CS4617 Computer Architecture

Lecture 4: Memory Hierarchy 2

Dr J Vaughan

September 17, 2014

Write stall

- Occurs when processor must wait for writes to complete during write-through
- Write stalls can be reduced by using a write buffer
- Processor execution overlapped with memory updating
- Write stalls still possible with write buffers

Two options on a write miss

- Write allocate: A cache line is allocated on the miss, followed by the write actions
- No-write allocate: Write misses do not affect cache. The block is modified in lower-level memory and stays out of cache until a read attempt occurs.

Example

- Fully associative write-back cache, initially empty
- The following sequence of Memory operation [address] occurs
  - Write Mem [100]
  - Write Mem [100]
  - Write Mem [200]
  - Write Mem [200]
  - Write Mem [100]

  Compare number of hits and misses for no-write allocate with write allocate

Solution

- No-write allocate: address 100 not in cache
    - Miss on write [100], no allocation on write
    - Miss on write [100]
    - Address 200 not in cache
    - Miss on read [200], line allocated
    - Hit on write [200]
    - Miss on write [100]
- Total: 4 misses, 1 hit

Solution

- Write allocate
  - Miss on write [100], line allocated
  - Hit on write [100]
  - Miss on read [200], line allocated
  - Hit on write [200]
  - Hit on write [100]
- Total: 2 misses, 3 hits
- Write-back caches tend to use write allocate
- Write-through caches tend to use no-write allocate

Cache performance

$$\textit{Average memory access time} = \textit{Hit time} + \textit{Miss rate} \times \textit{Miss penalty}$$

Example

Size (kB)	**Instruction Cache**	**Data Cache**	**Unified Cache**
16	3.82	40.9	51.0
32	1.36	38.4	43.3

Table: Misses per 1000 instructions for instructions, data and unified caches of different sizes (from Fig. B.6 Hennessy & Patterson)

Which has the lower miss rate: a 16kB instruction cache with a 16kB data cache or a 32kB unified cache?

Cache performance example: assumptions

- Assume 36% of instructions are data transfers
- Assume a hit takes 1 clock cycle
- Assume miss penalty is 100 clock cycles
- A load or store hit takes 1 extra clock cycle on a unified cache if there is only one cache port to satisfy two simultaneous requests.
- Assume write-through caches with a write buffer and ignore stalls due to the write buffer

Cache performance example: Solution for split cache

$$Miss\ \ rate = \frac{\frac{Misses}{1000\ \ instructions}/1000}{\frac{Memory\ \ accesses}{Instruction}}$$

Each instruction access comprises 1 memory access, so

$Miss\ \ rate_{16kB\ \ instruction} = \frac{3.82/1000}{1.0} = 0.004$ misses/memory access

36% instructions are data transfers, so

$Miss\ \ rate_{16kB\ \ data} = \frac{40.9/1000}{0.36} = 0.114$ misses/memory access

Cache performance example: Solution for unified cache

Unified miss rate needs to account for instruction and data accesses
$Miss\ rate_{32kB\ unified} = \frac{43.3/1000}{1.0+0.36} = 0.0318$ misses/memory access
From Fig. B.6, 74% of memory accesses are instruction references.
The overall miss rate for split caches is
$(74\% \times 0.004) + (26\% \times 0.114) = 0.0326$
Thus, a 32kB unified cache has a lower effective miss rate than two 16kB caches.

Cache performance example: solution (ctd)

$Average\ memory\ access\ time =$
$\%instructions \times (Hit\ time + Instruction\ miss\ rate \times Miss\ penalty) +$
$\%data \times (Hit\ time + Data\ miss\ rate \times Miss\ penalty)$
$Average\ memory\ access\ time_{split} =$
$74\% \times (1 + 0.004 \times 200) + 26\% \times (1 + 0.114 \times 200) =$
$(74\% \times 1.8) + (26\% \times 23.8) = 1.332 + 6.188 = 7.52$
$Average\ memory\ access\ time_{unified} =$
$74\% \times (1 + 0.0318 \times 200) + 26\% \times (1 + 1 + 0.0318 \times 200) =$
$74\% \times 7.36) + (26\% \times 8.36) = 5.446 + 2.174 = 7.62$
Thus the split caches in this example (2 memory ports per clock cycle) have a better average memory access time despite the worse effective miss rate.
Note that the miss penalty is 200 cycles here even though the problem stated it as 100 cycles. Also note the addition of an extra cycle in the 1-port unified cache to allow for conflict resolution between the instruction fetch and memory operand fetch/store units.

Average memory access time and processor performance

- If processor executes in-order, average memory access time due to cache misses predicts processor performance
- Processor stalls during misses and memory stall time correlates well with average memory access time
- $\mathit{CPU\ time} = (\mathit{CPU\ execution\ clock\ cycles} + \mathit{Memory\ stall\ clock\ cycles}) \times \mathit{Clock\ cycle\ time}$
- Clock cycles for a cache hit are usually included in CPU execution clock cycles

Example

- In-order execution computer
- Cache miss penalty 200 clock cycles
- Instructions take 1 clock cycle if memory stalls are ignored
- Average miss rate $= 2\%$
- Average 1.5 memory references per instruction
- Average 30 cache misses per 1000 instructions
- Calculate misses per instruction and miss rate

Answer

$$CPU\ time =$$
$$IC \times (CPI_{execution} + \frac{Memory\ \ stall\ \ clock\ \ cycles}{Instruction}) \times Clock\ \ cycle\ \ time$$
$$CPU\ \ time_{with\ \ cache}$$
$$= IC \times [1.0 + (30/1000 \times 200)] \times Clock\ \ cycle\ \ time$$
$$= IC \times 7.0 \times Clock\ \ cycle\ \ time$$

Answer (continued)

- Calculating performance using miss rate

$$\begin{aligned}&\mathit{CPU\ time}\\&= IC \times (CPI_{execution} + \mathit{Miss\ rate} \times \frac{\mathit{Memory\ \ accesses}}{\mathit{Instruction}} \times\\&\mathit{Miss\ penalty}) \times \mathit{Clock\ cycle\ time}\\&\mathit{CPU\ time}_{\mathit{with\ \ cache}}\\&= IC \times (1.0 + (1.5 \times 2\% \times 200)) \times \mathit{Clock\ cycle\ time}\\&= IC \times 7.0 \times \mathit{Clock\ cycle\ time}\end{aligned}$$

- Thus clock cycle time and instruction count are the same, with or without a cache
- CPI ranges from 1.0 for a "perfect cache" to 7.0 for a cache that can miss.
- With no cache, CPI increases further to $1.0 + 200 \times 1.5 = 301$

Cache behaviour is important in processors with low CPI and high clock rates

- The lower the $CPI_{execution}$, the higher the relative impact of a fixed number of cache miss clock cycles
- When calculating CPI, the cache miss penalty is measured in processor clock cycles for a miss
- Thus, even if memory hierarchies for 2 computers are the same, the processor with the higher clock rate has a larger number of clock cycles per miss and therefore a higher memory portion of CPI

Example: Impact of different cache organisations on processor performance

- CPI with perfect cache = 1.6
- Clock cycle time = 0.35ns
- Memory references per instruction = 1.4
- Size of caches = 128kB
- Cache block size = 64 bytes $\implies$ 6-bit offset into block
- Cache organisations
  - Direct mapped
  - 2-way associative $\implies$ 2 blocks/set $\implies$ 128 bytes/set $\implies$ 1k sets $\implies$ 10-bit index

Example (ctd)

- For set-associative cache, tags are compared in parallel. Both are fed to a multiplexor and one is selected if there is a tag match and the valid bit is set (Fig. B.5 Hennessy & Patterson)
- Speed of processor $\propto$ speed of cache hit
- Assume processor clock cycle time must be stretched $\times$ 1.35 to allow for the selection multiplexor in the set associative cache
- Cache miss penalty $=$ 65ns
- Hit time $=$ 1 clock cycle
- Miss rate of direct-mapped 128kB cache $=$ 2.1%
- Miss rate of 2-way set associative 128kB cache $=$ 1.9%
- Calculate
  1. Average memory access time
  2. Processor performance

Solution: Average memory access time

$$\textit{Average memory access time} = \textit{Hit time} + \textit{Miss rate} \times \textit{Miss penalty}$$
$$\textit{Average memory access time}_{1-way}$$
$$= 0.35 + (0.021 \times 65) = 1.72 ns$$
$$\textit{Average memory access time}_{2-way}$$
$$= 0.35 \times 1.35 + (0.019 \times 65)$$
$$= 1.71 ns$$

Note the stretching factor of 1.35 to allow for the multiplexer in the 2-way associative cache.

Average memory access time is better for the 2-way set-associative cache.

Solution: Processor performance

$CPU\ time =$
$IC \times (CPI_{execution} + \frac{Misses}{Instruction} \times Miss\ penalty) \times Clock\ cycle\ time$
$= IC \times [(CPI_{execution} \times Clock\ cycle\ time)$
$+(Miss\ rate \times \frac{Memory\ \ accesses}{Instruction} \times Miss\ penalty \times Clock\ cycle\ time)]$
Substitute 65ns for $(Miss\ penalty \times Clock\ cycle\ time)$
$CPU\ time_{1-way} = IC \times [1.6 \times 0.35 + (0.021 \times 1.4 \times 65)] = 2.47 \times IC$
$CPU\ time_{2-way} = IC \times [1.6 \times 0.35 \times 1.35 + (0.019 \times 1.4 \times 65)]$
$= 2.49 \times IC$
Relative performance is $\frac{CPU\ time_{2-way}}{CPU\ time_{1-way}} = \frac{2.49}{2.47} = 1.01$
Direct mapping is slightly better since the clock cycle is stretched for all instructions in the case of 2-way set-associative mapping, even thought there are fewer misses. Direct-mapped cache is also easier to build.

Miss penalty for out-of-order execution processors

Define the delay in this case to be the total miss latency minus that part of the latency that is overlapped with other productive processor operations.

$$\frac{\textit{Memory stall cycles}}{\textit{Instruction}} = \frac{\textit{Misses}}{\textit{Instruction}} \times (\textit{Total miss latency} - \textit{Overlapped miss latency})$$

- Out-of-order (OOO) execution processors are complex
- Must consider what is start and end of a memory operation in an OOO processor
- Must decide length of latency overlap: what is the start of overlap with processor or when is a memory operation stalling the processor?

Out-of-order execution processors

- In an OOO processor, only committed operations are seen at the retirement pipeline stage
- Define a processor as being stalled in a clock cycle if it does not retire the maximum possible number of instructions in that cycle
- Attribute the stall to the first instruction that could not be retired
- Note that applying an optimisation to improve a stall time may not improve execution time because another type of stall previously hidden behind the targeted stall could be revealed

Out-of-order execution processors (ctd)

- OOO processors can tolerate some cache miss delays without losing performance
- Simulation is normally used in the evaluation of memory hierarchy effects on OOO processor performance.
- The start of a memory operation can be measured from the time the memory instruction is queued in the instruction window, or when the address is generated or when the instruction is sent to the memory system. It is important to use the same convention in all calculations when making comparisons

Example

- Same as previous example, but this time the processor with the longer cycle time is OOO and still has a direct-mapped cache. In spite of direct mapping (no multiplexer needed), the stretch factor of 1.35 is applied to give a longer cycle time
- Assume that 30% of the 65ns miss penalty can be overlapped, so that the average CPU memory stall time is 45.5ns

Solution

$Average\ memory\ access\ time_{1-way,OOO}$
$= 0.35 \times 1.35 + (0.021 \times 45.5)$
$= 1.43ns$
$CPU\ time_{1-way,OOO}$
$= IC \times [1.6 \times 0.35 \times 1.35 + (0.021 \times 1.4 \times 45.5)]$
$= 2.09 \times IC$
This compares to $2.47 \times IC$ and $2.49 \times IC$ for the direct and 2-way in-order cases respectively.
Thus, even with a slower clock cycle time and a higher miss rate, the OOO computer can be faster if it can hide 30% of the miss penalty.