificación en tiempo real.

La computación en tiempo real se divide en dos tipos. Los sistemas de tiempo real duro deben completar una tarea crítica dentro de un lapso garantizado. En general los procesos se presentaran junto con una declaración de la cantidad de tiempo en que necesita terminar o realizar E/S. Entonces, el planificador admitirá el proceso, garantizando que terminará a tiempo, o bien rechazará la solicitud por ser imposible. Esto se denomina *reservación de recursos*. Requiere que el planificador sepa con exactitud cuanto tarda en ejecutarse cada tipo de función del SO y por lo tanto debe garantizarse que cada operación tardará cuando más cierto tiempo máximo. Una garantía así es imposible en un sistema con almacenamiento secundario o memoria virtual porque esos subsistemas causan una variación impredecible e inevitable en el tiempo que tarda en ejecutarse un proceso particular. Por lo tanto los sistemas de tiempo real consisten en software de propósito especial que se ejecuta en hardware dedicado a su proceso crítico, y carecen de la plena funcionalidad de las computadoras y sistemas operativos modernos.
La computación en tiempo real blando es menos restrictiva; requiere que los procesos críticos tengan mayor prioridad que otros menos afortunados.
La implementación de una funcionalidad de tiempo real blando requiere un diseño cuidadoso del planificador y aspectos relacionados del SO. En primer lugar el sistema debe contar con planificación por prioridad y los procesos de tiempo real deben tener la prioridad más alta y no deben degradarse con el tiempo. En segundo lugar la latencia de despacho debe ser pequeña. Cuanto menor es la latencia, mayor es la rapidez con que un proceso de tiempo real puede comenzar a ejecutarse una vez que está listo.
Para asegurar la primera propiedad, se puede eliminar el envejecimiento de procesos cuando hay procesos de tiempo real, esto asegura que la prioridad de los distintos procesos no cambie.
La segunda es más complicada, porque muchos SO están obligados a esperar a que se complete una llamada al sistema o bien a que ocurra un bloqueo por E/S antes de efectuar una conmutación de contexto. La latencia suele ser larga ya que las llamadas al sistema son complejas y los dispositivos de E/S son lentos.
Necesitamos que en las llamadas se puedan insertar puntos de expropiación donde se verifica si es preciso ejecutar algún proceso de alta prioridad.

Otro método es tener el núcleo expropiable, de esta forma el núcleo siempre puede ser desalojable, ya que cualquier dato del núcleo esta protegido contra actualización de cualquier proceso de alta prioridad. Ej. Solaris 2. Si un proceso de alta prioridad necesita leer o modificar datos del núcleo al que esta accediendo un proceso de baja prioridad se da una situación de *inversión de prioridad*, entonces se utiliza el *protocolo de herencia de prioridad* en el que todos los procesos que están accediendo a ese recurso heredan la prioridad alta hasta que terminan y luego recuperan su valor original.

Evaluación de algoritmos.

Para seleccionar un algoritmo debemos primero fijar la importancia relativa de nuestras medidas:

- Maximizar el aprovechamiento de la CPU bajo restricción de que el tiempo de respuesta máximo sea de un segundo.
- Maximizar el rendimiento de modo tal que el tiempo de retorno (promedio) sea linealmente proporcional al tiempo de ejecución total.

El siguiente paso es evaluar los diversos algoritmos que se estén considerando. Hay varios métodos de evaluación distintos, entre ellos:

- ***Modelado determinista.***
  Es un tipo de evaluación analítica que toma una carga de trabajo predeterminada específica y define el desempeño del algoritmo para esa carga de trabajo.
  Ventajas:
  Es sencillo y rápido; da números exactos, lo que permite comparar los algoritmos, pero requiere números exactos como entrada, y sus respuestas sólo se aplican a esos casos.
  Los usos principales son describir los algoritmos de planificación y proporcionar ejemplos. En un conjunto de ejemplos, podría indicar tendencias que después pueden analizarse y demostrarse por separado.
  Desventajas:
  Es demasiado específico y requiere demasiados conocimientos exactos, para ser útil. Los procesos que se ejecutan en muchos sistemas varían de un día a otro, de modo que no hay un conjunto estático de procesos que podamos usar.

- ***Modelos de colas.***
  Consiste en determinar las distribuciones características del sistema, tales como: la distribución de las ráfagas de CPU y E/S (comúnmente exponencial) y la distribución de los tiempos en que los procesos llegan al sistema. A partir de estas distribuciones es posible calcular el rendimiento promedio, el aprovechamiento, el tiempo de espera, etc. para la mayor parte de los algoritmos. Los modelos consisten asociar a cada recurso colas de procesos en espera. Conociendo frecuencias de llegada y rapidez de servicio es posible calcular el aprovechamiento, la longitud de colas, el tiempo de espera promedio, etc.
  Ventajas y desventajas:
  Puede servir para comparar algoritmos de planificación, pero tiene sus limitaciones. Las clases de algoritmos y distribuciones que pueden manejarse son relativamente limitadas y puede resultar difícil trabajar con las matemáticas de algoritmos o distribuciones complicados. Las distribuciones a menudo se definen de formas poco realistas pero matemáticamente manejables. También es necesario hacer varios supuestos independientes, que podrían no ser perfectamente válidos. A menudo los resultados de los modelos son aproximaciones del sistema real.

- ***Simulaciones.***
  Implica programar un modelo del sistema de computación. El simulador tiene una variable que representa un reloj; cuando se incrementa el valor de esta variable, el simulador modifica el estado del sistema de

modo que refleje las actividades de los dispositivos, los procesos y el planificador. Conforme se ejecuta la simulación, se recopilan e imprimen los datos estadísticos que indican el desempeño del algoritmo.
Los datos que se alimentan a la simulación se pueden generar de varias maneras. El método más común emplea un generador de números aleatorios. Otra forma es a través de cintas de rastreo que se crean vigilando el sistema real y registrando la secuencia de sucesos reales.
Ventajas:
Son una forma excelente de comparar dos algoritmos utilizando exactamente el mismo conjunto de entradas reales.
Desventajas:
Suelen ser costosas y requieren horas de tiempo de computadora. Las cintas de rastreo pueden requerir grandes cantidades de espacio de almacenamiento. El diseño la codificación y depuración del simulador no es una tarea trivial.

- ***Implementaciones.***
  La única forma exacta de evaluar un algoritmo de planificación es codificarlo, colocarlo en el SO y ver como funciona. El principal inconveniente es el costo. Es necesario codificar el algoritmo y modificar el SO para que lo apoye, crear las estructuras de datos que requiere y ver la reacción de los usuarios ante un SO que cambie constantemente. Otro inconveniente de cualquier evaluación de algoritmos es que el entorno en el que se usa el algoritmo cambiará no sólo cuando se escriben nuevos programas y los tipos de problemas cambian sino también por el desempeño del planificador.
  Existen algoritmos de planificación más flexibles que permiten los administradores o usuarios alterarlos o modificar las variables que utilizan.