Capítulo 4 Memoria virtual

Cómo la memoria virtual se mapea a la memoria física.

La memoria virtual es una técnica de gestión de la memoria que permite que el sistema operativo disponga, tanto para el software de usuario como para sí mismo, de mayor cantidad de memoria que esté disponible físicamente. La mayoría de los ordenadores tienen cuatro tipos de memoria: registros en la CPU, la memoria caché (tanto dentro como fuera del CPU), la memoria RAM y el disco duro. En ese orden, van de menor capacidad y mayor velocidad a mayor capacidad y menor velocidad.

Muchas aplicaciones requieren acceso a más información (código y datos) que la que se puede mantener en memoria física. Esto es así sobre todo cuando el sistema operativo permite múltiples procesos y aplicaciones ejecutándose simultáneamente. Una solución al problema de necesitar mayor cantidad de memoria de la que se posee consiste en que las aplicaciones mantengan parte de su información en disco, moviéndola a la memoria principal cuando sea necesario. Hay varias formas de hacer esto.

Una opción es que la aplicación misma sea responsable de decidir qué información será guardada en cada sitio (segmentación), y de traerla y llevarla. La desventaja de esto, además de la dificultad en el diseño e implementación del programa, es que es muy probable que los intereses sobre la memoria de dos o varios programas generen conflictos entre sí: cada programador podría realizar su diseño teniendo en cuenta que es el único programa ejecutándose en el sistema. La alternativa es usar memoria virtual, donde la combinación entre hardware especial y el sistema operativo hace uso de la memoria principal y la secundaria para hacer parecer que el ordenador tiene mucha más memoria principal (RAM) que la que realmente posee. Este método es invisible a los procesos. La cantidad de memoria máxima que se puede hacer ver que hay tiene que ver con las características del procesador. Por ejemplo, en un sistema de 32 bits, el máximo es 232, lo que da 4096 Megabytes (4 Gigabytes). Todo esto hace el trabajo del programador de aplicaciones mucho más fácil, al poder ignorar completamente la necesidad de mover datos entre los distintos espacios de memoria.

Aunque la memoria virtual podría estar implementada por el software del sistema operativo, en la práctica casi siempre se usa una combinación de hardware y software, dado el esfuerzo extra que implicaría para el procesador.

Operación básica

Cuando se usa memoria virtual, o cuando una dirección es leída o escrita por la CPU, una parte del hardware dentro de la computadora traduce las direcciones de memoria generadas por el software (direcciones virtuales) en:

La dirección real de memoria (la dirección de memoria física).

Una indicación de que la dirección de memoria deseada no se encuentra en memoria principal (llamado excepción de memoria virtual)

En el primer caso, la referencia a la memoria es completada, como si la memoria virtual no hubiera estado involucrada: el software accede donde debía y sigue ejecutando

normalmente. En el segundo caso, el sistema operativo es invocado para manejar la situación y permitir que el programa siga ejecutando o aborte según sea el caso. La memoria irreal es una técnica para proporcionar la simulación de un espacio de memoria mucho mayor que la memoria física de una máquina. Esta "ilusión" permite que los programas se ejecuten sin tener en cuenta el tamaño exacto de la memoria física.

La ilusión de la memoria virtual está soportada por el mecanismo de traducción de memoria, junto con una gran cantidad de almacenamiento rápido en disco duro. Así en cualquier momento el espacio de direcciones virtual hace un seguimiento de tal forma que una pequeña parte de él, está en memoria física y el resto almacenado en el disco, y puede ser referenciado fácilmente.

Debido a que sólo la parte de memoria virtual que está almacenada en la memoria principal es accesible a la CPU, según un programa va ejecutándose, la proximidad de referencias a memoria cambia, necesitando que algunas partes de la memoria virtual se traigan a la memoria principal desde el disco, mientras que otras ya ejecutadas, se pueden volver a depositar en el disco (archivos de paginación).

La memoria virtual ha llegado a ser un componente esencial de la mayoría de los sistemas operativos actuales. Y como en un instante dado, en la memoria sólo se tienen unos pocos fragmentos de un proceso dado, se pueden mantener más procesos en la memoria. Es más, se ahorra tiempo, porque los fragmentos que no se usan no se cargan ni se descargan de la memoria. Sin embargo, el sistema operativo debe saber cómo gestionar este esquema.

La memoria virtual también simplifica la carga del programa para su ejecución, llamada reubicación, este procedimiento permite que el mismo programa se ejecute en cualquier posición de la memoria física.

En un estado estable, prácticamente toda la memoria principal estará ocupada con fragmentos de procesos, por lo que el procesador y el S.O tendrán acceso directo a la mayor cantidad de procesos posibles, y cuando el S.O traiga a la memoria un fragmento, deberá expulsar otro. Si expulsa un fragmento justo antes de ser usado, tendrá que traer de nuevo el fragmento de manera casi inmediata. Demasiados intercambios de fragmentos conducen a lo que se conoce como hiperpaginación: donde el procesador consume más tiempo intercambiando fragmentos que ejecutando instrucciones de usuario. Para evitarlo el sistema operativo intenta adivinar, en función de la historia reciente, qué fragmentos se usarán con m

Gestión de memoria

El sistema de memoria virtual de los actuales computadores surgió para liberar al programador de una serie de tareas relacionadas con el uso que los programas debían realizar con la memoria. La memoria virtual automatiza la gestión entre los dos niveles principales de la jerarquía de memoria: memoria principal y disco. Antes de entrar en los mecanismos específicos de la memoria virtual revisaremos una serie de funciones que deben incorporarse en la gestión de memoria.

Espacio de Direcciones Lógicas vs. Físicas

• El concepto de espacio de direcciones lógicas que está ligado a un espacio de direcciones físicas separadas es central para la apropiada administración de la memoria.

– Dirección Lógica – generada por el CPU; también conocida como dirección virtual.

– Dirección Física – direcciones vistas por la unidad de memoria.

Las direcciones Lógicas y Físicas son las mismas en el tiempo de compilación y en el tiempo de carga.

En los esquemas de vinculación (liga) de direcciones; la dirección Lógica (Virtual) y la dirección Física difieren en el tiempo de ejecución .

1.1. Solapamiento (overlay)

El tamaño de la memoria principal disponible en los computadores ha aumentado de forma sostenida desde sus orígenes. Sin embargo, el tamaño de los programas ha crecido más rápidamente, por lo que la necesidad de ejecutar programas que no cabían en la memoria principal ha sido una constante en la historia de los computadores. Una forma de superar esta limitación es el uso de la técnica de solapamiento (overlay). Esta técnica divide en módulos el programa cuyo tamaño sobrepasa la capacidad de la memoria principal, y que reside por tanto en memoria secundaria (disco). Después se introducen en los lugares adecuados de cada módulo, y al margen de la lógica propia del programa, las instrucciones de E/S necesarias para cargar en memoria principal aquellos módulos cuyas instrucciones deban ejecutarse o cuyos datos vayan a ser referencia

Paginación por demanda

La paginación por demanda es un sistema de paginación con el cual, además de las ventajas de la paginación convencional, se busca disminuir los tiempos de respuesta y aumentar la cantidad de programas en memoria. Para lograr estos objetivos se hace uso de un intercambiador perezoso (llamado paginador) el cual carga a memoria solo las páginas que serán utilizadas por el programa en ejecución, de esta manera se logra un menor tiempo de carga y un ahorro en cuanto a espacio utilizado por dicho programa, ya que, por un lado, no necesitamos que todo el programa este en memoria para comenzar su ejecución mientras que, por otra parte, al no estar el programa completo en memoria, disminuimos considerablemente el espacio que éste ocupa.

Ya que el paginador solo busca las páginas que se necesitan para ejecutar algún programa, debemos agregar un bit que nos diga si las referencias de memoria son válidas o no, de lo contrario, al no encontrar una página no podríamos diferenciar si el paginador aún no la carga o si esta es realmente una referencia inválida.

4.2 Demanda de página

El proceso que se sigue es el siguiente:

- Se intenta leer la página requerida
- Si la página requerida ya esta en memoria, simplemente se lee.
- Si no está en memoria, revisa si la referencia es válida.
- Si la referencia es inválida, se aborta.
- Si la referencia es válida, se intenta cargar la página.
- Cuando la página sea cargada, se reintenta la instrucción.

Al buscar una página, si esta no está en memoria, necesitará ser cargada. A este proceso se le llama fallo de página.

Al iniciar la ejecución de un programa, la tabla de páginas cuenta con todas sus entradas inválidas por lo cual el paginador fallará hasta tener lo necesario para iniciar el programa. Luego de esta carga inicial se comprobará si la siguiente página a utilizar ya está en memoria, en caso de que la página se encuentre, ésta es leída, pero cuando la página no es encontrada (y es una referencia válida) tenemos dos posibilidades:

- Si existe un frame libre, se carga y se lee.
- Si no tenemos frames libres, se intercambia la página de algún frame por la información a utilizar.

El criterio utilizado para seleccionar qué página será intercambiada varía dependiendo de la implementación del sistema. Muchos de los problemas que presenta el sistema de paginación por demanda son debido a los fallos de página y principalmente a saber cuál es la página más conveniente para intercambiar. Esto se debe a que no podemos saber cuáles páginas serán utilizadas prontamente y cuales no se volverán a utilizar, existen variados algoritmos que buscan aminorar este problema, los cuales, serán analizados más adelante.

Ventajas

A continuación se verán algunas de las ventajas de utilizar paginación por demanda:

- Al no cargar las páginas que no son utilizadas ahorra memoria para otras aplicaciones.
- Al mejorar el uso de la memoria, mejora el grado de multiprogramación.
- Carga inicial más rápida ya que solo lee del disco lo que se utilizará.
- Capacidad de hacer funcionar programas que ocupan más memoria que la poseída.
- COW (Copia en escritura): Permite utilizar las mismas páginas para dos procesos (padre-hijo) hasta que uno de estos las modifique.

Desventajas

La paginación por demanda puede llevarnos a las siguientes situaciones:

Debido a la sobre-asignación podemos quedarnos sin frames libres para agregar nuevas páginas, si esto sucede debemos recurrir a un reemplazo.

Cada fallo de página requiere cargar a memoria una página a leer, si ocurren muchos fallos de página el rendimiento empeora notablemente.

Las páginas que son sacadas de los frames por intercambio pueden volver a ser llamadas, lo que ocasiona que se lea en múltiples ocasiones la misma información.

Como vemos, los grandes problemas del sistema de paginación por demanda son a causa de los fallos de página y como estos son tratados. A continuación profundizaremos sobre estos.