CS 110
Computer Architecture
Lecture 16:
Caches Part 3

Instructor:
Sören Schwertfeger

http://shtech.org/courses/ca/

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

Caches Review

- Direct-Mapped vs. Set-Associative vs. Fully Associative
- AMAT = Hit Time + Miss Rate * Miss Penalty
- 3 Cs of cache misses: Compulsory, Capacity, Conflict
- Effect of cache parameters on performance

Primary Cache Parameters

- Block size (aka line size)
  - how many bytes of data in each cache entry?
- Associativity
  - how many ways in each set?
  - Direct-mapped => Associativity = 1
  - Set-associative => 1 < Associativity < #Entries
  - Fully associative => Associativity = #Entries
- Capacity (bytes) = Total #Entries * Block size
- #Entries = #Sets * Associativity

Other Cache Parameters

- Write Policy
- Replacement policy

Write Policy Choices

- Cache hit:
  - **write through**: writes both cache & memory on every access
    - Generally higher memory traffic but simpler pipeline & cache design
  - **write back**: writes cache only, memory `written only when dirty entry evicted
    - A dirty bit per line reduces write-back traffic
    - Must handle 0, 1, or 2 accesses to memory for each load/store
- Cache miss:
  - **no write allocate**: only write to main memory
  - **write allocate** (aka fetch on write): fetch into cache
- Common combinations:
  - write through and no write allocate
  - write back with write allocate

Replacement Policy

In an associative cache, which line from a set should be evicted when the set becomes full?

- Random
- Least-Recently Used (LRU)
  - LRU cache state must be updated on every access
  - True implementation only feasible for small sets (2-way)
  - Pseudo-LRU binary tree often used for 4-8 way
- First-In, First-Out (FIFO) a.k.a. Round-Robin
  - Used in highly associative caches
- Not-Most-Recently Used (NMRU)
  - FIFO with exception for most-recently used line or lines

This is a second-order effect. Why?

Replacement only happens on misses

Sources of Cache Misses (3 C's)

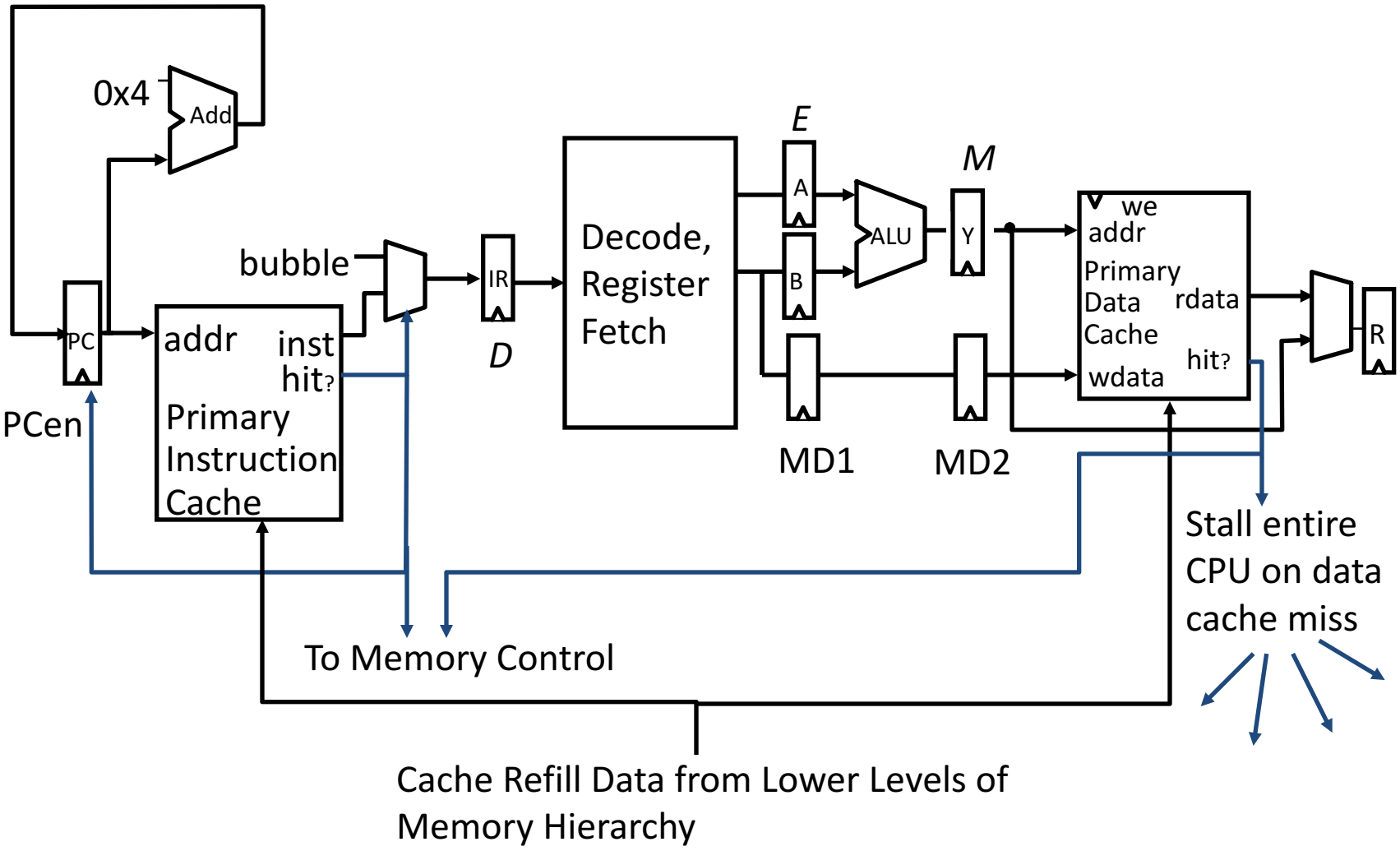
- *Compulsory* (cold start, first reference):
  - 1st access to a block, "cold" fact of life, not a lot you can do about it.
    - If running billions of instructions, compulsory misses are insignificant
- *Capacity*:
  - Cache cannot contain all blocks accessed by the program
    - Misses that would not occur with infinite cache
- *Conflict* (collision):
  - Multiple memory locations mapped to same cache set
    - Misses that would not occur with ideal fully associative cache

Impact of Cache Parameters on Performance

- AMAT = Hit Time + Miss Rate * Miss Penalty
  - Note, we assume always first search cache, so must charge hit time for both hits and misses!
- For misses, characterize by 3Cs

CPU-Cache Interaction

(5-stage pipeline)



Improving Cache Performance

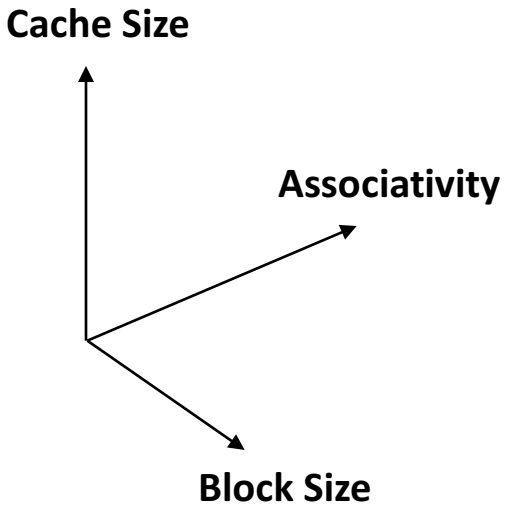
AMAT = Time for a hit + Miss rate x Miss penalty

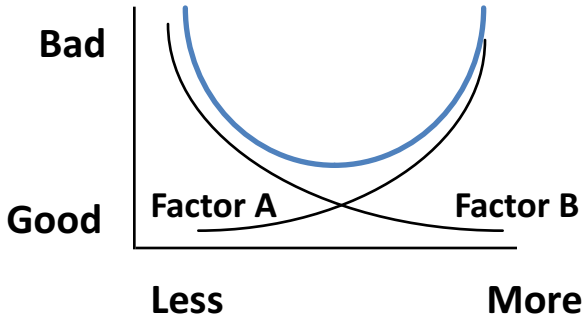
- Reduce the time to hit in the cache
  - E.g., Smaller cache
- Reduce the miss rate
  - E.g., Bigger cache
- Reduce the miss penalty
  - E.g., Use multiple cache levels

Cache Design Space

Computer architects expend considerable effort optimizing organization of cache hierarchy – big impact on performance and power!

- Several interacting dimensions
  - Cache size
  - Block size
  - Associativity
  - Replacement policy
  - Write-through vs. write-back
  - Write allocation
- Optimal choice is a compromise
  - Depends on access characteristics
    - Workload
    - Use (I-cache, D-cache)
  - Depends on technology / cost
- Simplicity often wins





Increasing Associativity?

- Hit time as associativity increases?
  - Increases, with large step from direct-mapped to >=2 ways, as now need to mux correct way to processor
  - Smaller increases in hit time for further increases in associativity
- Miss rate as associativity increases?
  - Goes down due to reduced conflict misses, but most gain is from 1->2->4-way with limited benefit from higher associativities
- Miss penalty as associativity increases?
  - Unchanged, replacement policy runs in parallel with fetching missing line from memory

Increasing #Entries?

- Hit time as #entries increases?
  - Increases, since reading tags and data from larger memory structures
- Miss rate as #entries increases?
  - Goes down due to reduced capacity and conflict misses
  - *Architects rule of thumb: miss rate drops ~2x for every ~4x increase in capacity (only a gross approximation)*
- Miss penalty as #entries increases?
  - Unchanged

At some point, increase in hit time for a larger cache may overcome the improvement in hit rate, yielding a decrease in performance

Questions

- For fixed total cache capacity and associativity, what is effect of larger blocks on each component of AMAT:
  - A: Decrease, B: Unchanged, C: Increase
- Hit Time?
- Miss Rate?
- Miss Penalty?

Questions

- For fixed total cache capacity and associativity, what is effect of larger blocks on each type of miss rate:
  - A: Decrease, B: Unchanged, C: Increase
- Compulsory?
- Capacity?
- Conflict?

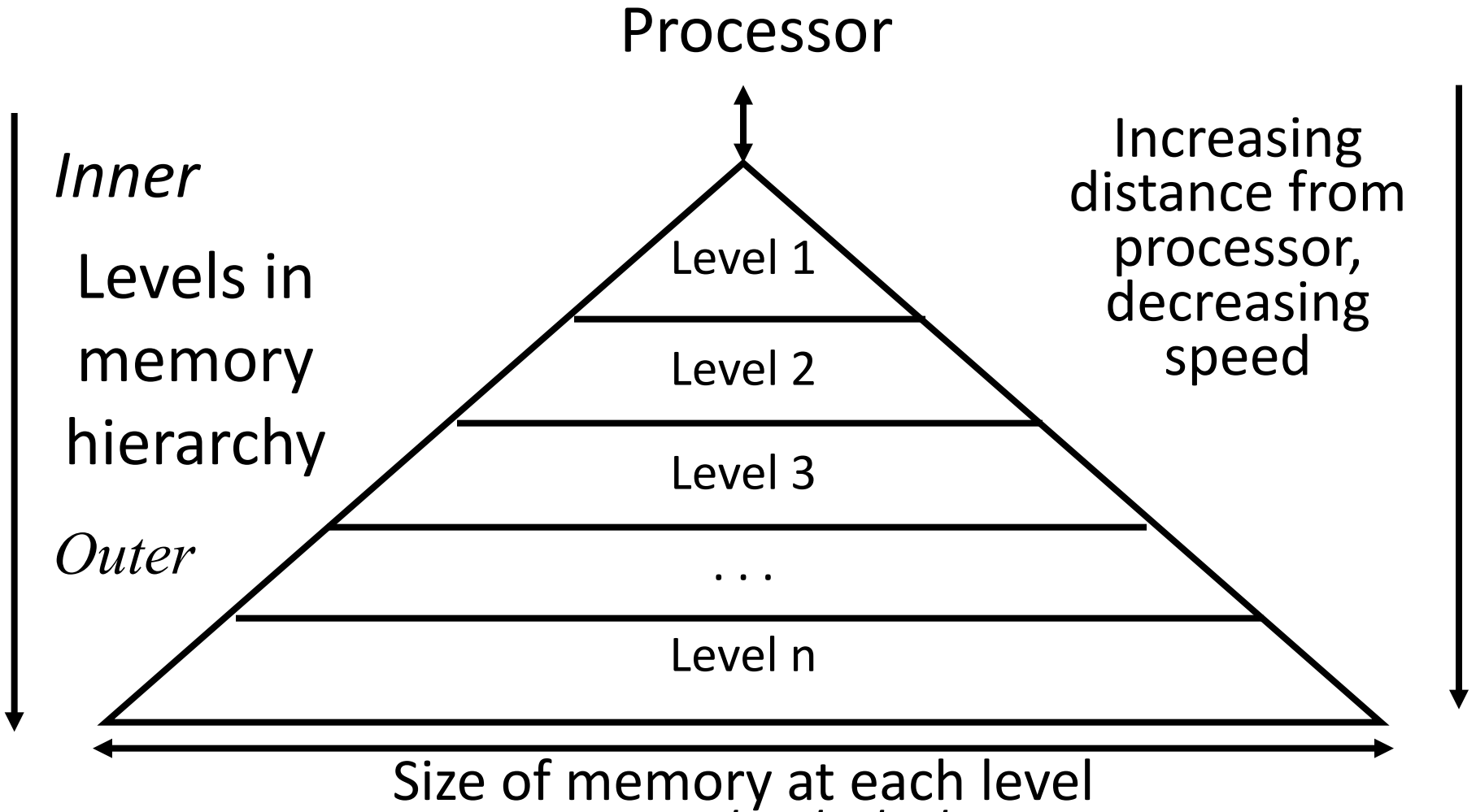
Increasing Block Size?

- Hit time as block size increases?
  - Hit time unchanged, but might be slight hit-time reduction as number of tags is reduced, so faster to access memory holding tags
- Miss rate as block size increases?
  - Goes down at first due to spatial locality, then increases due to increased conflict misses due to fewer blocks in cache
- Miss penalty as block size increases?
  - Rises with longer block size, but with fixed constant initial latency that is amortized over whole block

How to Reduce Miss Penalty?

- Could there be locality on misses from a cache?
- Use multiple cache levels!
- With Moore's Law, more room on die for bigger L1 caches and for second-level (L2) cache
- And in some cases even an L3 cache!
- IBM mainframes have ~1GB L4 cache off-chip.

Review: Memory Hierarchy

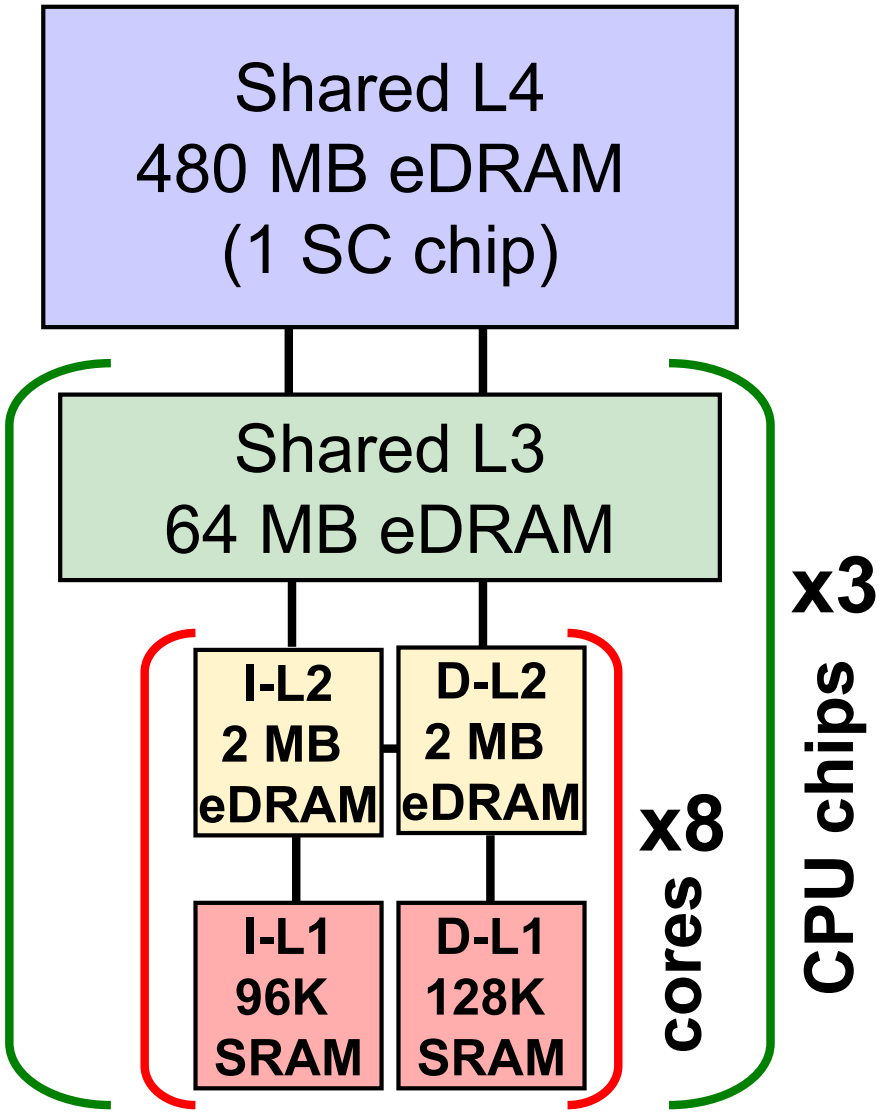


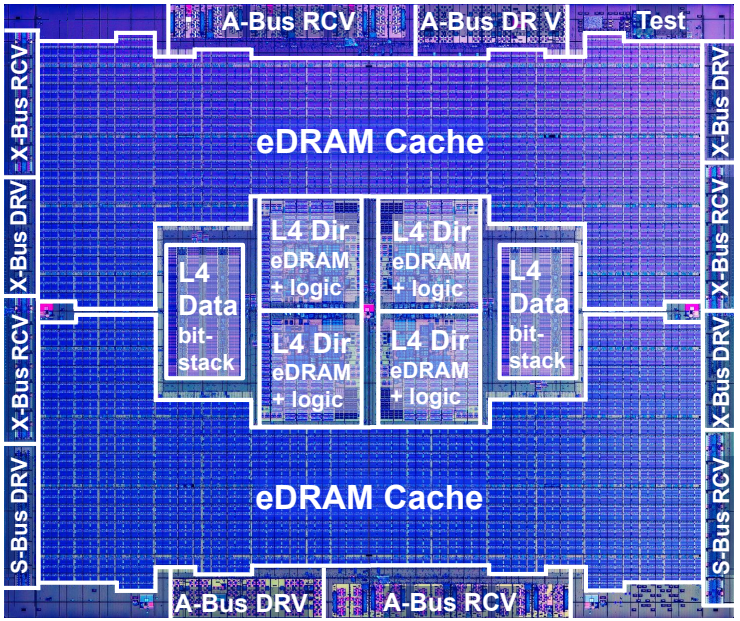
As we move to outer levels the latency goes up and price per bit goes down.

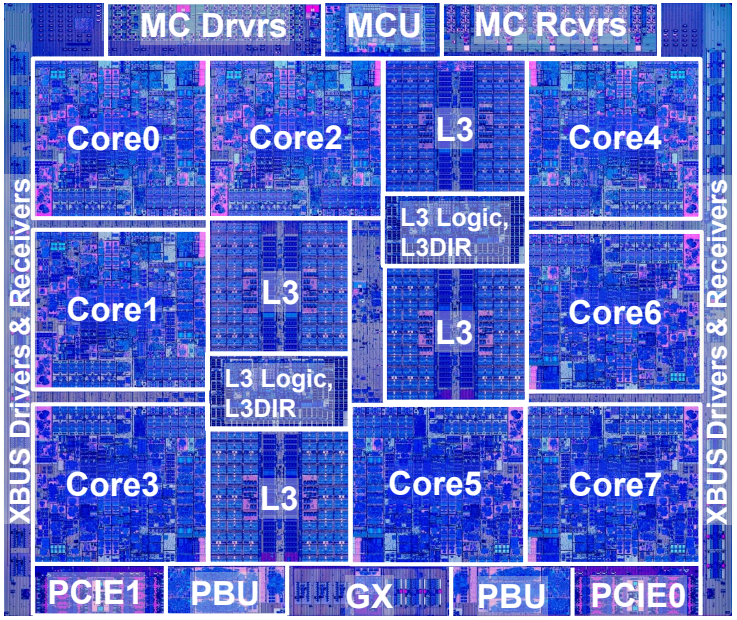
2015 IBM CPU

- At ISSCC 2015 in San Francisco IBM mainframe chip details
- z13 designed in 22nm SOI technology with seventeen metal layers, 4 billion transistors/chip
- 8 cores/chip, with 2MB L2 cache, 64MB L3 cache, and 480MB L4 off-chip cache.
- 5GHz clock rate, 6 instructions per cycle, 2 threads/core
- Up to 24 processor chips in shared memory node

IBM z13 Memory Hierarchy

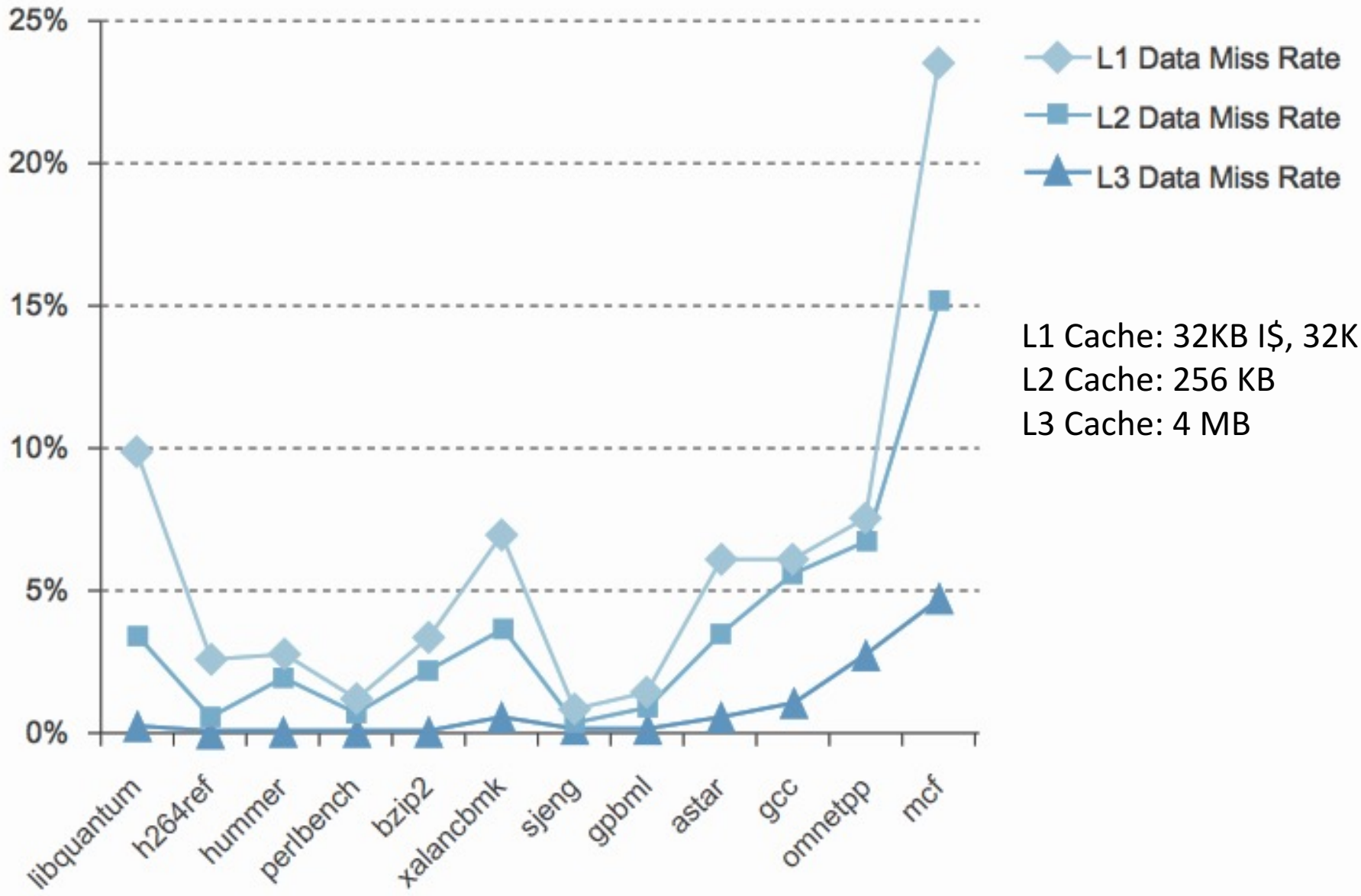






Local vs. Global Miss Rates

- *Local miss rate* – the fraction of references to one level of a cache that miss
- Local Miss rate L2$ = L2$ Misses / L1$ Misses
  = L2$ Misses / total_L2_accesses
- *Global miss rate* – the fraction of references that miss in all levels of a multilevel cache
  - L2$ local miss rate >> than the global miss rate



L1 Cache: 32KB I$, 32KB D$
L2 Cache: 256 KB
L3 Cache: 4 MB

FIGURE 5.47 The L1, L2, and L3 data cache miss rates for the Intel Core i7 920 running the full integer SPECCPU2006 benchmarks.

Local vs. Global Miss Rates

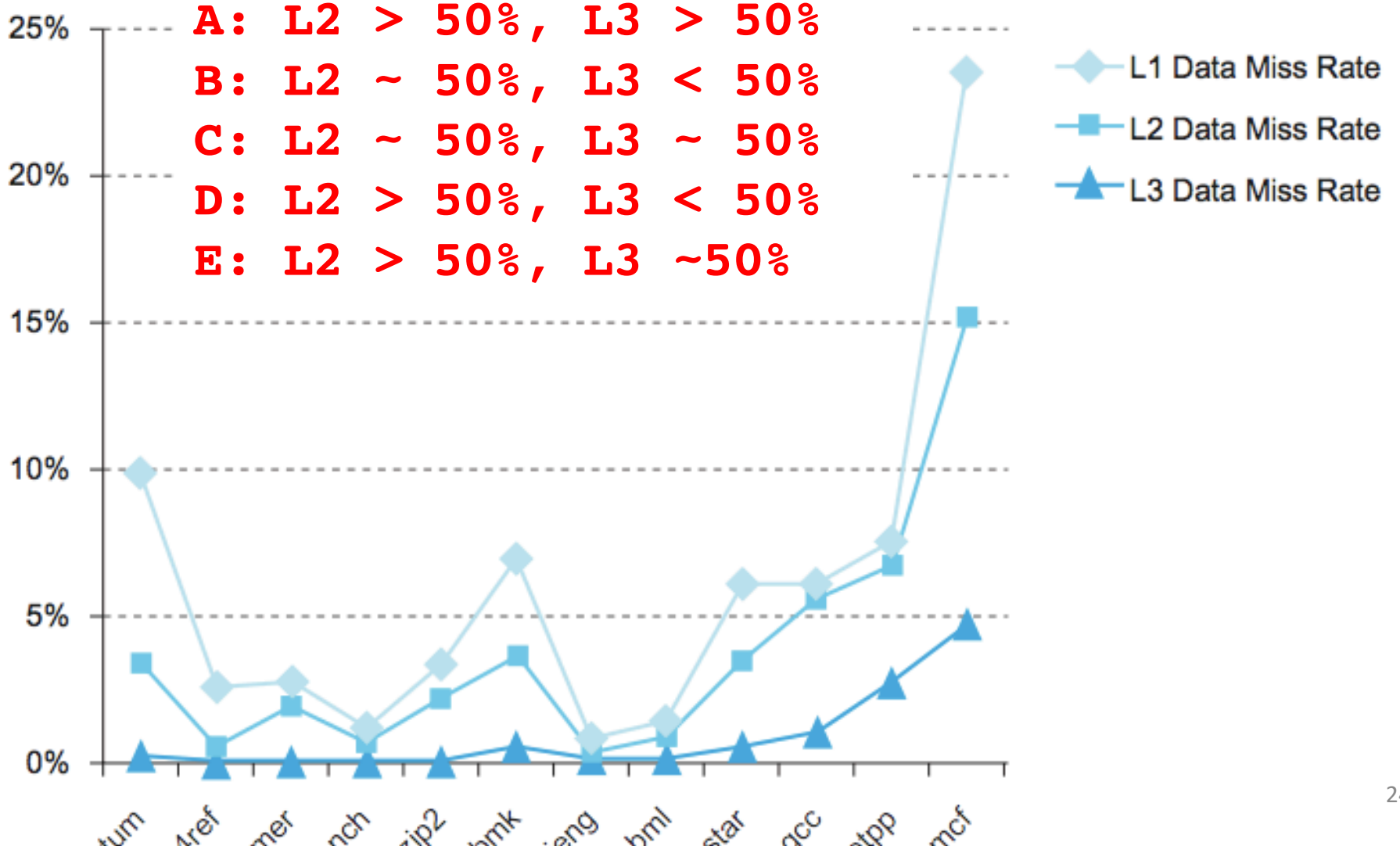
- *Local miss rate* – the fraction of references to one level of a cache that miss
- Local Miss rate L2$ = $L2 Misses / L1$ Misses
- *Global miss rate* – the fraction of references that miss in all levels of a multilevel cache
  - L2$ local miss rate >> than the global miss rate
- Global Miss rate = L2$ Misses / Total Accesses
  = (L2$ Misses / L1$ Misses) × (L1$ Misses / Total Accesses)
  = Local Miss rate L2$ × Local Miss rate L1$

- AMAT = Time for a hit + Miss rate × Miss penalty
- AMAT = Time for a L1$ hit + (local) Miss rate L1$ × (Time for a L2$ hit + (local) Miss rate L2$ × L2$ Miss penalty)

Question

- Overall, what are L2 and L3 local miss rates?

A: L2 > 50%, L3 > 50%
B: L2 ~ 50%, L3 < 50%
C: L2 ~ 50%, L3 ~ 50%
D: L2 > 50%, L3 < 50%
E: L2 > 50%, L3 ~50%



Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KB each for instructions/data per core	64 KB each for instructions/data per core
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L1 hit time (load-use)	Not Available	3 clock cycles
L2 cache organization	Unified (instruction and data) per core	Unified (instruction and data) per core
L2 cache size	256 KB (0.25 MB)	512 KB (0.5 MB)
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	Not Available	9 clock cycles
L3 cache organization	Unified (instruction and data)	Unified (instruction and data)
L3 cache size	8192 KB (8 MB), shared	2048 KB (2 MB), shared
L3 block size	64 bytes	64 bytes
L3 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L3 hit time	Not Available	38 (?)clock cycles

CPI/Miss Rates/DRAM Access SpecInt2006

Name	CPI	Data Only L1 D cache misses/1000 instr	Data Only L2 D cache misses/1000 instr	Instructions and Data DRAM accesses/1000 instr
perl	0.75	3.5	1.1	1.3
bzip2	0.85	11.0	5.8	2.5
gcc	1.72	24.3	13.4	14.8
mcf	10.00	106.8	88.0	88.5
go	1.09	4.5	1.4	1.7
hmmer	0.80	4.4	2.5	0.6
sjeng	0.96	1.9	0.6	0.8
libquantum	1.61	33.0	33.1	47.7
h264avc	0.80	8.8	1.6	0.2
omnetpp	2.94	30.9	27.7	29.8
astar	1.79	16.3	9.2	8.2
xalancbmk	2.70	38.0	15.8	11.4
Median	1.35	13.6	7.5	5.4

In Conclusion, Cache Design Space

- Several interacting dimensions
  - Cache size
  - Block size
  - Associativity
  - Replacement policy
  - Write-through vs. write-back
  - Write-allocation
- Optimal choice is a compromise
  - Depends on access characteristics
    - Workload
    - Use (I-cache, D-cache)
  - Depends on technology / cost
- Simplicity often wins