Main Memory

Introducción

Debido a que el CPU es compartido por diferentes procesos, el rendimiento y velocidad del CPU se puede mejorar por medio de memoria compartida por parte de los diferentes procesos. Los algoritmos que son utilizados van desde un enfoque primitivo de máquina hasta estrategias de paginación y segmentación.

Background

La memoria principal es esencial para la operación del CPU. El lo utiliza para escribir y leer direcciones y datos necesarios para la ejecución de procesos. La memoria no conoce como fueron generadas estas direcciones (por el program counter, indexación, direccionamiento indirecto, direccion literal, etc.) o para que son (instrucciones o data). El CPU dispone de registros para propósitos generales que trabajan con direcciones de memoria principal, nunca con otras como direcciones disco duro. Algunas veces tarda en completar todas estas instrucciones por lo cual le es necesario tener un espacio de memoria mas rápido llamado cache. También se cuenta con el problema que cada programa no debe acceder a otras regiones en la memoria si no se le es permitido. Los procesos cuentan con dos registros donde almacenan un inicio de la dirección de memoria y un limite que le es permitido usar. *f8.2
En modo usuario no es permitido ejecutar por su propia cuenta operaciones con las direcciones de memoria, únicamente en modo Kernel le es permitido debido los privilegios que tienen sus instrucciones, y este actúa como un intermediario entre el modo usuario y el sistema operativo para que se realicen operaciones con las memorias. Sin embargo, las direcciones como las manejamos en modo usuario no son realmente las direcciones que utiliza la memoria principal, en modo usuario se les conoce como direcciones simbólicas. Estas direcciones luego se convierten en direcciones físicas de memoria por medio de un proceso de Binding, este proceso se lleva a cabo por medio de tres formas: Compile Time, uno especifica la dirección a utilizar por el compilador para ejecutar las tareas, el compilador genera código absoluto. Load Time, no se conoce la dirección donde el proceso residirá por lo tanto el compilador genera código realojadle. Execution Time: si el proceso puede se movido de segmento de memoria durante su ejecución es mejor esperar hasta este tiempo de ejecución para conocer una dirección de memoria utilizable. El encargado de mapear las direcciones de memorias virtuales (lógicas) a direcciones físicas es llamado Memory-Management Unit (MMU). El se encarga de convertir memorias virtuales a memorias físicas y viceversa.

Swapping

La memoria principal nos limita con la capacidad de almacenamiento, es por ello que muchas veces un programa puede ser movido de memoria principal a disco duro y así permitir que otros programas sean ejecutados. El sistema mantiene una cola de programas listos a ejecutar administrado por un agente llamado Dispatcher. Si el programa siguiente a ejecutar no se encuentra en memoria el dispatcher remplaza algún programa en memoria principal con el siguiente programa a ejecutar. Cuando ya se dispone de mas memoria disponible este proceso se detiene. Muchas veces el swapping puede realizarse con secciones de código de algún

proceso, en lugar de todo. Esto trae consigo otros problemas, como por ejemplo un proceso puede estar esperando un I/O por lo cual no se podrá hacer swapping. Algunos sistemas operativos, como los móviles, evaden el swapping y utilizan otras técnicas como pedir voluntariamente a procesos que liberen memoria.

Contiguous Memory Allocation

La memoria principal debe almacenar el sistema operativo que se esta utilizando y los programas de modo usuarios que se ejecutan en el momento, es por ello que la memoria debe de alocar de manera eficiente todos los segmentos para no tener un menor rendimiento. Estos espacios de memoria se pueden proteger de no interrumpir uno de otro con la técnica de los registros que delimitan el rango utilizado. Antiguamente se utilizaban técnicas para la locación de memoria la segmentación de ella para cada proceso, lo que limitaba el multiprocessing que el CPU podía tener. U otra técnica como asignar una parte para el OS y todo el resto de memoria es para procesos de usuario y cuando se terminan estos procesos, el espacio de memoria se libera para algún otro proceso. Conforme el paso del tiempo se formarán hoyos de espacios de memoria libre los cuales el sistema operativo debe utilizar para almacenar data que sea lo suficientemente menor a ese espacio disponible e usa técnicas como: First Fit, Best Fit, Worst Fit. Estos métodos de locación sufren de fragmentación externa, es decir, el espacio de memoria se divide en pequeños fragmentos. La fragmentación externa ocurre cuando hay suficiente espacio para almacenar un request de memoria, pero no esta de manera contigua para hacerlo. La otra forma de fragmentación es la interna, la cual ocurre cuando hay memoria sobrante dentro de una partición previamente realizada. Una solución es la compactación, consiste en alocar todos estos espacios de memoria libres de manera contigua para que al final se tenga un espacio grande de memoria libre. Sin embargo, a veces no es posible debido a que la segmentación es estática o se realiza a nivel de assembler, este método solo es posible si la re-locación es dinámica.

Segmentation:

Segmentación provee un mecanismo que mapea las direcciones físicas de la computadora a una vista de usuario para que se puede interactuar con ella. Una memoria lógica es una colección de segmentos, cada segmento tiene su nombre y tamaño. El programador visualiza el manejo de memoria como arrays, listas, stacks, etc. Se piensa que la memoria esta constituida de una manera de 2 dimensiones, pero en realidad solo es una secuencia de bytes en una dimensión. Y se debe efectuar un mapeo para esta conversión de dimensiones. Esto lo efectúa una Segment Table donde se almacena el inicio y final de la dirección de memoria. Esto es esencial cuando se habla, como por ejemplo de, arrays ya que tienen un comienzo y un limite que debe de ser respetado.

Paging

Segmentacion permite que un espacio de memoria físico no sea contiguo. Paging es otro método que provee esta misma ventaja. Pero paging evita la fragmentación externa y promueve la compactación. Esto consiste en dividir la memoria física en segmentos de tamaño predeterminado llamado Frames y dividir igualmente la memoria lógica con un tamaño igual

llamado Pages. Cuando un proceso se ejecutará se carga la page correspondiente al frame correspondiente. Cada dirección generada por el CPU consta de dos partes: page number, page offset. El page number se utiliza como índice en una Page Table. Esta contiene la base de memoria física para cada pagina, esto se combina con el offset para definir la memoria que se enviara a la unidad de memoria. Cada sistema operativo implementa todo esto a su propia manera, algunas alojan pages por proceso. La implementación a nivel de hardware puede realizarse con registros dedicados para esta tarea, page-table base register (PTBR). Cambiar las tablas de páginas requiere cambiar solo este registro, lo que reduce sustancialmente el tiempo de cambio de contexto.