Memoria caché básica

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Evolución de la latencia

- Hay múltiples visiones del rendimiento.
  - $\textit{rendimiento} = \frac{1}{\textit{latencia}}$
  - Útil para comparar evolución de procesadores y memoria.

- **Procesadores**
  - Incrementos de rendimiento anuales de de entre 25% y 52
  - Efecto combinado de 1980 a 2010 $\rightarrow$ superior a $3,000$ veces.

- **Memoria**
  - Incrementos de rendimiento anuales alrededor del 7%
  - Efecto combinado de 1980 a 2010 $\rightarrow$ alrededor de 7.5 veces.

Efecto multi-core

- **Intel Core i7**.
  - 2 accesos a datos de 64 bits por ciclo.
  - 4 cores, 3.2 GHz $\rightarrow$ 25.6×10^9 accesos/seg
  - Demanda instrucciones: 12.8×10^9 de 128 bits.
  - Pico de ancho de banda: 409.6 GB/seg
- **Memoria SDRAM**.
  - DDR2 (2003): 3.2 GB/seg – 8.5 GB/seg
  - DDR3 (2007): 8.53 GB/seg – 18 GB/seg
  - DDR4 (2014): 17.06 GB/seg – 25.6 GB/seg
- **Soluciones**:
  - Memoria multipuerta, cachés segmentadas, cachés multinivel, cachés por core, separación instrucciones y datos.

Principio de localidad

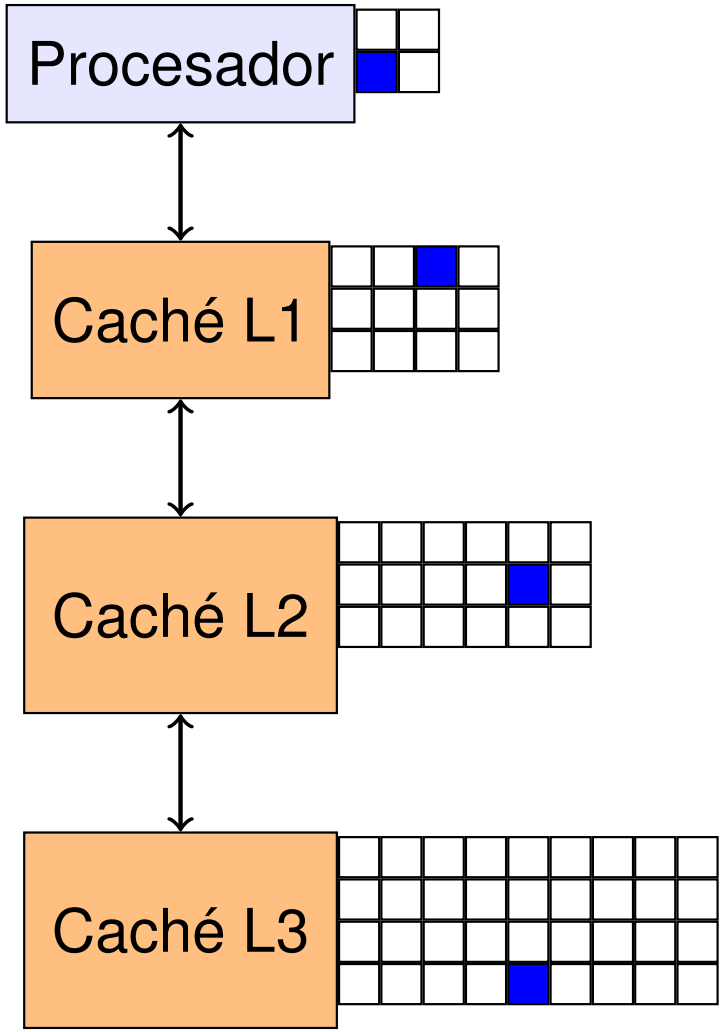
- **Principio de localidad**.
  - Es una **propiedad de los programas** que se explota en el diseño del hardware.
  - Los programas acceden una porción relativamente pequeña del espacio de direcciones.

- **Tipos de localidad**:
  - **Localidad temporal**: Los elementos accedidos recientemente tienden a volver a ser accedidos.
    - Ejemplos: Bucles, reutilización de variables, ...
  - **Localidad espacial**: Los elementos próximos a los accedidos recientemente tienden a ser accedidos.
    - Ejemplos: Ejecución secuencial de instrucciones, arrays, ...

Situación (2008)

- **SRAM** → *Static RAM*.
  - **Tiempo de acceso**: 0.5 ns – 2.5 ns.
  - **Coste por GB**: 2,000$ - 5,000$
- **DRAM** – *Dynamic RAM*.
  - **Tiempo de acceso**: 50ns – 70 ns
  - **Coste por GB**: 20$ - 75$.
- **Disco magnético**.
  - **Tiempo de acceso**: $5,000,000$ ns – $20,000,000$ ns.
  - **Coste por GB**: 0.20$ - 2$.

Jerarquía de memoria



Jerarquía de memoria

- **Bloque o línea**: Unidad de copia.
  - Suele estar formada por múltiples palabras.
- Si dato accedido está **presente** en nivel superior:
  - **Acierto**: Satisfecho por nivel superior.

$$h = \frac{aciertos}{accesos}$$

- Si dato accedido está **ausente**.
  - **Fallo**: Bloque se copia de nivel inferior.
  - Acceso a dato en nivel superior.
  - Tiempo necesario $\rightarrow$ **Penalización de fallo**.

$$m = \frac{fallos}{accesos} = 1 - h$$

Medidas

- Tiempo medio de acceso a memoria:

$$t_M = t_A + (1 - h)t_F$$

- **Penalización de fallo**:
  - Tiempo en remplazar y bloque y entregarlo a CPU.
  - **Tiempo de acceso**.
    - Tiempo de obtenerlo de nivel inferior.
    - Depende de latencia a nivel inferior.
  - **Tiempo de transferencia**.
    - Tiempo de transferir un bloque.
    - Depende de ancho de banda entre niveles.

Medidas

- **Tiempo de ejecución de CPU**:

$$t_{CPU} = (ciclos_{CPU} + ciclos_{\text{detencion memoria}}) \times t_{ciclo}$$

- **Ciclos de reloj CPU**:

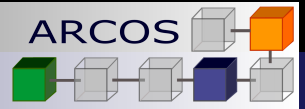
$$ciclos_{CPU} = IC \times CPI$$

- **Ciclos de detención de memoria**:

$$ciclos_{\text{detención memoria}} = n_{fallos} \times \text{penalización}_{fallo} =$$

$$IC \times fallo_{instr} \times \text{penalización}_{fallo} =$$

$$IC \times accesos\_memoria_{instr} \times (1 - h) \times \text{penalización}_{fallo}$$



Cuatro preguntas sobre la jerarquía de memoria

1. ¿Dónde se ubica un bloque en el nivel superior?
   - **Ubicación de bloque**.
2. ¿Cómo se localiza un bloque en el nivel superior?
   - **Identificación de bloque**.
3. ¿Qué bloque debe remplazarse en caso de fallo?
   - **Remplazo de bloque**.
4. ¿Qué ocurre en caso de escritura?
   - **Estrategia de escritura**.

P1: Ubicación de bloque

- **Correspondencia directa**.
  - Ubicación $\rightarrow$ *bloque* mod $n_{bloques}$
- **Correspondencia totalmente asociativa**.
  - Ubicación $\rightarrow$ Cualquier lugar.
- **Correspondencia asociativa por conjuntos**.
  - Ubicación de conjunto $\rightarrow$ *bloque* mod $n_{conjuntos}$
  - Ubicación dentro de conjunto $\rightarrow$ Cualquier lugar de conjunto.

P2: Identificación de bloque

- **Dirección de bloque**:
  - **Etiqueta**: Identifica la dirección en entrada.
    - Bit validez en cada entrada indica el contenido es válido.
  - **Índice**: Selecciona el conjunto.
- **Desplazamiento de bloque**:
  - Selecciona datos dentro de un bloque.
- **Con mayor asociatividad**:
  - Menos bit de **índice**.
  - Más bits de **etiqueta**.

P3: Remplazo de bloque

- **Relevante** para **asociativa** y **asociativa por conjuntos**:
  - **Aleatorio**.
    - Fácil de implementar.
  - **LRU**: Menos recientemente usado.
    - Complejidad creciente cuando aumenta asociatividad.
  - **FIFO**.
    - Aproxima LRU con menor complejidad.

	2 vías			4 vías			8 vías		
Tam.	LRU	Rand	FIFO	LRU	Rand	FIFO	LRU	Rand	FIFO
16 KB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

Fallos por 1000 instr., SPEC 2000.

Fuente: Computer Architecture: A Quantitative Approach. 5 Ed
Hennessy and Patterson. Morgan Kaufmann. 2012.

P4: Estrategia de escritura

Escritura inmediata (*write-through*)

- Todas las escrituras van a bus y memoria.
- Fácil de implementar.
- Problemas de rendimiento en SMP's

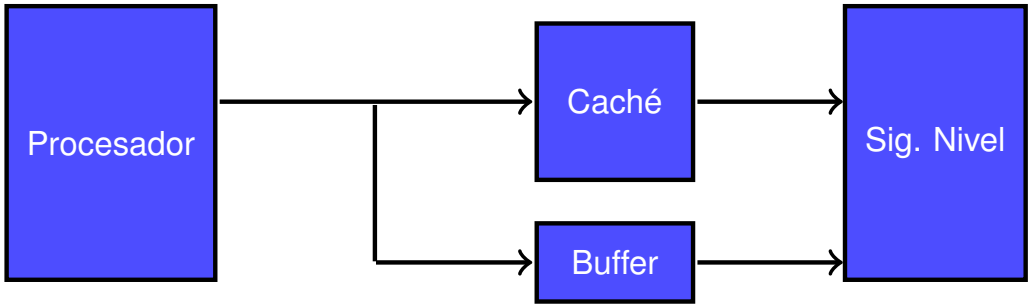
Post-escritura (*write-back*)

- Muchas escrituras son aciertos.
- Aciertos de escritura no van a bus y memoria.
- Problemas de propagación y serialización.
- Más complejo.

P4: Estrategia de escritura

- **¿Donde se escribe?**:
  - **write-trhough**: En bloque de caché y siguiente nivel de memoria.
  - **write-back**: Solamente en bloque de caché.
- **¿Qué ocurre cuándo bloque sale de caché?**:
  - **write-trhough**: Nada más.
  - **write-back**: Se actualiza siguiente nivel de memoria.
- **Depuración**:
  - **write-trhough**: Fácil.
  - **write-back**: Difícil.
- **Fallo provoca escritura**:
  - **write-trhough**: No.
  - **write-back**: Si.
- **Escritura repetida va al siguiente nivel**:
  - **write-trhough**: Si.
  - **write-back**: No.

Búfer de escritura



- **¿Por qué un búfer?**
  - Para evitar detenciones de CPU.
- **¿Por qué un búfer y no un registro?**
  - Ráfagas de escritura son comunes.
- **¿Puede haber riesgos RAW?**
- Si.
- **Alternativas**:
  - Vaciar búfer antes de lectura.
  - Comprobar el búfer antes de leer.

Penalización de fallo

- **Penalización de fallo**:
  - Latencia total del fallo.
  - Latencia expuesta (la que obliga a detener CPU).

Penalización de fallos

$$\frac{\text{ciclos\_detención}_{memoria}}{IC} =$$

$$\frac{fallos}{IC} \times \left(latencia_{total} - latencia_{solapada}\right)$$

Optimizaciones básicas de caché

- **Reducción** de **tasa de fallos**.
  - Aumento de tamaño de bloques.
  - Aumento de tamaño de caché.
  - Incremento de asociatividad.
- **Reducción** de **penalización de fallos**.
  - Cachés multinivel.
  - Prioridad de lecturas sobre escrituras.
- **Reducción** de **tiempo de acierto**.
  - Evitar traducción de direcciones en indexado en caché.

1: Aumento de tamaño de bloque

- Mayor tamaño de bloque → **menor la tasa de fallos**.
  - Mejor aprovechamiento de localidad espacial.

- Mayor tamaño de bloque → **mayor penalización por fallo**.
  - En caso de fallo hay que transferir bloques más grandes.
  - Más fallos porque la caché tiene menos bloques.

Necesidad de equilibrio:

- Memoria de alta latencia y alto ancho de banda:
  - Incrementar tamaño de bloque.
- Memoria con baja latencia y bajo ancho de banda:
  - Tamaño de bloque reducido.

2: Aumento de tamaño de la caché

- Mayor tamaño de caché $\rightarrow$ **menor tasa de fallos**.
  - Caben más datos en la caché.
- Puede **incrementar el tiempo de acierto**.
  - Hace falta más tiempo para encontrar el bloque.
- **Mayor coste**.
- **Mayor consumo de energía**.
- **Necesidad de encontrar un equilibrio**.
  - Especialmente en cachés on-chip.

3: Incremento de asociatividad

- Mayor asociatividad $\rightarrow$ **menor tasa de fallos**.
  - Hay menos conflictos al poderse usar más vías en un mismo conjunto.
- Puede **incrementar tiempo de acierto**.
  - Hace falta más tiempo para encontrar el bloque.
- **Consecuencia**:
  - 8 vías $\approx$ Totalmente asociativa.

4: Cachés multinivel

- **Objetivo**: **Reducción de la penalización de fallos**.
- **Evolución**:
  - **Mayor distancia** entre rendimiento de DRAM y CPU a lo largo del tiempo.
  - **Coste** de penalización de fallo **creciente**.
- **Alternativas**:
  - Hacer la caché **más rápida**.
  - Hacer la caché **más grande**.
- **Solución**:
  - **¡Las dos cosas!**
  - **Varios niveles de caché**.

Tasa de fallos global y local

- **Tasa de fallos local**:
  - Fallos en un nivel de la caché con respecto a accesos a ese nivel de caché.
  - Tasa de fallos en L1 $\rightarrow m(L1)$
  - Tasa de fallos en L2 $\rightarrow m(L2)$
- **Tasa de fallos global**:
  - Fallos en un nivel de caché con respecto a todos los accesos a memoria.
  - Tasa de fallos en L1 $\rightarrow m(L1)$
  - Tasa de fallos en L2 $\rightarrow m(L1) \times m(L2)$
- **Tiempo medio de acceso**:

$$t_a(L1) + m(L1) \times t_f(L1) =$$

$$t_a(L1) + m(L1) \times (t_a(L2) + m(L2) \times t_f(L2))$$

5: Prioridad de fallos de lectura sobre escritura

- **Objetivo**: **Reduce la penalización de fallos**.
  - Evita que un fallo de lectura tenga que esperar a que una escritura termine.
- **Cachés de escritura inmediata**.
  - El búfer de escritura **podría contener una modificación** del dato que se lee.
    a) Esperar el vaciado del búfer de escritura.
    b) Comprobar los valores del búfer de escritura.
- **Cachés con post-escritura**.
  - Un fallo de lectura **podría remplazar un bloque modificado**.
    - Copiar bloque modificado a búfer, leer y volcar bloque a memoria.
    - Aplicar opciones **a** o **b** al búfer.

3 Optimizaciones básicas

- Reducción de tasa de fallos
- Reducción de penalización de fallos
- Reducción de tiempo de acierto

6: Evitar traducción de direcciones durante indexado

- **Objetivo**: **Reduce el tiempo de acierto**.
- **Proceso de traducción**:
  - Dirección virtual → Dirección física.
  - Puede requerir accesos adicionales a memoria.
    - O al menos a TLB.
- **Idea**: **Optimizar el caso más común** (aciertos).
  - Utilizar las direcciones virtuales para la caché.
- **Tareas**:
  - **Indexar en la caché** → Se usa desplazamiento.
  - **Compara las direcciones** → Se usa dirección física traducida.

Problemas con cachés virtuales (I)

- **Protección**.
  - La protección a nivel de página se comprueba en **traducción de dirección virtual**.
  - **Solución**: Incorporar protección a caché.

- **Cambio de proceso**.
  - Las direcciones virtuales referencian **otras direcciones físicas**.
  - **Soluciones**:
    - Limpiar la caché.
    - Incorporar PID a etiqueta.

Problemas con cachés virtuales (II)

- **Aliasing**:
  - Dos direcciones virtuales diferentes para la misma dirección física.
  - **Hardware anti-aliasing**: Garantiza que cada bloque de caché corresponde a una única dirección física.
    - Comprobar múltiples direcciones e invalidar.
  - **Coloreado de páginas**: Fuerza a todos los alias a tener idénticos sus últimos **n** bits.
    - Hace imposible que dos alias están al mismo tiempo en la caché.
- **Direcciones de entrada/salida**:
  - E/S usa habitualmente direcciones físicas.
  - Correspondencia con direcciones virtuales para interaccionar con cachés virtuales.

Traducción de direcciones

- **Solución**:
  - Indexar virtualmente y usar etiquetas físicas.

- **Tareas**:
  - **Indexar la caché** → Usar desplazamiento de página.
    - Esta parte es idéntica en direcciones físicas y virtuales.
  - **Comparar etiquetas** → Usar direcciones físicas traducidas.
    - Comparación de etiquetas usa direcciones físicas.

Resumen

- Cachés como solución para mitigar el **muro de memoria**.
- La **evolución** tecnológica y el principio de **localidad** generan más presión sobre las cachés.
- **Penalización de fallos** dependiente de **tiempo de acceso** y **tiempo de transferencia**.
- Cuatro dimensiones clave en el **diseño de cachés**:
  - **Ubicación de bloque**, **identificación de bloque**, **remplazo de bloque**, y **estrategia de escritura**.
- Seis **optimizaciones básicas** de caché:
  - **Reducir tasa de fallos**: Aumentar tamaño de bloque, aumentar tamaño de caché, incrementar asociatividad.
  - **Reducir penalización de fallos**: Cachés multi-nivel, dar prioridad a lecturas sobre escrituras.
  - **Reducir tiempo de acierto**: Evitar traducción de direcciones.

Referencias

- **Computer Architecture. A Quantitative Approach**
  5th Ed.
  Hennessy and Patterson.
  Secciones: B.1, B.2, B.3.

- **Ejercicios recomendados**:
  - B.1, B.2, B.3, B.4, B.5, B.6, B.7, B.8, B.9, B.10, B.11

Memoria caché básica

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid