MEMORIA CACHE

Introducción

La memoria cache es una memoria pequeña y rápida que se interpone entre el CPU y la memoria principal para que el conjunto de opere a mayor velocidad. Para ello es necesario mantener en la caché aquellas zonas de memoria principal con mayor probabilidad de ser referenciadas. Esto es posible gracias a la propiedad de localidad de los programas.

¿Qué es caché?

La mayoría de las CPUs modernas de escritorio y servidor tienen al menos tres cachés independientes: un caché de instrucciones para acelerar el acceso a la instrucción ejecutable, un caché de datos para acelerar la recuperación de datos y la memoria principal. Utilizado para acelerar la traducción de direcciones virtuales a físicas para las instrucciones ejecutables y los datos. La memoria caché de datos generalmente está organizada como una jerarquía con múltiples niveles de caché, p. L1 / L2, etc. Esto también se conoce como cachés de niveles múltiples.

Por qué fue necesaria la implementación de memorias cache

La velocidad de la memoria se ha distanciado progresivamente de la velocidad de los procesadores
En la figura siguiente se muestran las gráficas de la evolución experimentada por el rendimiento de las CPUs y las memoria DRAM (soporte de la memoria principal de los computadores actuales) en los últimos años. Las curvas muestran que el rendimiento de la CPU aumentó un 35% anual desde 1980 hasta 1987; y un 55% anual a partir de ese año. En cambio la memoria ha mantenido un crecimiento sostenido del 7% anual desde 1980 hasta la fecha. Esto significa que si se mantiene la tendencia, el diferencial de rendimiento no sólo se mantendrá sino que aumentará en el futuro. Para equilibrar esta diferencia se viene utilizando una solución arquitectónica: la memoria caché.

Solución al problema

En la construcción del sistema de memoria se persiguen tres objetivos principales alta velocidad, alta capacidad y bajo coste si analizamos los diferentes tipos de tecnologías de memoria existen en la actualidad SRAM, DRAM y almacenamiento magnético (Pero como sabemos no hay ninguna tecnología que por si sola satisfaga los requisitos antes mencionados debido a que:

La SRAM es la más rápida pero enormemente costosa.(utilizado comúnmente en la cache)
La tecnología DRAM se encuentra en el punto intermedio, no es suficientemente rápida, ni suficientemente grande, pero tampoco excesivamente costosa.(utilizada comúnmente en memorias ram)

Como consecuencia de esto la solución al problema reside en combinar memorias rápidas y pequeñas con memorias lentas y grandes de tal forma que:
-las memorias rápidas y pequeñas contengan los datos con más probabilidad de acceso
-las memorias lentas y grandes contengan los datos con menos probabilidad de acceso

Con esto se consigue que la capacidad del sistema de memoria sea elevada pues contiene memorias grandes. Además en mayor parte de los casos el sistema de memoria será rápido pues en la mayor parte de los casos acedera a las memorias rápidas
Para poder llevar a cabo la práctica de la solución anterior es necesario que se cumplan las condiciones

Conocer a priori los datos con mayor probabilidad de acceso futuro para así poder almacenarlos en las memorias pequeñas y rápidas.

Que estos datos sean pocos y con una gran probabilidad de ser accedidos en el futuro De esta forma cabrían en las memorias pequeñas y rápidas y además servirían la mayor parte de los accesos futuros, obteniéndose un sistema de memoria rápido y en la mayor parte de los casos

La combinación de las diferentes tecnologías de memoria se hace a niveles de diferentes capacidad y velocidad, formando como lo que se conoce como jerarquía de memoria

Jerarquía de memoria

La CPU cuando desea acceder a uno posición de memoria del sistema accede al nivel de la memoria cache, si esta posición se encuentra dentro de la memoria chache la CPU accede a ella de forma muy rápida. Esta es la situación habitual pues la mayor parte de los accesos de la CPU al sistema de memoria son servidos directamente por la cache

Si por el contrario la posición no está dentro de la memoria cache, se produce un fallo que trae como consecuencia que un conjunto de posiciones consecutivas que incluyen la posición deseada se transfieran desde la memoria principal a la memoria cache. A este conjunto de posiciones de memoria consecutivas a la cache. En general cada vez que se produce un fallo en un nivel N del sistema de memoria se copia del nivel N+1 al nivel N el bloque que contiene la posición deseada y a continuación se realiza el proceso

Niveles de cache

Inicialmente cuando se introdujeron las cachés, el sistema típico tenía una única caché. Sin embargo recientemente lo habitual es disponer de dos niveles de caché o incluso tres en computadores de gama alta. Estos niveles se denominan L1, L2, etc., donde el nivel L1 es el más cercano al procesador.

El objetivo de la caché L2 es reducir la penalización debida a los fallos de caché del nivel L1. En teoría, cuantos más niveles de caché haya entre el procesador y la memoria principal mejor rendimiento proporciona el sistema de memoria pero con un coste mayor.

La organización de memoria caché ARM ha evolucionado con la arquitectura general de la familia ARM, lo que refleja la búsqueda incesante de rendimiento que es la fuerza motriz de todos los diseñadores de microprocesadores. La tabla 4.6 muestra esta evolución. Los modelos ARM7 usaron un caché L1 unificado, mientras que todos los modelos posteriores usan una caché de instrucciones / datos divididos. Todo el BRAZO

ARM

arquitectura RISC (Reduced Instruction Set Computer)

FIFO es el acronimo de first in, first out

Algoritmos de sustitución

Una característica interesante de la arquitectura ARM es el uso de un pequeño búfer de escritura de primer agotamiento (FIFO) para mejorar el rendimiento de escritura de la memoria. El buffer de escritura se interpone entre el caché y la memoria principal y consiste en un conjunto de direcciones y un conjunto de palabras de datos. El buffer de escritura es pequeño en comparación con el caché, y puede contener hasta cuatro direcciones independientes.

El búfer de escritura funciona de la siguiente manera: cuando el procesador realiza una escritura en un área de búfer, los datos se colocan en el búfer de escritura a la velocidad del reloj del procesador y el procesador continúa la ejecución. Se produce una escritura cuando los datos en la memoria caché se escriben en la memoria principal. Por lo tanto, los datos que se escriben se

transfieren de la memoria caché al búfer de escritura. El búfer de escritura realiza la escritura externa en paralelo. Sin embargo, si el búfer de escritura está lleno (ya sea porque ya hay el número máximo de palabras en el búfer o porque no hay ranura para la nueva dirección), entonces el procesador se detiene hasta que haya suficiente espacio en el búfer. A medida que avanzan las operaciones de no escritura, el búfer de escritura continúa escribiendo en la memoria principal hasta que el búfer esté completamente vacío. Los datos escritos en el búfer de escritura no están disponibles para volver a leerlos en la memoria caché hasta que los datos se hayan transferido del búfer de escritura a la memoria principal. Esta es la razón principal por la que el buffer de escritura es bastante pequeño. Aun así, a menos que haya una alta proporción de escrituras en un programa en ejecución, el buffer de escritura mejora el rendimiento.

¿Qué es el buffer de escritura?

Una memoria muy pequeña (FIFO) colocada entre el núcleo del procesador y la memoria principal. El objetivo es liberar el núcleo del procesador y la memoria caché del tiempo de escritura lento asociado con la escritura en la memoria principal.

Estrategias de escritura

Las estrategias de escritura establecen la relación entre las escrituras en la memoria caché y la memoria principal. El objetivo fundamental de estas estrategias es garantizar la coherencia entre la información de la caché y la memoria principal, afectando lo mínimo posible al rendimiento del sistema.

Política de escritura

Determina la forma de actualizar Mp cuando se realizan operaciones de escritura. Hay que diferenciar dos casos: cuando la posición de memoria sobre la que se va a escribir está en Mc (acierto) y cuando no lo está (fallo). Frente a aciertos en la caché existen dos alternativas: escritura directa y postescritura. Y frente a fallos en la caché otras dos: asignación en escritura y no asignación.

(Acierto)

Escritura directa o inmediata (write through)
-Todas las operaciones de escritura se realizan en Mc y Mp
- Inconveniente: genera un tráfico importante a Mp
- Solución: utilización de un buffer de escritura

Postescritura (copy back)
-Las actualizaciones se hacen sólo en Mc
- Se utiliza un bit de actualización asociado a cada marco de bloque para indicar la escritura del marco en Mp cuando es sustituido por la política de reemplazamiento
-Inconveniente: inconsistencia temporal entre Mc y Mp ==> complicación del acceso de la E/S a memoria que debe realizarse a través de Mc.

(Fallo)

Asignación en escritura (write allocate)
-El bloque se ubica en Mc cuando ocurre el fallo de escritura y a continuación se opera como en un acierto de escritura, es decir, con wirte through o copy back

No asignación en escritura (No write allocate)
-El bloque se modifica en Mp sin cargarse en Mc

Cómo la memoria caché mejora el rendimiento

Principio de la localidad de referencia: los programas de software de computadora frecuentemente ejecutan pequeños bucles de código que operan repetidamente en secciones locales de la memoria de datos.

¿Cuánto caché es suficiente?

Mayor caché conduce a una mayor latencia. El equilibrio es importante. Un procesador con una velocidad de reloj de 2.4 Ghz y 4 megabytes de caché L2 teóricamente funcionaría igual que un procesador de 2.6 Ghz con 2 megabytes de caché.

Memoria caché

La memoria caché es la memoria más rápida del sistema de memoria. En la actualidad se implementa con chips de memoria RAM estática, los cuales son relativamente caros y de baja capacidad.

La memoria caché se divide en bloques, todos ellos conteniendo el mismo número de bytes. Cada bloque contiene un conjunto de datos ubicados en posiciones de memoria consecutivas. Cada vez que la CPU intenta acceder a una dirección de memoria que no está en ningún bloque de la caché se produce un fallo de caché, y en caso contrario un acierto de caché. La consecuencia del fallo es que se suspende momentáneamente el acceso, se copia el bloque que contiene la dirección desde la memoria principal a la memoria caché y a continuación se reanuda el acceso. Desde el punto de vista de interacción con la memoria caché, la memoria principal se puede ver como un gran almacén de bloques del mismo tamaño que los bloques de la memoria caché.

Por ejemplo, una memoria caché de 32 bytes organizada en bloques de 4 bytes tendrá 8 bloques. Si la memoria principal es de 256 bytes, la memoria caché percibe a la memoria principal como 64 bloques de 4 bytes cada uno.

Toda memoria caché lleva asociado un controlador de memoria caché, el cual se encarga de gestionar todas las lecturas y escrituras en la memoria caché y decidir por ejemplo si una dirección está cacheada, es decir, ubicada dentro de la memoria caché, o por el contrario se ha producido un fallo de caché.

En general, el funcionamiento de cualquier memoria caché viene definido por las siguientes características: estrategia de correspondencia, de reemplazo y de escritura.