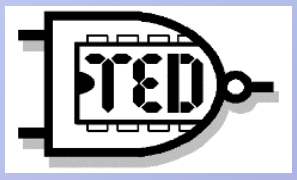
FACULTAD DE INGENIERIA
UNIVERSIDAD NACIONAL DE JUJUY

U.N.Ju. – F.I.

INGENIERÍA INFORMÁTICA

LICENCIATURA EN SISTEMAS

TÉCNICAS Y ESTRUCTURAS DIGITALES



MEMORIA VIRTUAL

Concepto

Aún cuando la tecnología detrás de la construcción de las implementaciones modernas de almacenamiento es realmente impresionante, el administrador de sistemas promedio no necesita estar al tanto de los detalles. De hecho, solamente existe un factor que los administradores de sistemas deberían tener en consideración:

Nunca hay suficiente RAM

Mientras que esta frase puede sonar al principio un poco cómica, muchos diseñadores de sistemas operativos han empleado una gran cantidad de tiempo tratando de reducir el impacto de esta limitación. Esto lo han logrado mediante la implementación de la *memoria virtual*: una forma de combinar RAM con un almacenamiento más lento para darle al sistema la apariencia de tener más RAM de la que tiene instalada realmente.

MEMORIA VIRTUAL

Vamos a comenzar con una aplicación hipotética. El código de máquina que conforma esta aplicación tiene un tamaño de 10000 bytes. También requiere otros 5000 bytes para el almacenamiento de datos y para la memoria intermedia de E/S. Esto significa que para ejecutar la aplicación, debe haber más de 15000 bytes de RAM disponible; un byte menos y la aplicación no será capaz de ejecutarse.

Este requerimiento de 15000 bytes se conoce como el *espacio de direcciones* de la aplicación. Es el número de direcciones únicas necesarias para almacenar la aplicación y sus datos. En las primeras computadoras, la cantidad de RAM disponible tenía que ser mayor que el espacio de direcciones de la aplicación más grande a ejecutar; de lo contrario, la aplicación fallaría con un error de "memoria insuficiente".

Un enfoque posterior conocido como *solapamiento* intentó aliviar el problema permitiendo a los programadores dictar cuales partes de sus aplicaciones necesitaban estar residentes en memoria en cualquier momento dado. De esta forma, el código requerido solamente para propósitos de inicialización podía ser sobreescrito con código a utilizar posteriormente.

MEMORIA VIRTUAL

Mientras que el solapamiento facilitó las limitaciones de memoria, era un proceso muy complejo y susceptible a errores. El solapamiento también fallaba en solucionar el problema de las limitaciones de memoria globales al sistema en tiempo de ejecución. En otras palabras, un programa con solapamiento requería menos memoria para ejecutarse que un programa sin solapamiento, pero si el sistema no tiene suficiente memoria para el programa solapado, el resultado final es el mismo: un error de falla de memoria.

Con la memoria virtual el concepto del espacio de direcciones de las aplicaciones toma un significado diferente. En vez de concentrarse en *cuánta* memoria necesita una aplicación para ejecutarse, un sistema operativo con memoria virtual continuamente trata de encontrar una respuesta a la pregunta:

"¿qué tan *poca* memoria necesita la aplicación para ejecutarse?".

Aunque inicialmente pareciera que nuestra aplicación hipotética requiere de un total de 15000 bytes para ejecutarse, recuerde que el acceso a memoria tiende a ser secuencial y localizado.

MEMORIA VIRTUAL

Debido a esto, la cantidad de memoria requerida para ejecutar la aplicación en un momento dado es menos que 15000 bytes; usualmente mucho menos. Considere los tipos de accesos de memoria requeridos para ejecutar una instrucción de máquina sencilla:

- La instrucción es leída desde la memoria.
- Se leen desde memoria los datos requeridos por la instrucción.
- Después de completar la instrucción, los resultados de la instrucción son escritos nuevamente en memoria.

El número real de bytes necesarios para cada acceso de memoria varían de acuerdo a la arquitectura de la CPU, la instrucción misma y el tipo de dato. Sin embargo, aún si una instrucción requiere de 100 bytes de memoria por cada tipo de acceso de memoria, los 300 bytes requeridos son mucho menos que el espacio de direcciones total de la aplicación de 15000 bytes. Si hubiese una forma de hacer un seguimiento de los requerimientos de memoria de la aplicación a medida que esta se ejecuta, sería posible mantener la aplicación ejecutándose usando menos memoria que lo que indicaría su espacio de direcciones.

MEMORIA VIRTUAL

Pero esto genera una pregunta:

Si solamente una parte de la aplicación está en memoria en un momento dado, ¿dónde está el resto?.

Almacenamiento de respaldo

La respuesta corta a esta pregunta es que el resto de la aplicación se mantiene en disco. En otras palabras, el disco actúa como un *almacenamiento de respaldo* para la RAM; un medio más lento y también más grande que actúa como un "respaldo" para un almacenamiento más rápido y más pequeño. Esto puede parecer al principio como un gran problema de rendimiento en su creación: después de todo, las unidades de disco son mucho más lentas que la RAM.

Aunque esto es cierto, es posible tomar ventaja del comportamiento de acceso secuencial y localizado de las aplicaciones y eliminar la mayoría de las implicaciones de rendimiento en el uso de unidades de disco como unidades de respaldo para la RAM. Esto se logra estructurando el subsistema de memoria virtual para que este trate de asegurarse de que esas partes de la aplicación que actualmente se necesitan — o que probablemente se necesitarán en un futuro cercano — se mantengan en RAM solamente por el tiempo en que son realmente requeridas.

ESPACIO de DIRECCIONES VIRTUALES

El espacio de direcciones virtuales es el espacio de direcciones máximo disponible para una aplicación. El espacio de direcciones virtuales varía de acuerdo a la arquitectura del sistema y del sistema operativo. El espacio de direcciones virtuales depende de la arquitectura puesto que es la arquitectura la que define cuántos bits están disponibles para propósitos de direccionamiento. El espacio de direcciones virtuales también depende del sistema operativo puesto que la forma en que el sistema operativo fue implementado puede introducir límites adicionales sobre aquellos impuestos por la arquitectura.

La palabra "virtual" en el espacio de direcciones virtuales, significa que este es el número total de ubicaciones de memoria direccionables disponibles para una aplicación, pero *no* la cantidad de memoria física instalada en el sistema, o dedicada a la aplicación en un momento dado.

En el caso de la aplicación de ejemplo, su espacio de direcciones virtuales es de 15000 bytes.

Para implementar la memoria virtual, para el sistema, es necesario tener un hardware especial de administración de memoria. Este hardware a menudo se conoce como una *MMU* (Memory Management Unit). Sin una MMU, cuando la CPU accede a la RAM, las ubicaciones reales de RAM nunca cambian: la dirección de memoria 123 siempre será la misma dirección física dentro de la RAM.

ESPACIO de DIRECCIONES VIRTUALES

Sin embargo, con una MMU, las direcciones de memoria pasan a través de un paso de traducción antes de cada acceso de memoria. Esto significa que la dirección de memoria 123 puede ser redirigida a la dirección física 82043 en un momento dado y a la dirección 20468 en otro. Como resultado de esto, la sobrecarga relacionada con el seguimiento de las traducciones de memoria virtual a física sería demasiado. En vez de esto, la MMU divide la RAM en *páginas*: secciones contiguas de memoria de un tamaño fijo que son manejadas por el MMU como unidades sencillas. Mantener un seguimiento de estas páginas y sus direcciones traducidas puede sonar como un paso adicional confuso e innecesario, pero de hecho es crucial para la implementación de la memoria virtual. Por tal razón, considere el punto siguiente:

Tomando la aplicación hipotética, ya mencionada, con un espacio de direcciones virtuales de 15000 bytes, asuma que la primera instrucción de la aplicación accede a los datos almacenados en la dirección 12374. Sin embargo, también asuma que la computadora anfitriona solamente tiene 12288 bytes de RAM física. ¿Qué pasa cuando el CPU intenta acceder a la dirección 12374?

Lo que ocurre se conoce como un *fallo de página*.

FALLO DE PÁGINA

Un fallo de página es la secuencia de eventos que ocurren cuando un programa intenta acceder a datos (o código) que está en su espacio de direcciones, pero que no está actualmente ubicado en la RAM del sistema. El sistema operativo debe manejar los fallos de página haciendo residentes en memoria los datos accedidos, permitiendo de esta manera que el programa continúe la operación como que si el fallo de página nunca ocurrió.

En el caso de nuestra aplicación hipotética, la CPU primeramente presenta la dirección deseada (12374) a la MMU. Sin embargo, la MMU no tiene traducción para esta dirección. Por tanto, interrumpe al CPU y causa que se ejecute un software, conocido como el manejador de fallos de página. El manejador de fallos de página determina lo que se debe hacer para resolver esta falla de página.

FALLO DE PÁGINA

El mismo puede:

- Encontrar dónde reside la página deseada en disco y la lee (este es usualmente el caso si el fallo de página es por una página de código).
- Determina que la página deseada ya está en RAM (pero no está asignada al proceso actual) y reconfigura la MMU para que apunte a él.
- Apunta a una página especial que solamente contiene ceros y asigna una nueva página para el proceso solamente si este intenta alguna vez escribir a la página especial (esto se llama una página de *copia en escritura* y es utilizada a menudo por páginas que contienen datos inicializados a cero).

DIRECCIONES DE TRABAJO

El grupo de páginas de memoria física actualmente dedicadas a un proceso específico se conoce como *conjunto de direcciones de trabajo* para ese proceso. El número de páginas en el conjunto de direcciones de trabajo puede crecer o reducirse, dependiendo de la disponibilidad general de páginas del sistema.

El conjunto de direcciones de trabajo crece si un proceso tiene fallos de páginas. El conjunto de direcciones de trabajo se reduce a medida que existen menos y menos páginas libres. Para evitar que se acabe la memoria completamente, se deben eliminar las páginas del conjunto de direcciones de trabajo y convertirlas en páginas libres, disponibles para un uso posterior. El sistema operativo reduce el conjunto de direcciones de trabajo mediante:

- Escribiendo las páginas modificadas a un área dedicada en un dispositivo de almacenamiento masivo (usualmente conocido como espacio de *intercambio* o de *paginado*).
- Marcando las páginas sin modificar como libres (no hay necesidad de escribir estas páginas fuera del disco pues no se han cambiado).

DIRECCIONES DE TRABAJO

Para determinar los conjuntos de trabajo apropiados para todos los procesos, el sistema operativo debe hacer un seguimiento de la información de uso de todas las páginas. De esta manera, el sistema operativo determina cuales páginas son usadas activamente (y deben mantenerse en memoria como residentes) y cuales no (y por lo tanto, se pueden eliminar de memoria). En la mayoría de los casos, se utiliza un tipo de algoritmo de "menos usado recientemente" para determinar cuáles páginas son elegibles para eliminarse de los conjuntos de trabajo de los procesos.

IMPLICACIONES

Mientras que la memoria virtual hace posible que las computadoras manejen más fácilmente aplicaciones más grandes y complejas, como con cualquier otra herramienta, esto viene a un precio. El precio en este caso es el de rendimiento: la memoria virtual de un sistema operativo tiene mucho más que hacer que un sistema operativo sin memoria virtual. Esto significa que el rendimiento nunca es tan bueno con memoria virtual como lo es cuando la misma aplicación esta 100% residente en memoria.

Sin embargo, esta no es razón suficiente para abandonar la idea. Los beneficios de la memoria virtual son demasiados para hacer esto. Y, con un poco de esfuerzo, es posible lograr un buen rendimiento. Lo que se debe hacer es examinar aquellos recursos de sistemas impactados por el uso pesado del subsistema de memoria virtual.

El punto a tener en mente es que el impacto en el rendimiento de la memoria virtual es mínimo cuando se utiliza tan poco como sea posible. Esto significa que el factor determinante para un buen rendimiento del subsistema de memoria virtual es tener suficiente RAM.

Lo siguiente (pero con mucho menos importancia) es suficiente capacidad de E/S de disco y de CPU. Sin embargo, tenga en cuenta que estos recursos solamente ayudan a que el rendimiento del sistema se degrade de una forma más limpia de intensivos fallos de página y del intercambio; pero hacen poco para ayudar el rendimiento del subsistema de memoria virtual (aunque obviamente pueden jugar un papel importante en el rendimiento global del sistema).

ORGANIZACIÓN

Organización y tipos de Memoria Virtual

Los distintos modelos se diferencian por sus políticas de solape y por los métodos que emplean en la organización de la memoria. Los más importantes son:

- Memoria paginada.
- Memoria segmentada.
- Memoria de segmentos paginados.

EJEMPLOS

Sea una computadora de 20 bits con memoria virtual paginada con páginas de 1 KB y un total de memoria física de 256 KB. Se pide, de forma razonada y breve: a)¿Cuál es el formato de la dirección virtual? Indique los campos y el número de bits de los mismos. b)¿Cuál es el número máximo de entradas de la tabla de páginas (de un nivel)? c)¿Cuántos marcos de página tiene la memoria principal? d)¿Cuáles son los campos que se incluyen en una entrada de la tabla de páginas? Indique también para qué se utiliza cada uno de los campos.

Solución:

a) Las páginas ocupan 1 KB = 2^{10} bytes. Como la dirección virtual ocupa 20 bits, se emplean 20 -10 = 10 bits para el número de página. Por tanto, el formato emplea los 10 bits superiores de la dirección para representar el número de página y los 10 bits inferiores para representar el desplazamiento dentro de la página.

b) El número máximo de entradas de la tabla de páginas coincide con el número máximo de páginas, es decir 2^{10}= 1024 entradas.

c) El número de marcos de página viene dado por 256 KB / 1 KB = 256 marcos.

d) En cada entrada de la tabla de página se incluye, entre otros: Bit de presencia, Bit de modificado, Bit de validez, Bits de permisos, Campo en el que se almacena el marco.