Master Informatics Eng.

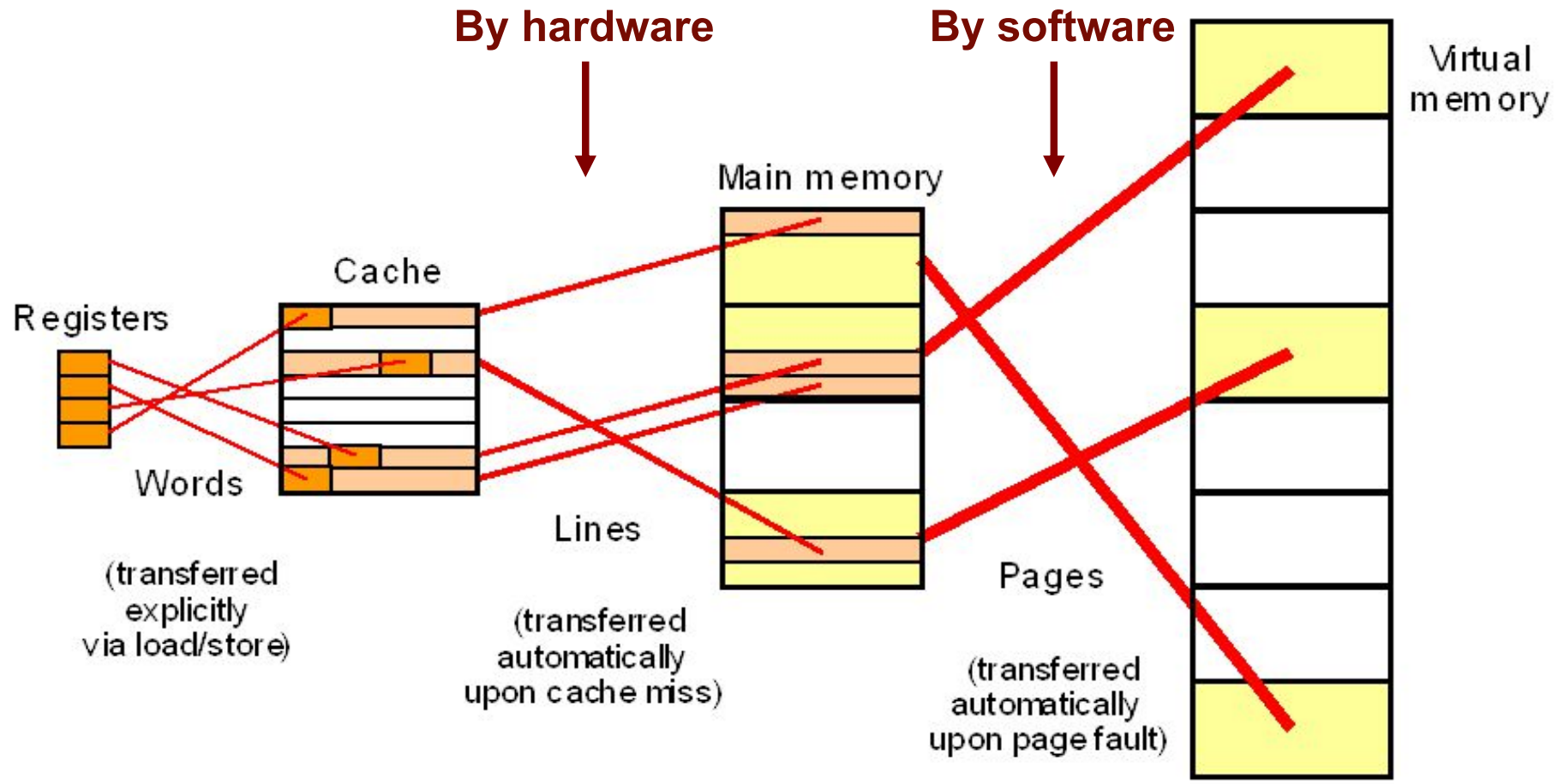
2020/21

A.J.Proença

Memory Hierarchy *(online)*

(most slides are borrowed)

Memory hierarchy: the big picture



Data movement in a memory hierarchy.

Exercise 1 on memory hierarchy

Consider the following study case:

- execution of a piece of code in the SeARCH node with the **Xeon Skylake (Gold 6130)**; detailed info on this Intel microarchitetcure in https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server) ;
- execution of the **same 2 instructions** (that are already in the instruction cache) **in all cores** of a single chip: load in register a double followed by a multiplication by another double in a memory distant location;
- Skylake cores are **6-way superscalar** and each core has 2 load units;
- these instructions are executed with a **cold data cache**.

Compute:

a) **The max required bandwidth** to access the external RAM when executing these 2 instructions (to simplify, consider **clock rate = 2 GHz**).

b) **The peak bandwidth** available in this Xeon device to access the installed DRAM-4 using **all memory channels**.

Resolution

Exercise 1 on memory hierarchy

- each clock cycle needs 2 mem accesses to fetch 2 doubles
- **max required bandwidth** to fetch a cache line
  for each double (cache is cold & doubles are far away):

  ???? GB/s
- note: the following 7 pairs of doubles are already in cache
- RAM in each Skylake Gold 6130: 6x DDR4-2666 (6x8 GiB)
- **peak bandwidth** of 6x DDR4-2666 in 6 memory channels:
  ??? GB/s

Resolution

Exercise 1 on memory hierarchy

- each clock cycle needs 2 mem accesses to fetch 2 doubles
- **max required bandwidth** to fetch a cache line
  for each double (cache is cold & doubles are far away):
  (16 cores x 2 lines x 64 B/line) x clock_frequency =
  2048 B x 2 GHz = **4096 GB/s**
- note: the following 7 pairs of doubles are already in cache
- RAM in each Skylake Gold 6130: 6x DDR4-2666 (6x8 GiB)
- **peak bandwidth** of 6x DDR4-2666 in 6 memory channels:
  6 mem_chan x 2.666 GT/s x 64 b/chan = **128 GB/s**

Exercise 2 on memory hierarchy

Similar to exercise 1 (same node/chip in the cluster), but consider now:

- execution of code taking advantage of the **AVX-512** facilities;
- execution of the same **2 vector instructions** (that are already in the instruction cache) **in all cores**: load in register a vector of doubles followed by a multiplication by another vector of doubles in memory;
- the Skylake cores are **6-way superscalar** and **2-way MT**, and each core supports 2 simultaneous vector loads;
- the Skylake 6130 base clock rate with **AVX-512** code is **1.3 GHz**;
- these instructions are executed with a **cold data cache**.

Compute/estimate:

The **max required bandwidth** to access the external RAM when executing these 2 vector instructions.
Compare with the **peak bandwidth** computed before.

* https://en.wikichip.org/wiki/intel/xeon_gold/6130

Resolution

Exercise 2 on memory hierarchy

- each clock cycle needs 2 mem accesses to fetch 2 vectors with 8 doubles each (512 bits)
- **max required bandwidth** to fetch a cache line for each vector with 8 doubles (cache is cold):

  ???? GB/s

- note: same max required bandwidth as exercise 1, but this **mem access is required at each clock cycle**
- RAM in each Skylake Gold 6130: 6x DDR4-2666 (6x8 GiB)
- **peak bandwidth** of 6x DDR4-2666 in 6 memory channels: **??? GB/s**

Resolution

Exercise 2 on memory hierarchy

- each clock cycle needs 2 mem accesses to fetch 2 vectors with 8 doubles each (512 bits)
- **max required bandwidth** to fetch a cache line for each vector with 8 doubles (cache cold, AVX-512 clock rate lower*):
  (16 cores x 2 lines x 64 B/line) x clock_rate =
  2048 B x **1.3** GHz = **2662.4 GB/s**
- note: same max required bandwidth as exercise 1,
  but this **mem access is required at each clock cycle**
- RAM in each Skylake Gold 6130: 6x DDR4-2666 (6x8 GiB)
- **peak bandwidth** of 6x DDR4-2666 in 6 memory channels:
  6 mem_chan x 2.666 GT/s x 64 b/chan = **128 GB/s**

* https://en.wikichip.org/wiki/intel/xeon_gold/6130

Memory Hierarchy Basics

$$CPU_{exec\text{-}time} = (CPU_{clock\text{-}cycles} + Mem_{stall\text{-}cycles}) \times \text{Clock cycle time}$$

$$CPU_{exec\text{-}time} = (IC \times CPI_{CPU} + Mem_{stall\text{-}cycles}) \times \text{Clock cycle time}$$

With introduction of a single-level cache:

$$Mem_{stall\text{-}cycles} = IC \times \text{... Miss rate ... Mem accesses ... Miss penalty...}$$

$$Mem_{stall\text{-}cycles} = IC \times \text{Misses/Instruction} \times \text{Miss Penalty}$$

$$\frac{\text{Misses}}{\text{Instruction}} = \frac{\text{Miss rate} \times \text{Memory accesses}}{\text{Instruction count}} = \text{Miss rate} \times \frac{\text{Memory accesses}}{\text{Instruction}}$$

For each additional cache-level `i` (including LLC to memory):

$$Mem_accesses_{level_i} = Misses/Instruction_{level_i-1}$$

$$Miss_penalty_{level_i} = (Hit_rate \times Hit_time + Miss_rate \times Miss_penalty)_{level_i+1}$$

Exercise 3 on Cache Performance

- Given
  - I-cache miss rate = 2%
  - D-cache miss rate = 4%
  - Miss penalty = 100 cycles
  - Base CPI (ideal cache) = **2**
  - Load & stores are 36% of instructions
- Miss cycles per instruction
  - I-cache: 0.02 x 100 = 2
  - D-cache: 0.36 x 0.04 x 100 = 1.44
- Actual CPI = 2 + 2 + 1.44 = **5.44**

Exercise 4 on Multilevel Cache

- **CPU**: base CPI = 1, clock rate = 4GHz
- **L1 cache**: L1 miss rate/instruction = 2%
- **L2 cache**: access time = **5ns**, L2 miss rate/instruction = 25%, global miss rate = 2% x 25% = **0.5%**
- **Main memory**: access time = **100ns**
- **With just primary cache**
  - Miss penalty = 100ns / 0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = **9**
- **With L1 & L2 cache**
  - L1 miss penalty, L2 hit = **?? cycles**
  - L2 miss penalty = **?? cycles**
- **CPI** = 1 + **2%** × **??** cycles + **0.5%** × **???** cycles = **???**
- **Performance ratio** = 9 / **???** = **???**

Exercise 4 on Multilevel Cache

- **CPU**: base CPI = 1, clock rate = 4GHz
- **L1 cache**: L1 miss rate/instruction = 2%
- **L2 cache**: access time = **5ns**, L2 miss rate/instruction = 25%, global miss rate = 2% x 25% = **0.5%**
- **Main memory**: access time = **100ns**
- **With just primary cache**
  - Miss penalty = 100ns / 0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = **9**
- **With L1 & L2 cache**
  - L1 miss penalty, L2 hit = 99.5% × 5ns / 0.25ns ≈ **20 cycles**
  - L2 miss penalty = 100ns / 0.25ns = **400 cycles**
- **CPI** = 1 + **2%** × **20** cycles + **0.5%** × **400** cycles = **3.4**
- **Performance ratio** = 9 / **3.4** = **2.6**

Exercise 5 on multilevel performance

Characterize the memory system of Xeon Skylake Gold 6130:

1. **L1 I-cache**
   - size **?** KiB/core, **?**-way set associative, **?** sets, line size **?** B, hit time **?** cycles, **?** B/cycle on transfer bandwidth L1 to the instruction fetch unit

   L1 D-cache
   - size **?** KiB/core, **?**-way set associative, **?** sets, line size **?** B, hit time **?** cycles, **?** B/cycle on load bandwidth L1 to load buffer unit
2. **L2 cache**
   - size **?** KiB/core, **?**-way set associative, **?** sets, line size **?** B, hit time **?** cycles, **?** B/cycle on load bandwidth L2 to L1
3. **L3 cache**
   - size **?** KiB/core, **?**-way set associative, **?** sets, line size **?** B, hit time **?** cycles, **?** B/cycle on load bandwidth L3 to L2
4. **DRAM**, DDR4-2666
   - up to **?** GT/s, bandwidth **?** GB/s per channel, **?** mem channels, aggregate bandwidth **?** GB/s , **?** B/cycle on peak load bandwidth DRAM to L3, NUMA-local latency **?** ns, NUMA-remote latency **?** ns

Exercise 5 on multilevel performance

Characterize the memory system of Xeon Skylake Gold 6130:

1. L1 I-cache

- size **32** KiB/core, **8**-way set associative, **64** sets, line size **64** B, hit time **?** cycles, **16** B/cycle on transfer bandwidth L1 to the instruction fetch unit

L1 D-cache

- size **32** KiB/core, **8**-way set associative, **64** sets, line size **64** B, hit time **4** cycles, **2x64** B/cycle on load bandwidth L1 to load buffer unit

2. L2 cache

- size **1** MiB/core, **16**-way set associative, **1024** sets, line size **64** B, hit time **14** cycles, **64** B/cycle on load bandwidth L2 to L1

3. L3 cache

- size **1.375** MiB/core, **11**-way set associative, **2048** sets, line size **64** B, hit time **50-70** cycles, **64** B/cycle on load bandwidth L3 to L2

4. **DRAM**, DDR4-2666

- up to **2.666** GT/s, bandwidth **21.33** GB/s per channel, **6** mem channels, aggregate on peak load bandwidth DRAM to L3 **128** GB/s,, NUMA-local latency **80** ns, NUMA-remote latency **120-140** ns

Exercise 6 on multilevel performance

Similar to problem 1 (same node/chip in the cluster, code), but consider now:

- execution of scalar code in a **2 GHz** single-core (already in L1 I-cache);
- code already takes advantage of all data cache levels (**L1, L2 & L3**), where 50% of data is placed on the RAM modules in the memory channels of the other PU chip (**NUMA architecture**);
- remember: the Skylake cores are **6-way superscalar** and **2-way MT**, and each core supports **2 simultaneous loads**;
- **cache latency time on hit**: take the average of the specified values;
- **memory latency**: 80 nsec (**NUMA local**), 120 nsec (**NUMA remote**);
- **miss rate per instruction** :
  - at **L1: 2%;** at **L2: 50%;** at **L3: 80%** (these are not global values!).

Compute/estimate:

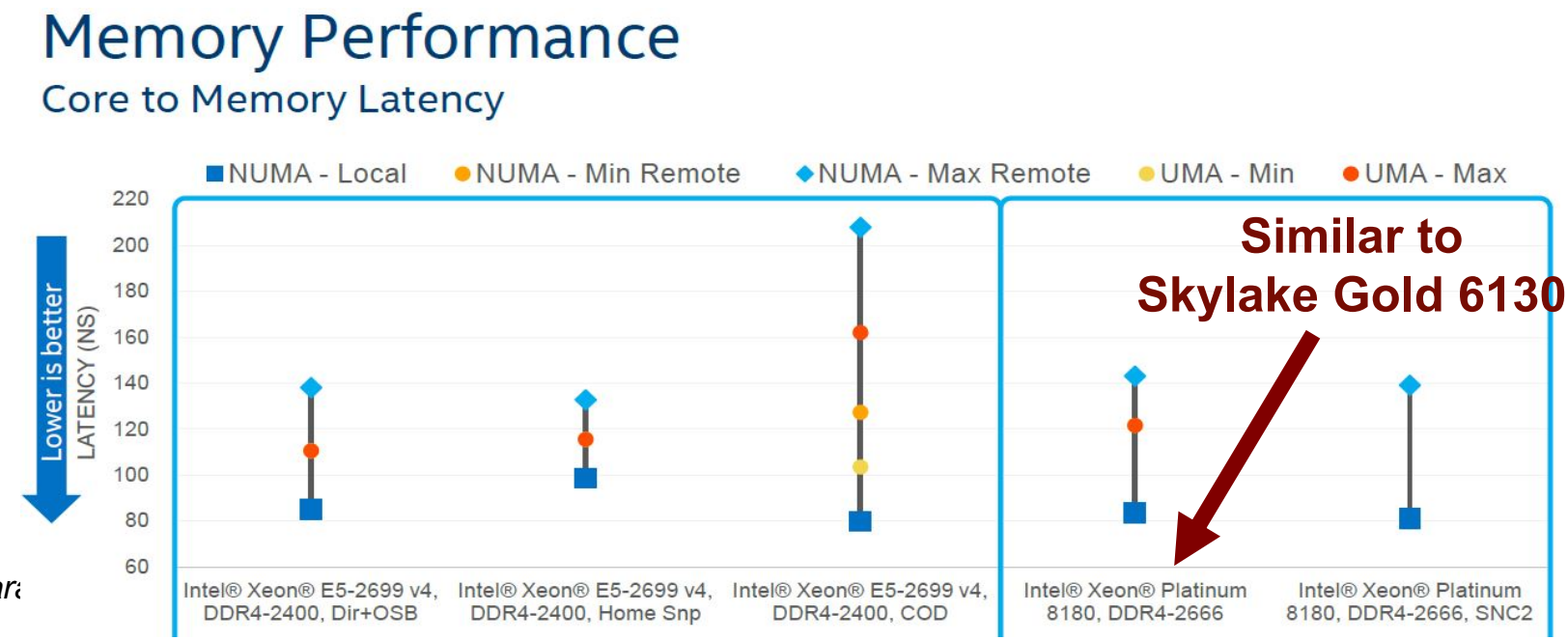
1. **The miss penalty** per instruction at each cache level.
2. **The average memory stall cycles** per instruction that degrades CPI.

Resolution

Exercise 6 on memory hierarchy

- **PU**: base CPI = 1, clock rate = 2 GHz
- **L1 cache**: L1 miss rate/instruction = 2%;
- **L2 cache**: access time = **14** cycles, global miss rate = 2% x 50% = **1%**
- **L3 cache**: access time = **60** cycles, L3 miss rate = 80%,
  global miss rate = 1% x 80% = **0.8%**
- **Main memory**: NUMA local access time = **80ns**, NUMA remote = **120ns**
  average memory access = ((80ns+120ns)/2) / 0.5ns = **200 cycles**

- **CPI?**



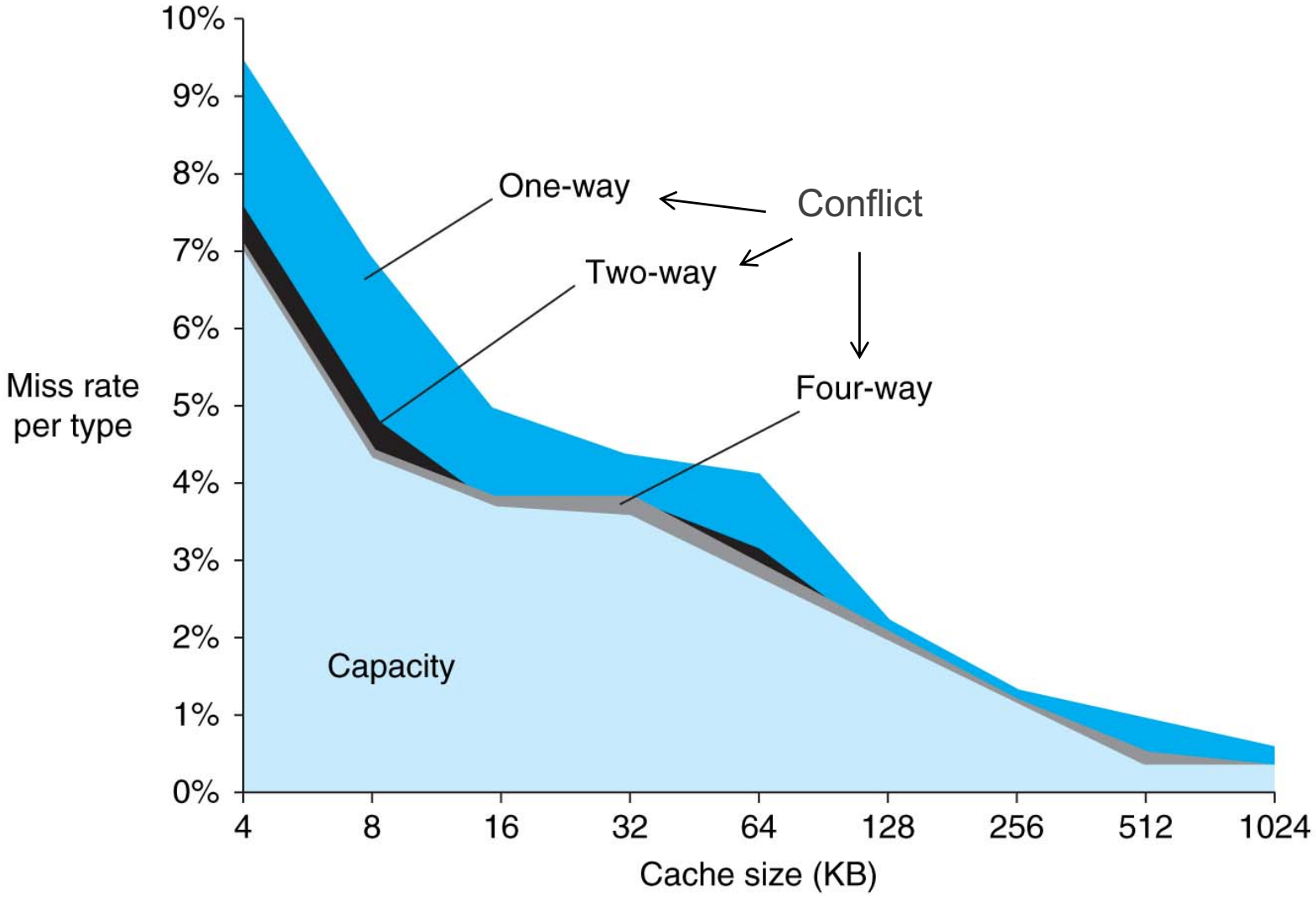
Memory Hierarchy Basics

- Miss rate
  - Fraction of cache access that result in a miss
- Causes of misses (3C's +1)
  - Compulsory
    - First reference to a block
  - Capacity
    - Blocks discarded and later retrieved
  - Conflict
    - Program makes repeated references to multiple addresses from different blocks that map to the same location in the cache
  - **Coherency**
    - **Different processors should see same value in same location**

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.

The 3C's in different cache sizes



MORGAN KAUFMANN
Copyright © 2012, Elsevier Inc. All rights reserved.

The cache coherence pb

- Processors may see different values through their caches:

Time	Event	Cache contents for processor A	Cache contents for processor B	Memory contents for location X
0				1
1	Processor A reads X	1		1
2	Processor B reads X	1	1	1
3	Processor A stores 0 into X	0	1	0

MORGAN KAUFMANN
Copyright © 2012, Elsevier Inc. All rights reserved.

Cache Coherence

- Coherence
  - All reads by any processor must return the most recently written value
  - Writes to the same location by any two processors are seen in the same order by all processors

    (Coherence defines the behaviour of reads & writes to the same memory location)

- Consistency
  - When a written value will be returned by a read
  - If a processor writes location A followed by location B, any processor that sees the new value of B must also see the new value of A

    (Consistency defines the behaviour of reads & writes with respect to accesses to other memory locations)

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.

Enforcing Coherence

- Coherent caches provide:
  - *Migration*: movement of data
  - *Replication*: multiple copies of data

- Cache coherence protocols
  - Directory based
    - Sharing status of each block kept in one location
  - Snooping
    - Each core tracks sharing status of each block

MORGAN KAUFMANN
Copyright © 2012, Elsevier Inc. All rights reserved.

Memory Hierarchy Basics

- Six basic cache optimizations:
  - Larger block size
    - Reduces compulsory misses
    - Increases capacity and conflict misses, increases miss penalty
  - Larger total cache capacity to reduce miss rate
    - Increases hit time, increases power consumption
  - Higher associativity
    - Reduces conflict misses
    - Increases hit time, increases power consumption
  - Multilevel caches to reduce miss penalty
    - Reduces overall memory access time
  - Giving priority to read misses over writes
    - Reduces miss penalty
  - Avoiding address translation in cache indexing
    - Reduces hit time

Copyright © 2012, Elsevier Inc. All rights reserved.

Ten Advanced Optimizations

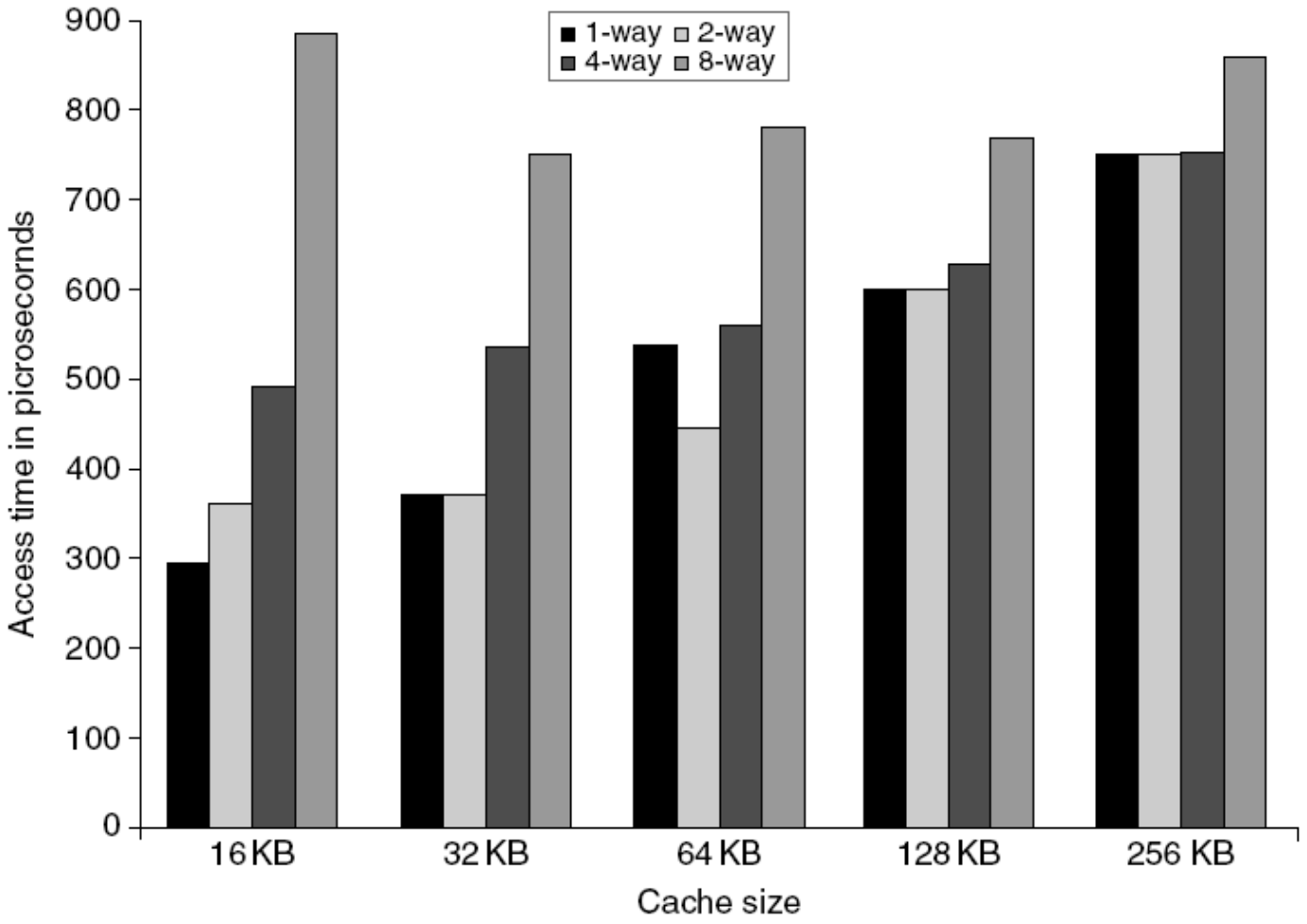
- Reducing the hit time
  1. Small & simple first-level caches
  2. Way-prediction
- Increase cache bandwidth
  3. Pipelined cache access
  4. Nonblocking caches
  5. Multibanked caches
- Reducing the miss penalty
  6. Critical word first
  7. Merging write buffers
- Reducing the miss rate
  8. Compiler optimizations
- Reducing the miss penalty or miss rate via parallelism
  9. Hardware prefetching of instructions and data
  10. Compiler-controlled prefetching

1. Small and simple 1st level caches

- Small and simple first level caches
  - Critical timing path:
    - addressing tag memory, then
    - comparing tags, then
    - selecting correct set
  - Direct-mapped caches can overlap tag compare and transmission of data
  - Lower associativity reduces power because fewer cache lines are accessed

Copyright © 2012, Elsevier Inc. All rights reserved.

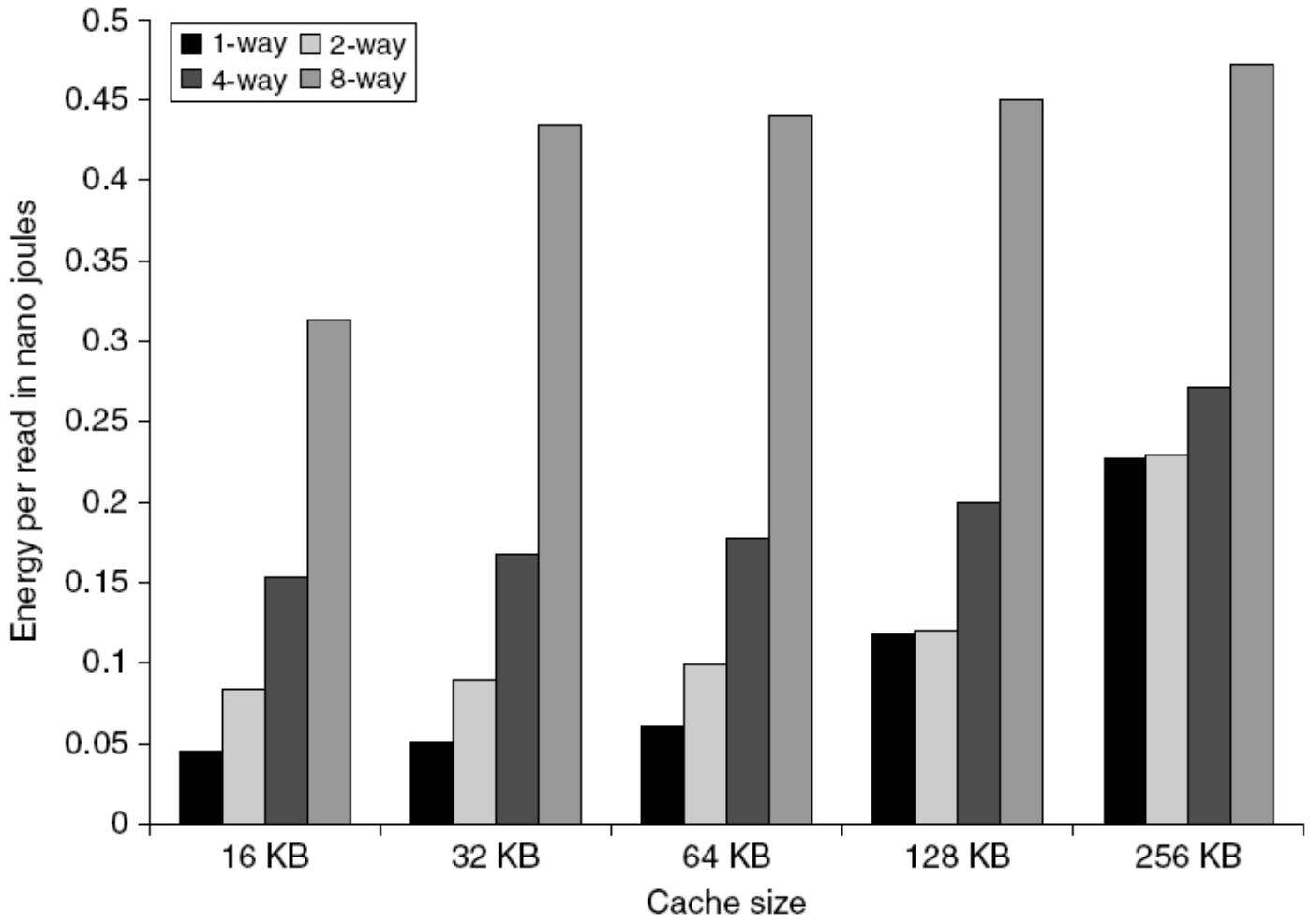
L1 Size and Associativity



Access time vs. size and associativity

MORGAN KAUFMANN
Copyright © 2012, Elsevier Inc. All rights reserved.

L1 Size and Associativity



Energy per read vs. size and associativity

MORGAN KAUFMANN
Copyright © 2012, Elsevier Inc. All rights reserved.

2. Way Prediction

- To improve hit time, predict the way to pre-set mux
  - Mis-prediction gives longer hit time
  - Prediction accuracy
    - > 90% for two-way
    - > 80% for four-way
    - I-cache has better accuracy than D-cache
  - First used on MIPS R10000 in mid-90s
  - Used on ARM Cortex-A8
- Extend to predict block as well
  - “Way selection”
  - Increases mis-prediction penalty

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.

3. Pipelining Cache

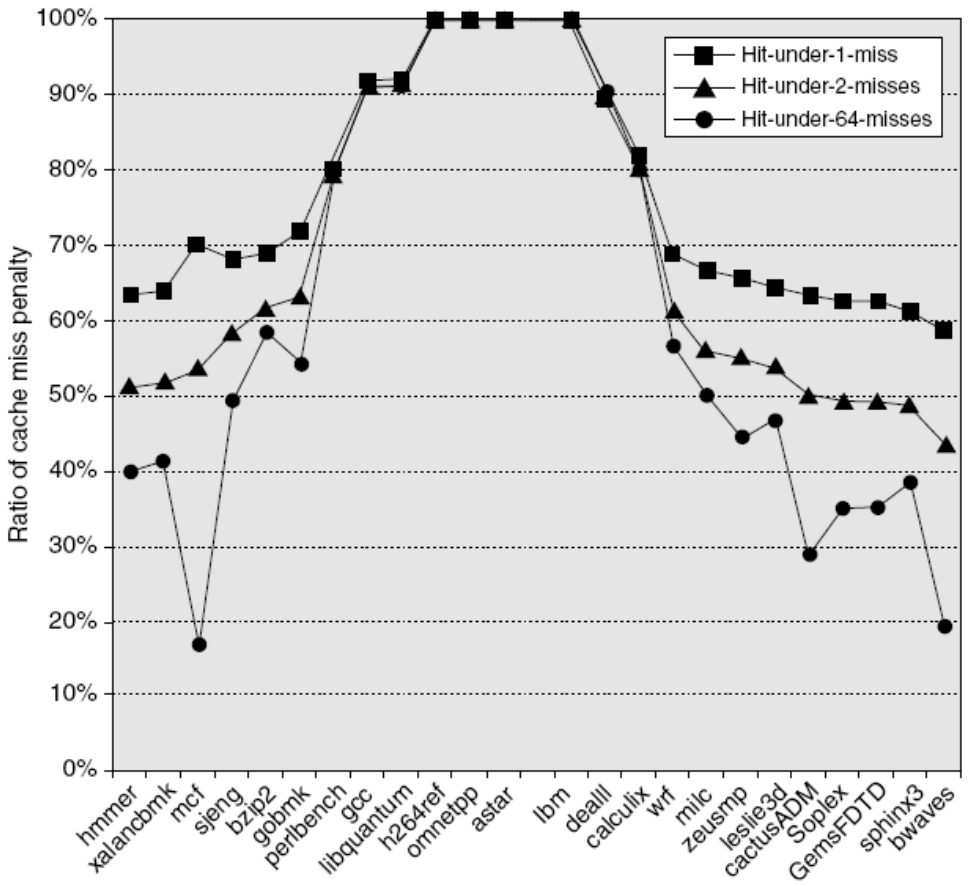
- Pipeline cache access to improve bandwidth
  - Examples:
    - Pentium: 1 cycle
    - Pentium Pro – Pentium III: 2 cycles
    - Pentium 4 – Core i7: 4 cycles
- Increases branch mis-prediction penalty
- Makes it easier to increase associativity

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.

4. Nonblocking Caches

- Allow hits before previous misses complete
  - "Hit under miss"
  - "Hit under multiple miss"
- L2 must support this
- In general, processors can hide L1 miss penalty but not L2 miss penalty



Copyright © 2012, Elsevier Inc. All rights reserved.

5. Multibanked Caches

- Organize cache as independent banks to support simultaneous access
  - ARM Cortex-A8 supports 1-4 banks for L2
  - Intel i7 supports 4 banks for L1 and 8 banks for L2

- Interleave banks according to block address

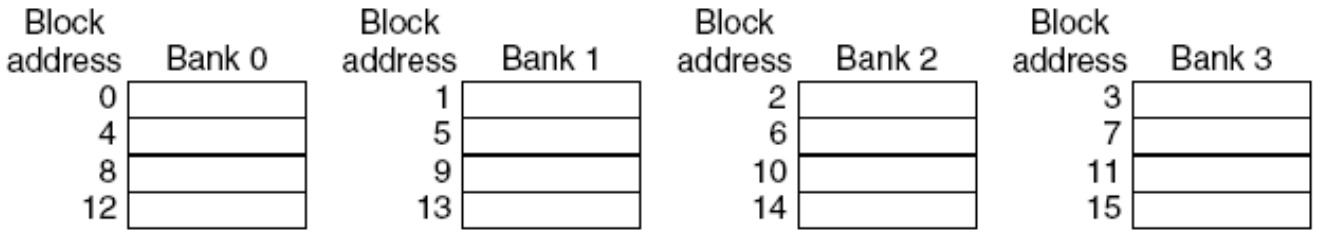


Figure 2.6 Four-way interleaved cache banks using block addressing. Assuming 64 bytes per blocks, each of these addresses would be multiplied by 64 to get byte addressing.

Copyright © 2012, Elsevier Inc. All rights reserved.

6. Critical Word First, Early Restart

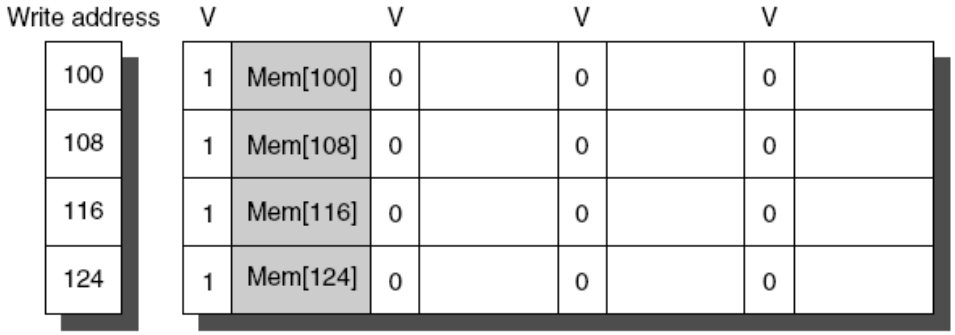
- Critical word first
  - Request missed word from memory first
  - Send it to the processor as soon as it arrives
- Early restart
  - Request words in normal order
  - Send missed work to the processor as soon as it arrives

- Effectiveness of these strategies depends on block size and likelihood of another access to the portion of the block that has not yet been fetched

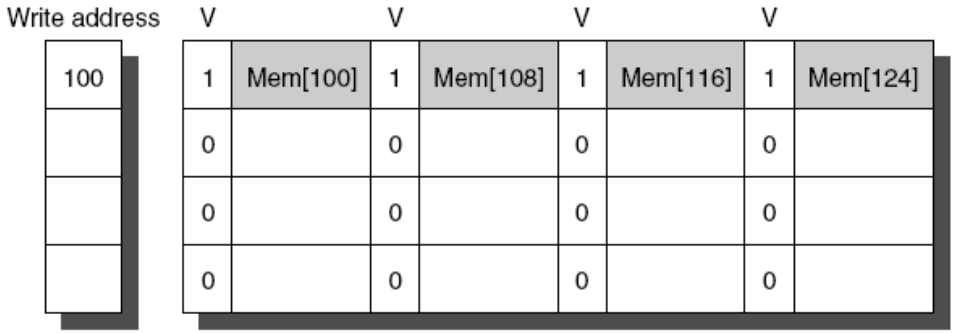
Copyright © 2012, Elsevier Inc. All rights reserved.

7. Merging Write Buffer

- When storing to a block that is already pending in the write buffer, update write buffer
- Reduces stalls due to full write buffer
- Do not apply to I/O addresses



No write buffering



Write buffering

MORGAN KAUFMANN
Copyright © 2012, Elsevier Inc. All rights reserved.

8. Compiler Optimizations

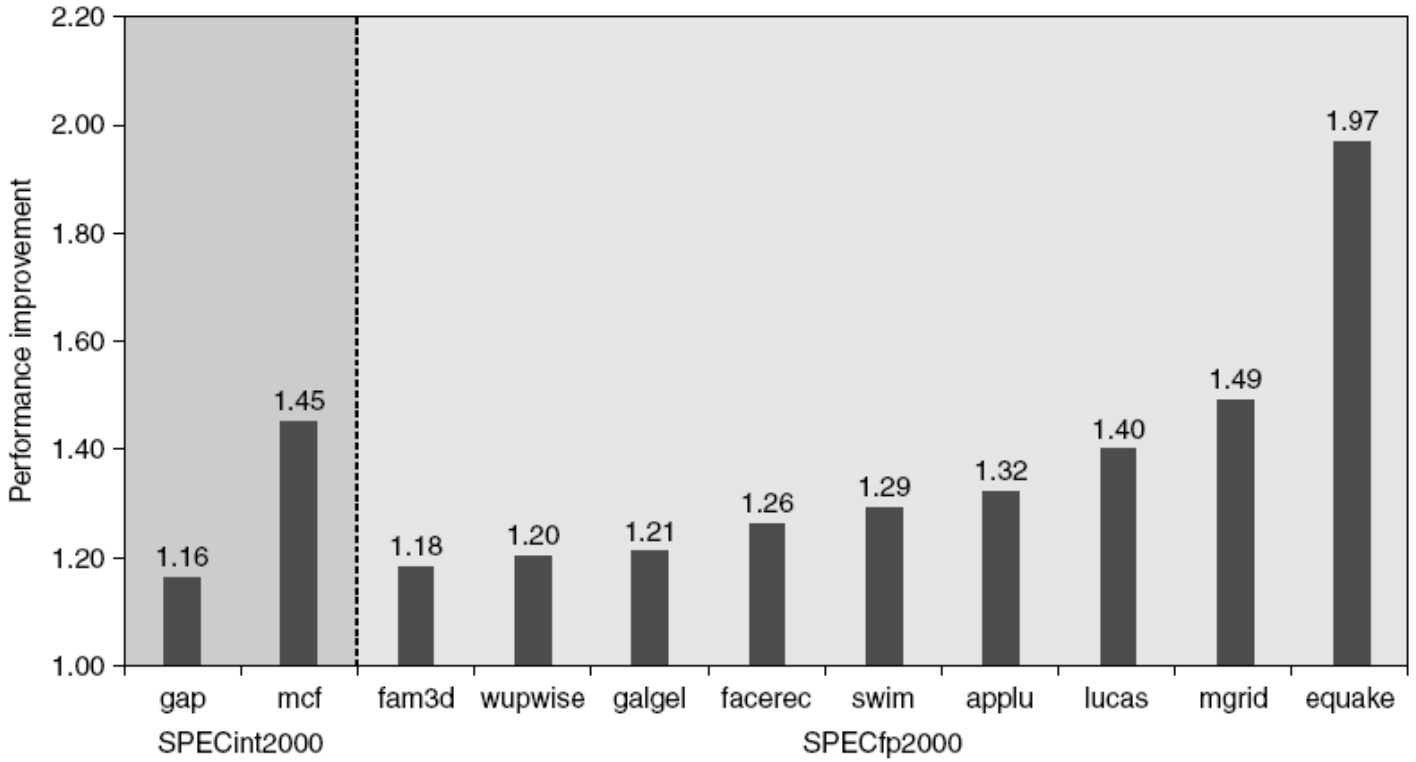
- Loop Interchange
  - Swap nested loops to access memory in sequential order
- Blocking
  - Instead of accessing entire rows or columns, subdivide matrices into blocks
  - Requires more memory accesses but improves locality of accesses

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.

9. Hardware Prefetching

- Fetch two blocks on miss (include next sequential block)



Pentium 4 Pre-fetching

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.

10. Compiler Prefetching

- Insert prefetch instructions before data is needed
- Non-faulting: prefetch doesn't cause exceptions

- Register prefetch
  - Loads data into register
- Cache prefetch
  - Loads data into cache

- Combine with loop unrolling and software pipelining

Copyright © 2012, Elsevier Inc. All rights reserved.

Summary

Technique	Hit time	Band-width	Miss penalty	Miss rate	Power consumption	Hardware cost/ complexity	Comment
Small and simple caches	+			–	+	0	Trivial; widely used
Way-predicting caches	+				+	1	Used in Pentium 4
Pipelined cache access	–	+				1	Widely used
Nonblocking caches		+	+			3	Widely used
Banked caches		+			+	1	Used in L2 of both i7 and Cortex-A8
Critical word first and early restart			+			2	Widely used
Merging write buffer			+			1	Widely used with write through
Compiler techniques to reduce cache misses				+		0	Software is a challenge, but many compilers handle common linear algebra calculations
Hardware prefetching of instructions and data			+	+	–	2 instr., 3 data	Most provide prefetch instructions; modern high-end processors also automatically prefetch in hardware.
Compiler-controlled prefetching			+	+		3	Needs nonblocking cache; possible instruction overhead; in many CPUs

MORGAN KAUFMANN

Copyright © 2012, Elsevier Inc. All rights reserved.