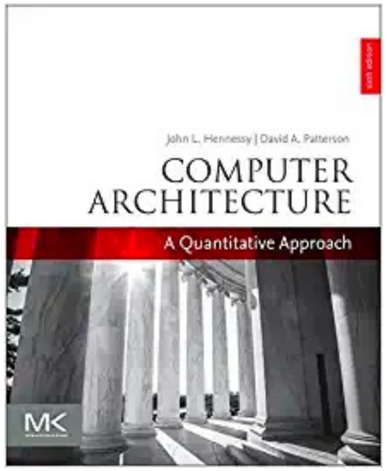


Chapter 2

Memory Hierarchy Design

Part 3: Case Study with POWER

"Ideally one would desire an indefinitely large memory capacity such that any particular … word would be immediately available. … We are … forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."
– A. W. Burks, H. H. Goldstine, and J. von Neumann, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument (1946)*

Acknowledgements

- Thanks to many sources for slide material

  © 1990 Morgan Kaufmann Publishers, © 2001-present Elsevier
  Computer Architecture: A Quantitative Approach by J. Hennessy & D. Patterson

  © 1994 Morgan Kaufmann Publishers, © 2001-present Elsevier
  Computer Organization and Design by D. Patterson & J. Hennessy

  © 2002 K. Asinovic & Arvind, MIT

  © 2002 J. Kubiatowicz, University of California at Berkeley

  © 2006, © 2010 No Starch Press for Inside the Machine by J. Stokes

  © 2007 W.-M. Hwu & D. Kirk, University of Illinois & NVIDIA

  © 2007-2010 J. Owens, University of California at Davis

  © 2010 CRC Press for Introduction to Concurrency in Programming Languages by M. Sottile, T. Mattson, and C. Rasmussen

  © 2017, IBM POWER9 Processor Architecture by Sadasivam et al., IBM

  © 2016, © 2019 POWER9 Processor User's Manual, IBM

  © The OpenPOWER Foundation

  © 2022, W. Feng, Virginia Tech

Recall: Basic Cache Optimizations

Review Appendix B, as needed.

- 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
- Higher number of cache levels
  - Reduces overall memory access time, increases complexity
- Giving priority to read misses over writes
  - Reduces miss penalty, increases complexity
- Avoiding address translation in cache indexing
  - Reduces hit time

Recall: Advanced Optimizations for Caching

Key

$ = cache

- Reduce Hit Time
  - (1) Small & simple first-level $ and (2) way prediction
    - Side effect: Reduce power consumption
- Increase Cache Bandwidth
  - (3) Pipelined $, (4) non-blocking $, and (5) multi-banked $
    - Side effect: Varying impacts on power consumption
- Reduce Miss Penalty
  - (6) Critical word first and (7) merging write buffers
    - Side effect: Little impact on power
- Reduce Miss Rate
  - (8) Compiler optimizations. Side effect: Reduces power consumption
- Reduce Miss Penalty or Miss Rate via Parallelism
  - (9) Hardware pre-fetching and (10) compiler pre-fetching

Recall: Avg. Memory Access Time

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

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

- How to reduce the *average memory access time?*
  - Reduce hit time
  - Reduce miss rate
  - Reduce miss penalty

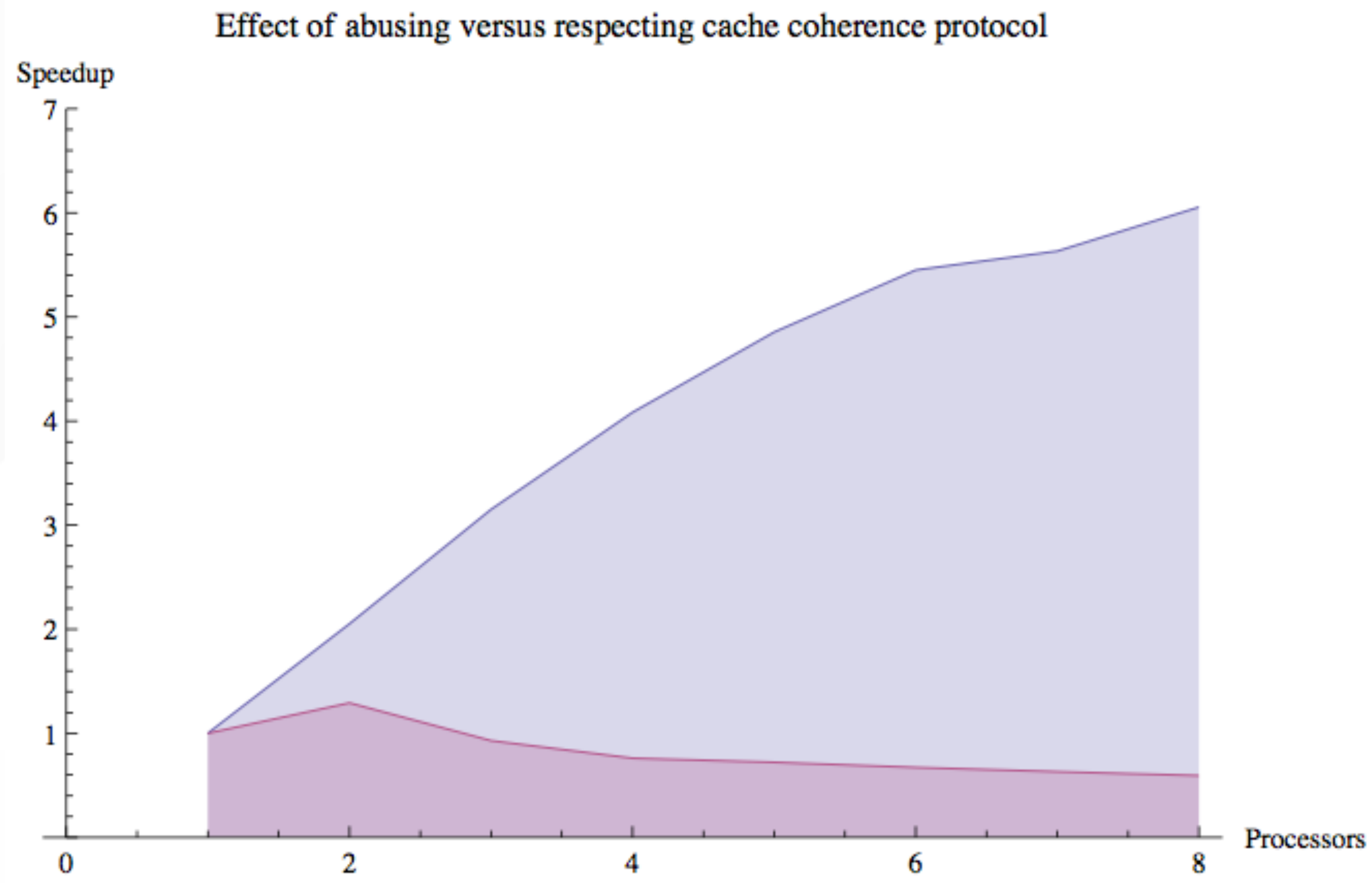
Recall: Cache Coherence & Performance

Summary

- Unlike details with pipelining (e.g., ILP) that only concern compiler writers, you the programmer need to acknowledge that cache coherence is going on “under the covers.” Why?

The coherence protocol can DRAMATICALLY impact your performance!

Impact of Cache Coherence in Multicore CPUs



Case Study: POWER Architecture

- POWER8: 64-bit implementation of the POWER ISA
  - Compute
    - Capable of up to eight-way simultaneous multithreading (SMT)
    - Enhanced branch prediction, which uses local & global prediction tables with a selector table to choose the preferred predictor
  - Memory
    - Block Size: 128 B
    - L1 Instruction $: 32 kB, eight-way set-associative
    - L1 Data $: 64 kB, eight-way set-associative
    - L2 $ per Core/Chip: 512 kB / 6 MB
    - L3 $ per Core/Chip: 8 MB / 96 MB
    - L4 $ per Chip: 128 MB
    - Enhanced prefetch with instruction speculation awareness and data prefetch depth awareness

Case Study: POWER Architecture

	POWER8	POWER9	POWER10	Replacement?
Block Size	128 B	128 B	128 B	-
L1 Instr. $	32 kB (8-way)	32 kB (8-way)	48 kB	
L1 Data $	64 kB (8-way)	32 kB (8-way)	32 kB	Pseudo-LRU
L2 $ Core/Chip	512 kB / 6 MB	512 kB (8-way)	2 MB	
L3 $ "Core"/Chip	8 MB / 96 MB	10 MB (20-way)	-/120 MB	
L4 $ Chip	128 MB	128 MB		

Note: POWER10 read/write bandwidth of memory and L3/L2/L1 cache is ***double*** that of POWER9.

Sources: IBM and 7-cpu.com

Case Study: POWER Architecture

- POWER8 Additional Details
  - 62 mm^2 (for 6 cores/chip version), 22 nm, 15 layers, Cu, SOI,
  - L1 Instruction $ = 32 KB, 8-WAY
  - L1 Data $ = 64 KB, 128 B/line, 8-WAY
  - L2 cache = 512 KB per core, 128 B/line, 8-WAY
  - L3 local cache (Fast-L3 Region cache) = 8 MB (eDRAM), 128 B/line, 8-WAY
  - L3 cache = (8 MB * 5) per chip (eDRAM) consist of LOCAL-L3 from another cores, 128 B/line,
  - L4: Off chip: 16 MB memory buffer chip per channel, 8 chips per socket.

Case Study: POWER Architecture

- POWER9 Additional Details
  - 695 mm^2 (for 24 cores chip version), 14 nm, 17 layers, Cu, SOI finFET.
  - L1 Instruction \$ = 32 KB, (4x 32-byte sectors), 8-WAY, effective-address index, real-address tags.
  - L1 Data \$ = 32 KB, 128 B/line (2x 64-byte sectors), 8-WAY. 8 banks. Store-through (to L2 \$) policy; no allocate on store misses. Pseudo-LRU. 64-byte reload interface from the L2 \$ can supply 64 bytes in every processor clock. Effective address index, real address tags.
  - L2 \$ = 512 KB per core, 128 B/line, 8-WAY. 2-bank. 1 processor read port, 2 snoop read ports, and 1 write port per physical bank
  - L3 local cache (Fast-L3 Region cache) = 10 MB (eDRAM) per 2 cores, 128 B/line, 20-WAY
  - L3 \$ = (10 MB * 12) per chip (eDRAM) consist of LOCAL-L3 from another cores, 128 B/line (2x 64-byte sector), victim cache for L2 cache, and victim cache for other on-chip L3 caches. 64-byte wide data bus to L2 for reads.