Trabajo Practico 6

Tema N8: Implementación de Memoria Virtual en Hardware

Materia: Arquitectura y Sistemas Operativos

Profesora: Teresa Benítez.

GRUPO 10: Agüero Rosa,

Aliende Héctor,

Bertozzi Fernando,

Sierra Enzo,

Stevens Danilo.

Comisión: Comisión 1.2

Año:2024

ELEMENTOS A TENER EN CUENTA PARA EL DISEÑO DE HARDWARE PARA SOPORTAR LA PAGINACION Y LA TABLAS DE PAGINAS

En las siguientes secciones analizaremos otras cuestiones que deben considerar los diseñadores de sistemas operativos para poder obtener un buen rendimiento de un sistema de paginación.

Políticas de asignación local contra las de asignación global

Una cuestión importante es cómo se debe asignar la memoria entre los procesos ejecutables en competencia.
En general, los algoritmos globales funcionan mejor, en especial cuando el tamaño del conjunto de trabajo puede variar durante el tiempo de vida de un proceso. Si se utiliza un algoritmo local y el conjunto de trabajo crece, se producirá una sobre paginación, aun cuando haya muchos marcos de página libres. Si el conjunto de trabajo se reduce, los algoritmos locales desperdician memoria.
Si se utiliza un algoritmo global, el sistema debe decidir en forma continua cuántos marcos de página asignar a cada proceso. Una manera es supervisar el tamaño del conjunto de trabajo, según lo

indicado por los bits de envejecimiento, pero este método no necesariamente evita la sobrepaginación. El conjunto de trabajo puede cambiar de tamaño en microsegundos, mientras que los bits de envejecimiento son una medida burda, esparcida a través de varios pulsos de reloj.
Otro método es tener un algoritmo para asignar marcos de página a los procesos. Una manera es determinar periódicamente el número de procesos en ejecución y asignar a cada proceso una parte igual. Así, con 12,416 marcos de página disponibles (es decir, que no son del sistema operativo) y 10 procesos, cada proceso obtiene 1241 marcos. Los seis restantes pasan a una reserva, para utilizarlos

Control de carga

Aun con el mejor algoritmo de reemplazo de páginas y una asignación global óptima de marcos de página a los procesos, puede ocurrir que el sistema se sobrepagine. De hecho, cada vez que los con

juntos de trabajo combinados de todos los procesos exceden a la capacidad de la memoria, se puede esperar la sobrepaginación. Un síntoma de esta situación es que el algoritmo PFF indica que algunos procesos necesitan más memoria, pero ningún proceso necesita menos memoria. En este caso no hay forma de proporcionar más memoria a esos procesos que la necesitan sin lastimar a algún otro proceso. La única solución real es deshacerse temporalmente de algunos procesos.
Una buena forma de reducir el número de procesos que compiten por la memoria es intercambiar algunos de ellos enviándolos al disco y liberar todas las páginas que ellos mantienen. Por ejemplo, un proceso puede intercambiarse al disco y sus marcos de página dividirse entre otros procesos que están sobrepaginando. Si el sobrepaginado se detiene, el sistema puede operar de esta forma por un tiempo. Si no se detiene hay que intercambiar otro proceso y así en lo sucesivo hasta que se detenga el sobrepaginado. Por ende, incluso hasta con la paginación se sigue necesitando el intercambio, sólo que ahora se utiliza para reducir la demanda potencial de memoria, en vez de reclamar páginas.

Tamaño de página

El tamaño de página es un parámetro que a menudo el sistema operativo puede elegir. Incluso si el hardware se ha diseñado, por ejemplo, con páginas de 512 bytes, el sistema operativo puede considerar fácilmente los pares de páginas 0 y 1, 2 y 3, 4 y 5, y así en lo sucesivo, como páginas de 1 KB al asignar siempre dos marcos de página de 512 bytes consecutivos para ellas.
Para determinar el mejor tamaño de página se requiere balancear varios factores competitivos. Como resultado, no hay un tamaño óptimo en general. Para empezar, hay dos factores que

están a favor de un tamaño de página pequeño. Un segmento de texto, datos o pila elegido al azar no llenará un número integral de páginas. En promedio, la mitad de la página final estará vacía. El espacio adicional en esa página se desperdicia. A este desperdicio se le conoce como **fragmentación interna**. Con *n* segmentos en memoria y un tamaño de página de *p* bytes, se desperdiciarán *np*/2 bytes en fragmentación interna. Este razonamiento está a favor de un tamaño de página pequeño.
Otro argumento para un tamaño de página pequeño se hace aparente si consideramos que un programa consiste de ocho fases secuenciales de 4 KB cada una. Con un tamaño de página de 32 KB, se deben asignar 32 KB al programa todo el tiempo. Con un tamaño de página de 16 KB,

Espacios separados de instrucciones y de datos

La mayor parte de las computadoras tienen un solo espacio de direcciones que contiene tantos programas
como datos, como se muestra en la figura 3-25(a). Si este espacio de direcciones es lo bastante grande, todo funciona bien. No obstante, a menudo es demasiado pequeño, lo cual obliga a los programadores a pararse de cabeza tratando de ajustar todo en el espacio de direcciones.

Páginas compartidas

Otra cuestión de diseño es la compartición. En un sistema de multiprogramación grande, es común que varios usuarios ejecuten el mismo programa a la vez. Evidentemente es más eficiente compartir las páginas para evitar tener dos copias de la misma página en memoria al mismo tiempo. Un problema es que no todas las páginas se pueden compartir. En especial, sólo pueden compartirse las páginas que son de sólo lectura como el texto del programa, pero las páginas de datos no.

Política de limpieza

La paginación funciona mejor cuando hay muchos marcos de página libres que se pueden reclamar al momento en que ocurran fallos de página. Si cada marco de página está lleno y además modificado, antes de que se pueda traer una nueva página se debe escribir una página anterior en el disco.
Para asegurar una provisión abundante de marcos de página libres, muchos sistemas de paginación tienen un proceso en segundo plano conocido como **demonio de paginación**, que está inactivo la mayor parte del tiempo pero se despierta en forma periódica para inspeccionar el estado de la memoria.
Si hay muy pocos marcos de página libres, el demonio de paginación empieza a seleccionar páginas para desalojarlas mediante cierto algoritmo de reemplazo de páginas. Si estas páginas han sido modificadas después de haberse cargado, se escriben en el disco.

interfaz de memoria virtual

Hasta ahora, en todo nuestro análisis hemos supuesto que la memoria virtual es transparente para los procesos y los programadores; es decir, todo lo que ven es un gran espacio de direcciones virtuales en una computadora con una memoria física (más) pequeña. Con muchos sistemas esto es cierto pero, en algunos sistemas avanzados, los programadores tienen cierto control sobre el mapa de memoria y pueden utilizarlo de maneras no tradicionales para mejorar el comportamiento de un programa. En esta sección analizaremos unas cuantas de estas formas.
Una razón por la que se otorga a los programadores el control sobre su mapa de memoria es para permitir que dos o más procesos compartan la misma memoria. Si los programadores pueden nombrar regiones de su memoria, tal vez sea posible para un proceso dar a otro proceso el nombre de una región de memoria, de manera que el proceso también pueda asociarla. Con dos (o más) procesos

CUESTIONES TÉCNICAS DE IMPLEMENTACIÓN

Los implementadores de los sistemas de memoria virtual tienen que elegir entre los principales algoritmos
teóricos: entre el algoritmo de segunda oportunidad y el de envejecimiento, entre la asignación de páginas local o global, y entre la paginación bajo demanda o la prepaginación. Pero también tienen que estar al tanto de varias cuestiones prácticas de implementación. En esta sección daremos un vistazo a unos cuantos de los problemas comunes y ciertas soluciones.

Participación del sistema operativo en la paginación

Hay cuatro ocasiones en las que el sistema operativo tiene que realizar trabajo relacionado con la paginación: al crear un proceso, al ejecutar un proceso, al ocurrir un fallo de página y al terminar un proceso. Ahora examinaremos brevemente cada una de estas ocasiones para ver qué se tiene que hacer.
Cuando se crea un proceso en un sistema de paginación, el sistema operativo tiene que determinar qué tan grandes serán el programa y los datos (al principio), y crear una tabla de páginas para ellos. Se debe asignar espacio en memoria para la tabla de páginas y se tiene que inicializar. La tabla de páginas no necesita estar residente cuando el proceso se intercambia hacia fuera, pero tiene que estar en memoria cuando el proceso se está ejecutando. Además, se debe asignar espacio en el área de intercambio en el disco, para que cuando se intercambie una página, tenga un lugar a donde ir. El área de intercambio también se tiene que inicializar con el texto del programa y los datos, para que cuando el nuevo proceso empiece a recibir fallos de página, las páginas se puedan traer. Algunos sistemas paginan el texto del programa directamente del archivo ejecutable, con lo cual se ahorra espacio en disco y tiempo de inicialización. Por último, la información acerca de la tabla de páginas y el área de intercambio en el disco se debe registrar en la tabla de procesos.
Cuando un proceso se planifica para ejecución, la MMU se tiene que restablecer para el nuevo proceso y el TLB se vacía para deshacerse de los restos del proceso que se estaba ejecutando antes. La tabla de páginas del nuevo proceso se tiene que actualizar, por lo general copiándola o mediante un apuntador a éste hacia cierto(s) registro(s) de hardware.

Manejo de fallos de página

Finalmente, estamos en una posición para describir con detalle lo que ocurre en un fallo de página. La secuencia de eventos es la siguiente:

1. El hardware hace un trap al kernel, guardando el contador de programa en la pila. En la mayor parte de las máquinas, se guarda cierta información acerca del estado de la instrucción actual en registros especiales de la CPU.
2. Se inicia una rutina en código ensamblador para guardar los registros generales y demás información volátil, para evitar que el sistema operativo la destruya. Esta rutina llama al sistema operativo como un procedimiento.
3. El sistema operativo descubre que ha ocurrido un fallo de página y trata de descubrir cuál página virtual se necesita. A menudo, uno de los registros de hardware contiene esta información. De no ser así, el sistema operativo debe obtener el contador de programa, obtener la instrucción y analizarla en software para averiguar lo que estaba haciendo cuando ocurrió el fallo.
4. Una vez que se conoce la dirección virtual que produjo el fallo, el sistema comprueba si esta dirección es válida y si la protección es consistente con el acceso. De no ser así, el proceso recibe una señal o es eliminado. Si la dirección es válida y no ha ocurrido un fallo de página, el sistema comprueba si hay un marco de página disponible. Si no hay marcos disponibles, se ejecuta el algoritmo de reemplazo de páginas para seleccionar una víctima.
5. Si el marco de página seleccionado está sucio, la página se planifica para transferirla al disco y se realiza una conmutación de contexto, suspendiendo el proceso fallido y dejando que se ejecute otro hasta que se haya completado la transferencia al disco. En cualquier caso, el marco se marca como ocupado para evitar que se utilice para otro propósito.

Respaldo de instrucción

Cuando un programa hace referencia a una página que no está en memoria, la instrucción que produjo el fallo se detiene parcialmente y ocurre un trap al sistema operativo. Una vez que el sistema operativo obtiene la página necesaria, debe reiniciar la instrucción que produjo el trap. Es más fácil decir esto que hacerlo.

Para ver la naturaleza del problema en el peor de los casos, considere una CPU que tiene instrucciones con dos direcciones, como el procesador Motorola 680x0, utilizado ampliamente en sistemas integrados. Por ejemplo, la instrucción

MOV.L #6(A1),2(A0)

es de 6 bytes (vea la figura 3-28). Para poder reiniciar la instrucción, el sistema operativo debe determinar en dónde se encuentra el primer byte de la instrucción. El valor del contador de programa al momento en que ocurre el trap depende de cuál fue el operando que falló y cómo se ha implementado el microcódigo de la CPU.

Bloqueo de páginas en memoria

Aunque no hemos hablado mucho sobre la E/S en este capítulo, el hecho de que una computadora tenga memoria virtual no significa que estén ausentes las operaciones de E/S. La memoria virtual y la E/S interactúan en formas sutiles. Considere un proceso que acaba de emitir una llamada al sistema para leer algún archivo o dispositivo y colocarlo en un búfer dentro de su espacio de direcciones. Mientras espera a que se complete la E/S, el proceso se suspende y se permite a otro proceso ejecutarse. Este otro proceso recibe un fallo de página.

Almacén de respaldo

En nuestro análisis de los algoritmos de reemplazo de páginas, vimos cómo se selecciona una página para eliminarla. No hemos dicho mucho con respecto a dónde se coloca en el disco cuando se pagina hacia fuera de la memoria. Ahora vamos a describir algunas cuestiones relacionadas con la administración del disco.

El algoritmo más simple para asignar espacio de página en el disco es tener una partición de intercambio especial en el disco o aún mejor es tenerla en un disco separado del sistema operativo (para balancear la carga de E/S).