ECSE 425 Lecture 22: Cache Optimizations

H&P Appendix C

Last Time

- Two Questions
  - Q3: Block replacement
  - Q4: Write strategy
- Cache Performance
  - Miss rate
  - Average memory access time
  - CPU time

Today

- Basic Cache Optimizations
  - Reducing the miss rate
  - Reducing the miss penalty
  - Reducing the time to hit in the cache

Cache Optimizations

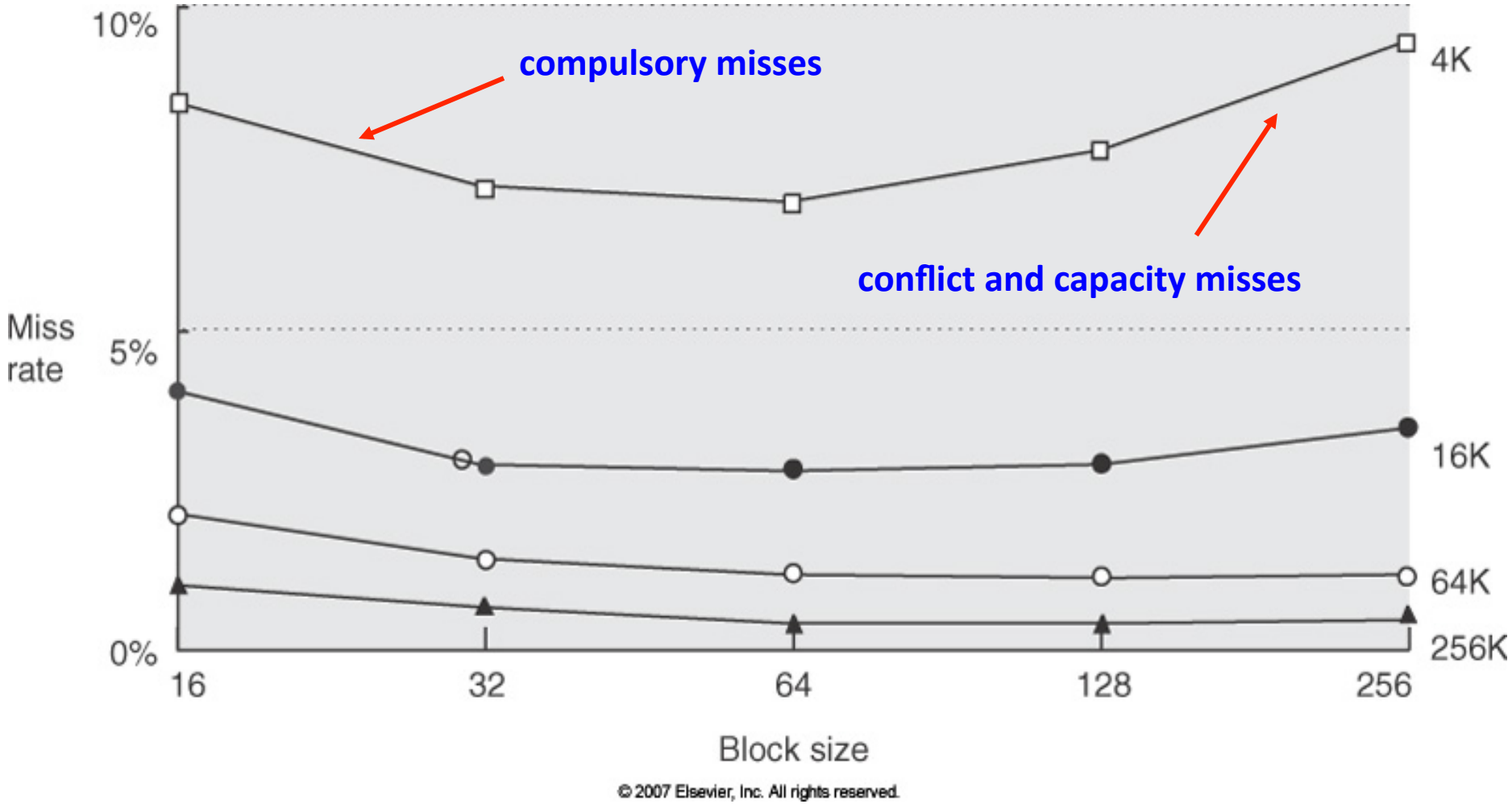
- AMAT = HitTime + MissRate × MissPenalty
  - Improve performance by reducing the components
- Miss rate reduction:
  - larger block size,
  - larger cache size,
  - higher associativity,
- Miss penalty reduction:
  - multilevel caches,
  - prioritize reads over writes,
- Hit time reduction:
  - avoiding address translation when indexing the cache

The Three Cs (of Cache Misses)

- Compulsory: first access to a block
  - First access always misses, regardless of organization
- Capacity: cache cannot contain all needed blocks
  - A block is retrieved, replaced, and retrieved again
  - Misses that would occur in a fully-associate cache
- Conflict: set can't contain all needed blocks
  - Some FA cache hits are misses in an *n*-way SA cache if more than *n* requested blocks map to the same set

Miss rate reduction #1: Larger Block Sizes

Larger blocks exploit spatial locality (but might increase MP)



Block size example

- Compare two caches with different block sizes
- Main memory
  - 80 cycle overhead
  - 16 B every 2 cycles (16 B takes 82, 32 B takes 84)
- 4 KB, 16 B blocks
  - 8.57% miss rate
- 256 KB, 256 B blocks
  - 0.49% miss rate
- What is the AMAT for each?

Miss rate reduction #2: Larger Caches

- Higher cost, higher power, longer hit time
- But miss rate doesn't fall linearly with size
  - If a cache can be enlarged to contain the working set, performance jumps significantly!

Miss rate reduction #3: Associativity

- 2-way set associative cache of size N
  - Same miss rate as a direct mapped cache of size 2N
  - Holds for cache sizes less than 128KB
- 8-way SA is as practically equivalent to FA
  - With respect to reducing cache misses
  - Holds for cache sizes up to 1024KB
- As associativity increases, hit times increases
  - Due to increased search complexity

Multilevel Caches

- Miss penalty reduction #1
- Reduce penalty by adding a cache to the cache!
- Penalty at a given level is determined by the AMAT of the next lower level

$$\text{AMAT}_i = \text{HitTime}_i + \text{MissRate}_i \times \text{MissPenalty}_i$$

$$\text{MissPenalty}_i = \text{AMAT}_{i+1}$$

SO

$$\text{AMAT}_{L1} = \text{HitTime}_{L1} + \text{MissRate}_{L1} \times (\text{HitTime}_{L2} + \text{MissRate}_{L2} \times \text{MissPenalty}_{L2})$$

$$= \underbrace{\text{HitTime}_{L1} + \text{MissRate}_{L1} \times \text{HitTime}_{L2}}_{\text{New hit time}} + \underbrace{\text{MissRate}_{L1} \times \text{MissRate}_{L2}}_{\text{New miss rate}} \times \text{MissPenalty}_{L2}$$

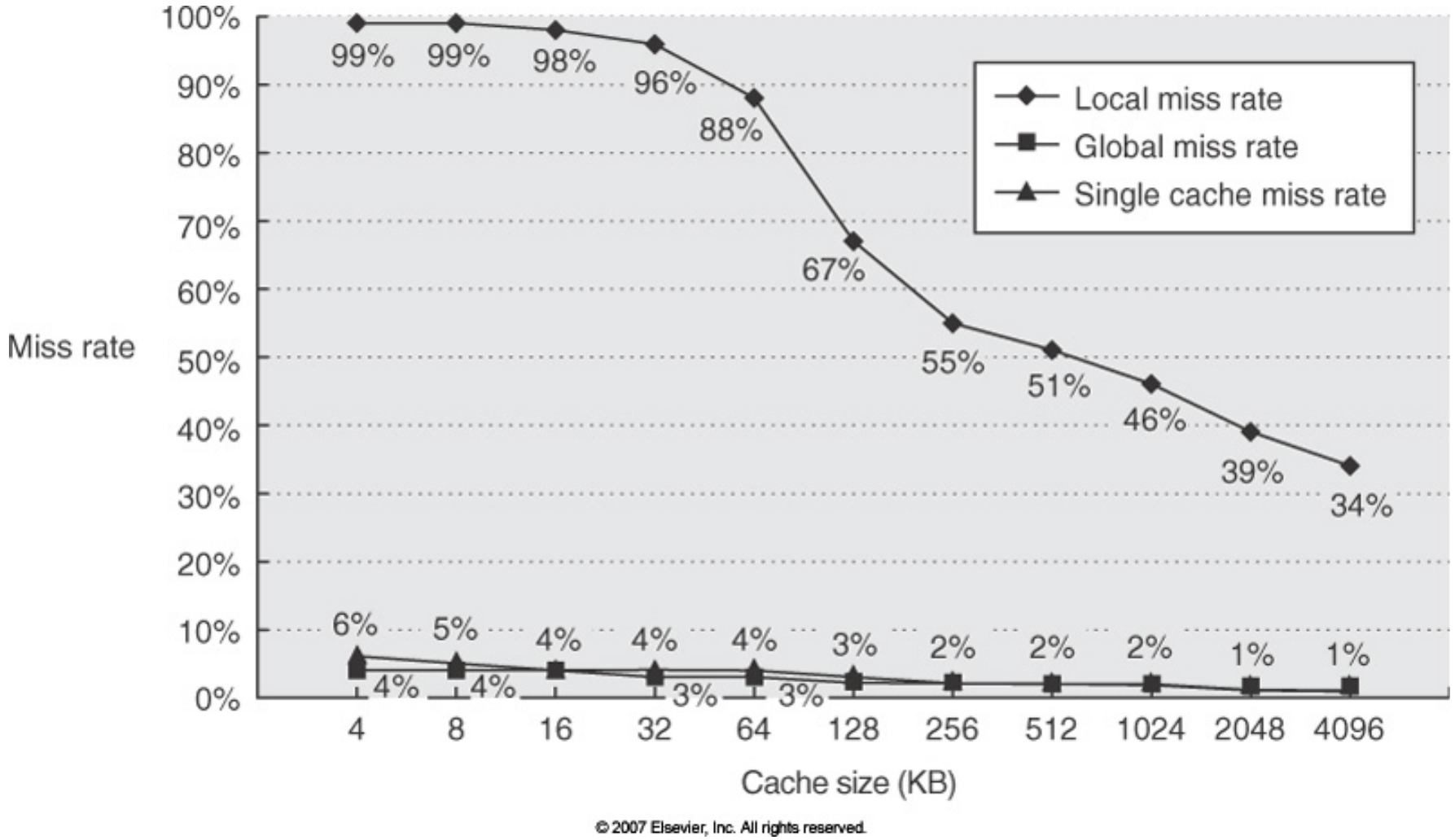
Average memory stalls per instruction = Misses per instruction $_{L1}$ × Hit time $_{L2}$ + Misses per instruction $_{L2}$ × Miss penalty $_{L2}$

Miss Rates in Multilevel Caches

- Local and global miss rates
  - Local mr1, mr2: misses per access to a cache
    - Small for L1, high for L2 (relatively fewer accesses)
  - Global mr1 x mr2: misses per access

Miss Rates in Multilevel Caches, Cont'd

- Split L1, 64KB (2-way SA with LRU) with L2 vs. single level cache



Typical Multilevel Caches

- L1 should be fast for a small hit time
  - Speed of L1 cache influences cycle time
  - L1 is usually small
  - L1 is usually split
- L2 should be bigger
  - Reduce miss rate with sophisticated organization
  - Speed of L2 cache only affects the miss penalty
  - L2 is usually unified
- L3 should be even bigger
  - On-chip, shared among processors

Example: Multi-level Caching

- L1: 40 misses per 1000 instructions
  - 1 cycle hit time
- L2: 20 misses per 1000 instructions
  - 10 cycle hit time
- Main memory: 200 cycle miss penalty
- 1.5 memory accesses per instruction
- AMAT? Average stall cycles per instruction?

Example: L2 Cache Associativity

- What's the L1 miss penalty under two different L2 configurations?
- Direct mapped L2:
  - Hit time = 10 CC
  - MR = 25%
- 2-way associative L2:
  - Hit time = 10.1 CC
  - MR = 20%
- Miss penalty MP = 200 CC

Multilevel Exclusion

- What if a design cannot afford a L2 cache that is much larger than L1
  - Wastes most of L2 with copies L1 data?
- In this case, use multilevel exclusion
  - L1 data is never found in L2 cache
  - L1 miss results in a swap instead of replacement
- AMD Opteron (2x64KB L1 and 1 MB L2)
- Also important for many-core designs with shared L2 (aggregate L1 > shared L2—exclusion!)

Prioritize Reads Over Writes

- Miss penalty reduction #2
- Serve reads before writes have been completed
- Write buffers complicate this
  - Used with write-through caches
  - Cause RAW hazards

Write Buffer RAW

- Assume a direct-mapped cache
  - Suppose addresses 512 and 1024 map to same set

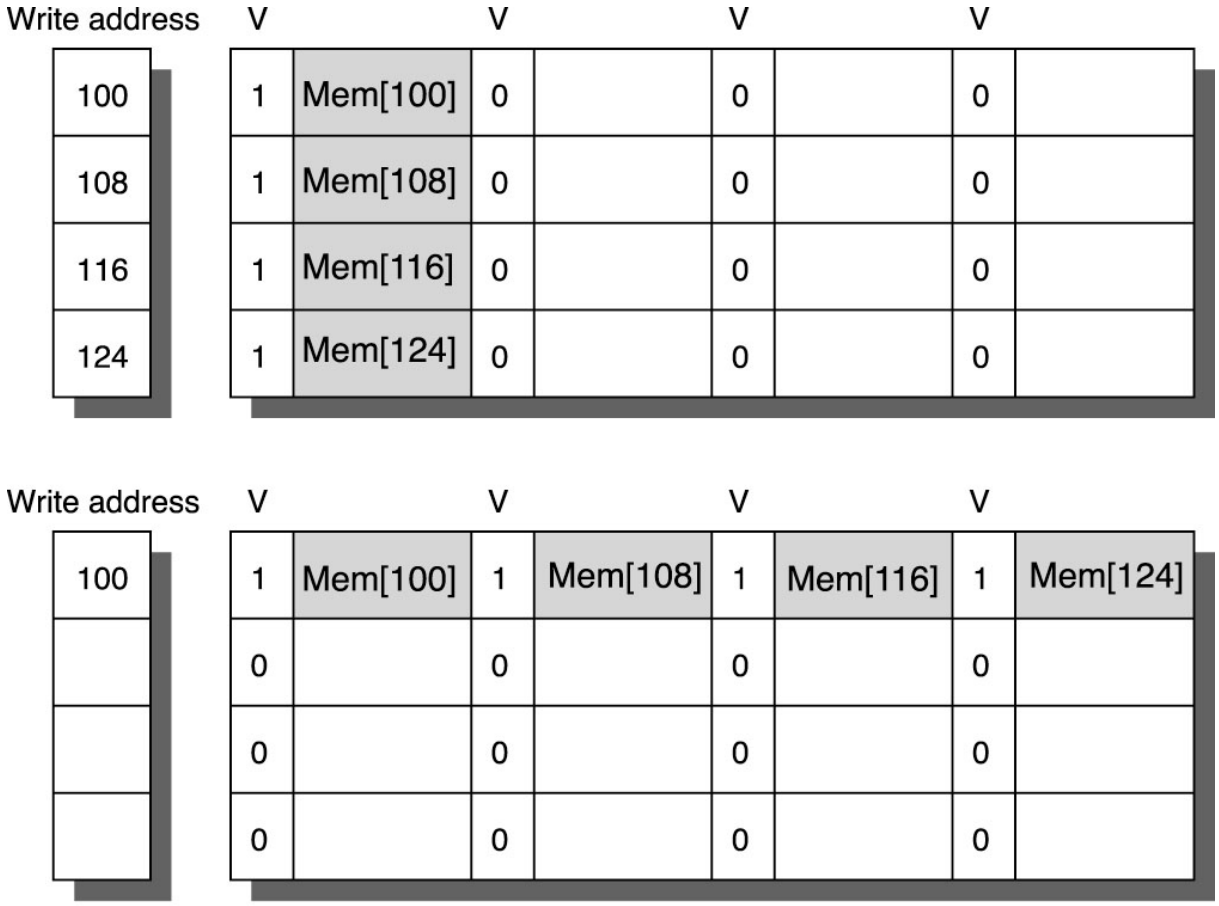
```
SW R3, 512(R0)      ; R3 in write buffer
LW R1, 1024(R0)     ; miss, replace block
LW R2, 512(R0)      ; miss
```

  - If write buffer has not completed store, wrong value written to cache block and R2
- Solution: on read miss, check write buffer
  - If there's no conflict, and if memory system is available, let read miss continue
  - Otherwise (a) wait, or (b) get the value from the buffer

Merging Write Buffer

- When writing to the write buffer, check if address matches an existing write buffer entry
  - Combine data with that write – write merging
  - Multiword writes are faster than one word at a time

Merging Write Buffer



Hit Time Reduction

- Avoid address translation during cache indexing
- We will come back to this after we study virtual memory…

Summary

- Reduce the miss rate
  - Increase the block size
  - Increase the cache size
  - Increase associativity
- Reduce the miss penalty
  - Multilevel caching
  - Prioritize reads over writes
- Reduce the hit time
  - Avoid address translation during cache indexing

Next Time

- Virtual memory
  - Read Appendix C.4!