Subsistema de memoria

Organización de computadoras 2015

Universidad Nacional de Quilmes

1 Subsistema de memoria

El sistema de computos necesita tener varios tipos de memorias, para ser utilizadas en diferentes situaciones, cumpliendo distintos objetivos. Antes de mostrar porque es esto cierto, veamos cómo caracterizar las memorias según los diferentes aspectos.

En primer lugar, una unidad puede ser interna o externa al sistema según si es fija o desmontable (esto es, portable a otro sistema).

Además puede ser volátil, si su contenido se pierde al perder la alimentación eléctrica, o persistente, si no necesita electricidad para mantener la información.

Puede ser de lectura y escritura, como se requiere en una memoria principal, o bien de solo lectura para almacenar información estática que no necesita modificarse.

Por último, se las puede clasificar segun su método de acceso, en secuencial, directo, aleatorio o asociativo. El **método de acceso secuencial** requiere que la dirección (o identificación) de cada dato esté almacenada junto con él, y por lo tanto el dispositivo debe recorrerse secuencialmente hasta encontrar la identificación buscada. El **método de acceso directo** es el utilizado por los discos magnéticos, donde la direccion del dato se basa en su ubicación física, en particular la busqueda se da en dos etapas: se accede primero a la zona que incluye al dato y luego se busca secuencialmente dentro de esa zona. En el **método aleatorio** cada dato tiene un mecanismo de acceso único y cableado físicamente (cada acceso es independiente de los accesos anteriores). Finalmente, el **método asociativo** organiza cada unidad de información con una etiqueta que la describe en función de su contenido. Entonces, para recuperar un dato se debe analizar un determinado conjunto de bits dentro de la celda, buscando la coincidencia con la clave o patrón buscado. Esta comprobación del contenido de las celdas se lleva a cabo de manera simultánea en todas las celdas.

1.1 Memorias ROM

Para algunas aplicaciones, el programa no necesita ser modificado y entonces puede ser almacenado de manera permanente en una memoria de solo lectura ó ROM (*Read Only Memory*). Ejemplos de memorias ROM pueden encontrarse en videojuegos, calculadoras, hornos de microondas, computadoras en automóviles, etc. Además casi todas las computadoras personales necesitan una memoria ROM donde almacenar el primer programa que da arranque al sistema operativo a partir del acceso (en la mayoria de los casos) a un disco rígido primario.

1.2 Jerarquía de memorias

Como se dijo, la **memoria principal** es volátil y de acceso aleatorio. Si es volátil entonces se necesita otra posibilidad de almacenamiento donde los programas puedan almacenarse de manera persistente y desde donde se recuperen bajo demanda del usuario (cuando dispara la ejecución de un programa).

Esta **memoria secundaria** puede ser resuelta con un disco rígido o un disco de estado sólido (SSD) donde se mantengan persistentes (instalados) los programas. Cuando se necesita de la ejecución de un programa, hecho que puede darse a partir del requerimiento de un usuario o por invocación de otro programa, este debe ser cargado en memoria y permanece allí hasta que la memoria se sobreescribe o se apaga el sistema.

La pregunta que puede surgir es ¿porque no montar la memoria principal en una tecnología persistente, como puede ser un disco de estado sólido? Para responderla es importante destacar que existe una relación de compromiso entre el tiempo de acceso, el costo por bit y la capacidad de almacenamiento (ver figura 1). Un disco tiene mayor capacidad (y por lo tanto menor costo por bit) pero un tiempo de acceso mucho mayor. Esto impacta directamente en el desempeño de la CPU pues el tiempo de ejecución de una instrucción está condicionado por el tiempo de acceso a memoria principal. Además, en este punto es importante notar que algunos programas pueden no ser ejecutados nunca y algunos datos pueden no ser accedidos, aunque se los tenga disponibles en el sistema.

Figure 1: Relacion de compromiso

Es por esto que se incorpora una memoria mas pequeña y rápida (de acceso aleatorio) donde se cargan los programas a la hora de ser ejecutados y los datos cuando se los

necesita. Si lo que se necesita es asegurar una velocidad aceptable y no se necesita gran capacidad, ¿Porqué no implementar la memoria principal con regitros de la CPU? Para responder esto se debe tener en mente que el costo sea razonable. En general las arquitecturas cuentan con pocas decenas de registros debido al costo que eso implica. Por otro lado, si se implementa la memoria principal con una RAM se otorga flexibilidad para extender su tamaño pues se trata de un componente externo a la CPU y en cambio los registros suelen estar definidos en el diseño electrónico de la misma. Este esquema se describe en la figura 2.

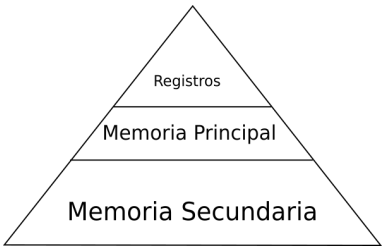

Figure 2: Jerarquía de memoria

2 Memoria Caché

Desde hace un par de décadas, el progreso tecnologico de las computadoras personales esta marcando una tendencia a duplicar, cada año y medio, la velocidad de los procesadores y el tamaño de la memoria principal pero la velocidad de las memorias apenas crece un 10%. Esto ocasiona un crecimiento de la brecha entre la velocidad del procesador y la velocidad de la memoria, causando un mayor impacto en el tiempo de ejecución de los programas, pues la velocidad con la que la CPU puede ejecutar instrucciones está limitada por el tiempo de acceso a memoria, dado que al menos una vez es necesario accederla dentro del ciclo de ejecución de instrucción.

Un enfoque para buscar una solución a esta brecha es incorporar una memoria mas pequeña que la memoria principal pero mas rápida, teniendo en mente que no es viable construir la memoria principal a partir de registros internos a la CPU. Esta idea se basa en que los accesos que se solicitan no son al azar, sino que respetan una lógica que tiene relación con la naturaleza de la ejecución de los programas, pudiendose distinguir dos tipos de accesos: lectura de instrucciones y lectura de datos. La lectura de instrucciones es, en su mayoría, un recorrido de celdas consecutivas de memoria, y asi mismo la lectura de los datos de un arreglo también lo es. Por otro lado, las instrucciones de una rutina o de una repetición se utilizan mas de una vez. Este patrón de acceso se describe con los **principios de localidad**: el principio de localidad espacial enuncia que las posiciones de memoria cercanas a alguna accedida recientemente son mas probables de ser requeridas que las mas distantes, y el principio de localidad temporal dice que cuando un programa hace referencia a una posicion de memoria, se espera que vuelva a hacerlo en poco tiempo.

A partir de esta idea, se busca tener las celdas mas usadas (y no todas) en una memoria mas inmediata, intermedia a la cpu y la memoria principal (ver figura 3), denominada **Memoria Caché**.

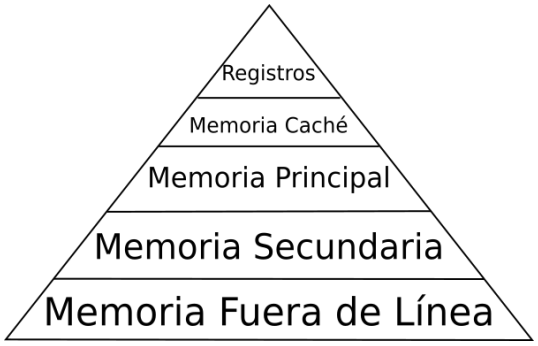

Figure 3: Jerarquía con Memoria Caché

La memoria caché tiene como objetivo proveer una velocidad de acceso cercana a la de los registros pero al mismo tiempo proveer un mayor tamaño de memoria. Para esto, la caché contiene una copia de porciones de la memoria principal, y en el momento de leer una celda, primero se verifica si se encuentra en la caché. **Si esto no ocurre** debe traerse de la memoria principal un bloque de celdas a la caché y luego la celda se entrega a la CPU. El principio de localidad dice que cuando un bloque de datos es traído a la caché porque se necesitó una de sus celdas, entonces es probable que próximamente se necesiten otras celdas del mismo bloque.

3 Memoria Secundaria

3.1 Discos Magnéticos

Para la lectura o la escritura del disco, el cabezal debe estar posicionado al comienzo del sector deseado. En el caso de los dispositivos de cabezal móvil, el posicionamiento implica mover el cabezal hasta la pista deseada (*seek*) y luego esperar a que el sector deseado pase por debajo del cabezal (*latency*). De esta manera, el **tiempo de acceso al disco** es la suma del seek time, el latency time y el tiempo de transmisión.

Tiempo de Búsqueda (seek time): Es el tiempo que le toma a las cabezas de Lectura/Escritura moverse desde su posición actual hasta la pista donde esta localizada la información deseada. Como la pista deseada puede estar localizada en el otro lado del disco o en una pista adyacente, el tiempo de búsqueda variara en cada búsqueda. En la actualidad, el tiempo promedio de búsqueda para cualquier búsqueda arbitraria es igual al tiempo requerido para mirar a través de la tercera parte de las pistas. Los HD de la actualidad tienen tiempos de búsqueda pista a pista tan cortos como 2 milisegundos y tiempos promedios de búsqueda menores a 10 milisegundos y tiempo máximo

de búsqueda (viaje completo entre la pista más interna y la más externa) cercano a 15 milisegundos .

Latencia (latency): Cada pista en un HD contiene múltiples sectores una vez que la cabeza de Lectura/Escritura encuentra la pista correcta, las cabezas permanecen en el lugar e inactivas hasta que el sector pasa por debajo de ellas. Este tiempo de espera se llama latencia. La latencia promedio es igual al tiempo que le toma al disco hacer media revolución y es igual en aquellos drivers que giran a la misma velocidad. Algunos de los modelos más rápidos de la actualidad tienen discos que giran a 10000 RPM o más reduciendo la latencia.

Command Overhead: Tiempo que le toma a la controladora procesar un requerimiento de datos. Este incluye determinar la localización física del dato en el disco correcto, direccionar al "actuador" para mover el rotor a la pista correcta, leer el dato, redireccionarlo al computador.

Transferencia: Los HD también son evaluados por su transferencia, la cual generalmente se refiere al tiempo en la cual los datos pueden ser leídos o escritos en el drive, el cual es afectado por la velocidad de los discos, la densidad de los bits de datos y el tiempo de acceso. La mayoría de los HD actuales incluyen una cantidad pequeña de RAM que es usada como cache o almacenamiento temporal. Dado que los computadores y los HD se comunican por un bus de Entrada/Salida, el tiempo de transferencia actual entre ellos esta limitado por el máximo tiempo de transferencia del bus, el cual en la mayoría de los casos es mucho más lento que el tiempo de transferencia del drive.

3.2 Discos de estado sólido

Una unidad de estado sólido o SSD (acrónimo en inglés de *solid-state drive*) es un dispositivo de almacenamiento de datos que usa una memoria no volátil, como la memoria flash, para almacenar datos, en lugar de los platos giratorios magnéticos encontrados en los discos mencionados antes.

En comparación con los anteriores, las unidades de estado sólido son menos sensibles a los golpes, son prácticamente inaudibles y tienen un menor tiempo de acceso y de latencia.

La mayoría de los SSD utilizan memoria flash basada en compuertas NAND, que retiene los datos sin alimentación. Para aplicaciones que requieren acceso rápido, pero no necesariamente la persistencia de datos después de la pérdida de potencia, los SSD pueden ser construidos a partir de memoria de acceso aleatorio (RAM). Estos dispositivos pueden emplear fuentes de alimentación independientes, tales como baterías, para mantener los datos después de la desconexión de la corriente eléctrica.

3.3 Redundant Array of Inexpensive Drives

RAID es una tecnología que emplea el uso simultáneo de dos o mas discos duros para alcanzar mejor performance, mayor fiabilidad y mayor capacidad de almacenamiento. Los diseños de RAID involucran dos objetivos de diseño: mejor fiabilidad y mejor performance de lectura y escritura. Cuando un conjunto de discos físicos se utilizan en un RAID, se los denomina conjuntamente **vector RAID**. Este vector distribuye la información a través de varios discos, pero esto es transparente para el sistema operativo, pues lo maneja como si se tratara de un solo disco.

Algunos vectores RAID son redundantes en el sentido que se almacena información extra, derivada de la información original, de manera que el fallo de un disco en el vector no provocará pérdida de infomación. Una vez reemplazado el disco, su información se reconstruye a partir de los otros discos y la información extra. Claramente, un vector redundante tiene menos capacidad de almacenamiento.

Existen varias combinaciones de estos enfoques dando diferentes relaciones de compromiso entre proteccion contra la perdida de datos, capacidad y velocidad. Estas combinaciones se denominan niveles, y los niveles 0,1 y 5 son los mas comunmente encontrados pues cubren la mayoría de los requerimientos.

RAID 0 (*striped disks*): Distribuye la información a través de distintos discos de manera que se mejore la velocidad y la capacidad total, pero toda la información se pierde si alguno de los discos falla.

RAID 1 (*mirrored disks*): Utiliza dos (en algunos casos mas) discos que almacenan exactamente la misma información. Esto permite que la información no se pierda mientras al menos un disco funcione. La capacidad total de este esquema es la misma que la de un disco, pero el fallo de un dispositivo no compromete el funcionamento del arreglo de discos.

RAID 5 (*striped disks with parity*): Este esquema combina tres o mas discos de una manera que se protege la información contra la perdida de un disco y la capacidad de almacenamiento del arreglo se reduce en un disco.

La tecnología RAID involucra importantes niveles de cálculo durante lecturas y escrituras. Si el RAID está implementado por hardware, esta tarea es llevada a cabo por el controlador. En otros casos, el sistema operativo requiere que el procesador lleve a cabo el mantenimiento del RAID, lo que impacta en la performance del sistema.

Los RAID redundantes pueden continuar trabajando sin interrupcion cuando ocurre una falla, pero son vulnerables a fallas futuras. Algunos sistemas de alta disponibilidad permiten que el disco con fallas sea reemplazado sin necesidad de reiniciar el sistema.

Nivel de redundancia

$$red = capTotal/capU$$

Fuente

1. *Organización y Arquitectura de Computadoras , William Stallings, Capítulo 4*