Planificador en sistemas multiprocesador

En un multiprocesador, la planificación involucra:

1. **Asignación de procesos a procesadores**. El planificador debe tratar cada proceso como un recurso colectivo y asigna procesos a procesadores según lo demanden. Esta asignación puede ser estática o dinámica. Si un proceso se asigna a un procesador hasta que concluya(estática), se crea una cola para este procesador. Así, se sobrecarga menos la función de planificación. Sin embargo, esto implica que un procesador pueda estar con la cola vacía y otro con trabajo acumulado. Para ello, se puede usar una cola común a todos los procesadores, así un proceso puede ser ejecutado sobre varios procesadores en varios momentos de tiempo diferentes. Si esto es así, la información de contexto de todos los procesos debe estar disponible para todos los procesadores. Otra opción más es la que utiliza Linux, que balancea dinámicamente la carga de procesos y así los hilos se mueven de una cola de un procesador a una cola de otro procesador.

De todos modos, necesitamos una forma de asignar procesos a procesadores. Tenemos dos maneras principales:

- Maestro/esclavo, con la que las funciones claves del núcleo de S.O se ejecutan siempre en un mismo procesador y los demás ejecutan programas de usuario. El maestro planifica los trabajos y los esclavos solicitan al maestro los servicios. Sin embargo, un fallo en el maestr hace que falle el sistema y el maestro puede ser un cuello de botella para el rendimiento del sistema.
- Camaradas, cualquier procesador puede ejecutar el núcleo y cada procesador se auto-planifica. Complica el sistema operativo pues se debe asegurar que dos procesadores no escojan el mismo proceso.

2. **Uso de la multiprogramación en procesadores individuales** En los multiprocesadores tradicionales, cada procesador individual debe ser capaz de cambiar entre procesos para conseguir un buen rendimiento. No obstante, para aplicaciones de mediano tamaño ejecutando en un multiprocesador es más complejo, pues hay que conseguir que cada procesador rinda al máximo y eso puede ser difícil para un monoprocesador por ejemplo si la aplicación consta de varios hilos que deben ejecutarse a la vez.
3. **Activación de procesos** O la elección del proceso a ejecutar. En un monoprocesor multiprogramado se intenta dar prioridad a procesos para mejorar el rendimiento. Si tenemos un sistema multiprogramado, esto es innecesario y puede ser hasta contraproducente.

Planificación de procesos

En los sistemas multiprogramados tradicionales, los procesos están en una única cola prar los procesadores o múltiples colas con prioridad. (Hay un ejemplo muy largo)

Planificación de Hilos

En los hilos, la ejecución se separa de la definición de un proceso. Una aplicación se puede implementar como un conjunto de hilos que trabajan juntos en el mismo espacio de direcciones. En un monoprocesador, los hilos son ayuda a estructurar programas y a facilitar los cambios entre hilos. En un multiprocesador trabajan paralelamente y si esto ocurre mejora mucho el rendimiento. Hay varias propuestas para la planificación multiprocesador de hilos:

- Compartición de carga: con una cola global de hilos listos. Así, la carga se distribuye uniformemente y no se necesita un planificador centralizado. Se usa un sistema **FCFS** (Primero en llegar, primero en ser servido). Además, la cola se organiza con prioridad dándosela a los hilos de los trabajos con menor número de hilos. Además, si llega un trabajo con menor número de hilos que un trabajo en ejecución, se epulsan los hilos pertenecientes al trabajo planificado. Sin embargo, también tine desventajas como la memoria que ocupa la cola, que es poco probable que los hilos expulsados se ejecuten en el mismo procesador y que si los hilos se tratan como un conjunto de hilos es probable que todos los hilos ganen acceos a procesadores a la vez y esto se convierta en un problema.
- Planificación en pandilla: varios hilos relacionados se planifican para ejecutarse a la vez sobre un conjunto de procesadores. Tiene beneficios como que si se ejecutan en paralelo procesos muy relacionados puede reducirse el bloqueo por sincronización y se necesiten menos cambios de proceso o que la sobrecarga de la planificación se reduzca. Con este **método** se reducen os cambios de proceso
- Asignación de procesador dedicado: Opuesto a compartición de carga, cada proceso tiene un número de procesadores igual al número de hilos en su programa. Como vnetajas tienq eu es altamente paralelo y que evita el cmabio de proceso mejorando la velocidad del programa.
- Planificación dinámica: el número de hilos de un proceso puede variar durante su ejecución. Aquí, la planificación del SO se limita a la ubicación del procesador y se realiza de forma que cuando un trabajo solicita algún procesador:

a. Se le dan si se puede

b. Si el trabajo que realiza la solicitud acaba de llegar se ubica quitándole un procesador a otro trabajo que tenga más de uno
c. Si no se puede procesar la solicitud, se mantiene pendiente hasta que haya un procesador disponible.

Al liberarse los procesadores, se examina la cola de solicitudes insatisfechas y se asigna un único procesador a cada trabajo en la lista que no tenga ninguno asignado.

Planificación en sistemas de tiempo real

Este computación se define como aquella en la que la correción del sistema depende del resultado lógico y del momento en el que se producen los resultados. Las tareas tienen un grado de urgencia, reaccionan a eventos del exterior, y por ello se les asocia un tiempo límite. Se clasifican en: * Tareas de tiempo real duro, que deben cumplirse en su plazo límite o se produce un error fatal en el sistema * Tarea de tiempo real suave, que tiene un plazo deseable asociado pero no obligatorio.

Características de los SO de tiempo REAL

En áreas generales, estos SO tienen 5 requisitos: 1. Determinismo Esto indica que realiza las operaciones en instantes de tiempo fijos o dentro de intervalos predeterminados. El determinismo depende de la velocidad de responder a una interrupción y de manejo de solicitudes en el tiempo requerido 2. Reactividad Se preocupa de cuánto tiempo tarda el SO de servir la interrupción después de haberla detectado. Incluye por ello el tiempo de comenzar a ejecutar la rutina de servicio de la interrupción (RSI), la cantidad de tiempo en realizar la RSI y el efecto de anidamiento de interrupciones.

3. Control del usuario En los SO de tiempo no real, o el usuario no tiene control sobre la planificación o el SO da una guía burda. En los SO de tiempo real, es esencial darle al usuario un buen control sobre la prioridad de las tareas. El usuario puede distringuir entre tareas duras y suaves y dar prioridad entre cada clase. En tiempo real el usuario también podrá decidir qué procesos viven en memoria principal o qué algoritmos de transferencias a disco usar.
4. Fiabilidad Es más importante para los SO de tiempo real, pues un fallo en tiempo no real puede solventarse reiniciando el sistema pero podría romperse el procesador.
5. Operación de fallo suave Se refiere a la habilidad del sistema de fallar de forma que se siga conservando la maxima capacidad y datos como sea posible. Un aspecto importante es la estabilidad, que se refiere a que

cuando no se puedan cumplir los plazos de todas las tareas, se cumplan al menos los de las tareas prioritarias.

Para cumplir esos requisitos, los SO de tiempo real tienen rapidos cambios de proceso, capacidad de respuesta a interrupciones externas, multitarea, minimización de intervalos durante los cuales se deshabilitan las interrupciones.

La mayoría de SO de tiempo real son incapaces de tratar con plazos límite, pero pueden responder de forma que cuando se aproxime un plazo de tiempo la tarea se planifica rápidamente.

Planificación de Tiempo Real

Los enfoques de la planificación dependen de cuándo el sistema realiza análisis de planificabilidad, de si la planificacion se realiza estática o dinámencamente y de si el resultado del análisis produce un plan de planificació de acuerdo al cual se desarrollarán las tareas en tiempo de ejecución. Siguiendo eso, hay varios algoritmos:

- Estáticos dirigidos por tabla: El resultado es una planificación que determina cuándo debe comenzar a ejecutarse cada tarea. Esta se aplica a tareas periódicas. El planificador intenta encontrar un plan que permita cumplir todos los requisitos de estas. Es un enfoque previsible pero no flexible, pues un cambio en los requisitos requiere rehacer toda la planificación.
- Estáticos expulsivos por prioridad: No se obtiene una planificación, se asigna una prioridad a las tareas y asi el planificador expulsivo tradicional se puede usar basandose en las prioridades. No es de tiempo real, puede haber varios factores que determinen la prioridad, como el consumo de CPU o E/S.
- Dinámicos basados en un plan: La factibilidad se determina en tiempo de ejecución (dinámicamente) en vez de antes de comenzar la ejecución(estáticamente). Una tarea es aceptada como ejecutable si es posible satisfacer sus restricciones de tiempo. Al llegar una tarea se intenta crear un plan que contenga las tareas anteriores y la nueva. Si la nueva puede ser planificada sin afectar a las demás, esta es aceptada.
- Dinámicos de mejor esfuerzo: No se hace análisis de factibilidad, se intentan cumplir todos los plazos y si el plazo falla se aborta la ejecución de cualquier proceso. Es fácil de implementar, pero no sabremos si un tiempo será suficiente para cumplir una tarea hasta que pase ese tiempo o se complete la tarea.

El problema de la inversión de prioridad

Esta sucede cuando las circunstancias dentro del sistema fuerzan a una tarea de mayor prioridad a esperar por culpa de una de menor prioridad.

Un ejemplo sucede si la de menor prioridad ha bloqueado un recurso y la de mayor prioridad lo intenta bloquear también. Lo que ocurrirá es que la de mayor prioridad pasará a bloqueado hasta que el recurso esté disponible.

La **inversión de prioridad ilimitada** es un caso más serio, en el cual la duración de la inversioón de prioridad depende del tiempo necesario para conseugir el recurso compartido y de las acciones de otras tareas no relacionadas.

En los sistemas prácticos se usan dos enfoques para evitar la inversión de prioridad:

- Herencia de prioridad: en la que una tarea de menor prioridad hereda la prioridad de una tarea de mayor prioridad pendiente de un recurso que compartan. Esto sucede tan pronto como la tarea de mayor prioridad se bloquea en el recurso y finaliza cuando la tarea de menor prioridad libera el recurso.
- Techo de prioridad: Se asocia una prioridad con cada recurso y esta es un nivel más alta que la prioridad de usuario más prioritario. Entonces el planificador asigna esta prioridad a cualquier tarea que acceda al recurso. Al terminar la tarea de usar el recurso, se le devuelve su prioridad normal.