Understanding Cache Coherence Misses

True Sharing Misses

Arise from actual communication of data through cache coherence

- First write by a processor to a shared block causes invalidation
- When another processor reads a modified word, a miss occurs

False Sharing Misses

Arise from invalidation-based coherence with single valid bit per block

- Occurs when a block is invalidated because some word in the block, other than the one being read, is written
- Would not occur if block size were a single word

These two types of coherence misses combine with the traditional "three C's" (compulsory, capacity, and conflict) to determine overall multiprocessor cache performance.

Example: Identifying Sharing Misses

- Assume words x1 and x2 are in the same cache block, which is in the shared state in the caches of both P1 and P2.

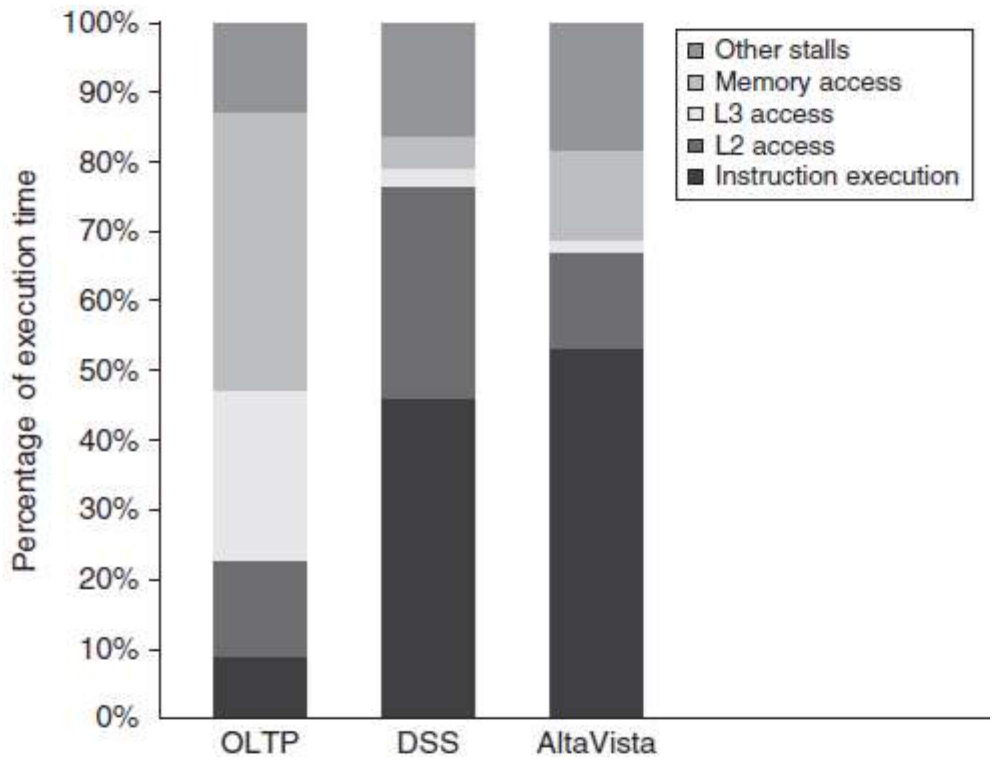
Time	P1	P2	Classification
1	Write x1		True sharing miss (x1 was read by P2)
2		Read x2	False sharing miss (x2 invalidated by write of x1)
3	Write x1		False sharing miss (block shared due to read in P2)
4		Write x2	False sharing miss (same reason as step 3)
5	Read x2		True sharing miss (value being read was written by P2)

Commercial Workload Study

Benchmark	% Time user mode	% Time kernel	% Time idle
OLTP (online transaction-processing)	71	18	11
DSS (decision support system)	87	4	9
AltaVista (web search)	>98	<1	<1

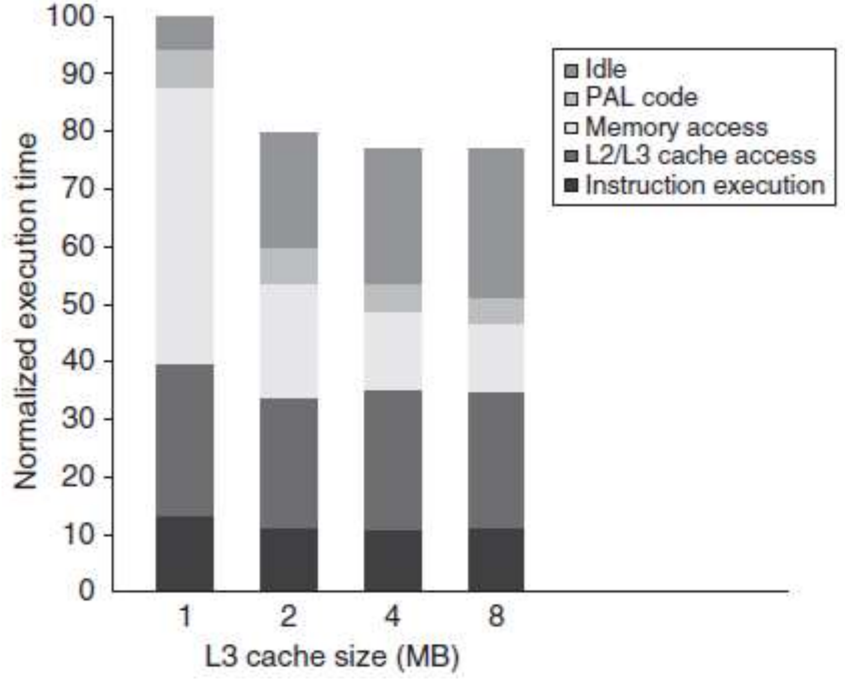
Level	Characteristic	Alpha 21164	Intel i7
L1	Size	8 KB I / 8 KB D	32 KB I / 32 KB D
	Associativity	Direct mapped	4-way I / 8-way D
	Block size	32 B	64 B
	Miss penalty	7	10
L2	Size	96 KB	256 KB
	Associativity	3-way	8-way
	Block size	32 B	64 B
	Miss penalty	21	35
L3	Size	2 MB	2 MB per core
	Associativity	Direct mapped	16-way
	Block size	64 B	64 B
	Miss penalty	80	~100

Execution Time of Workloads

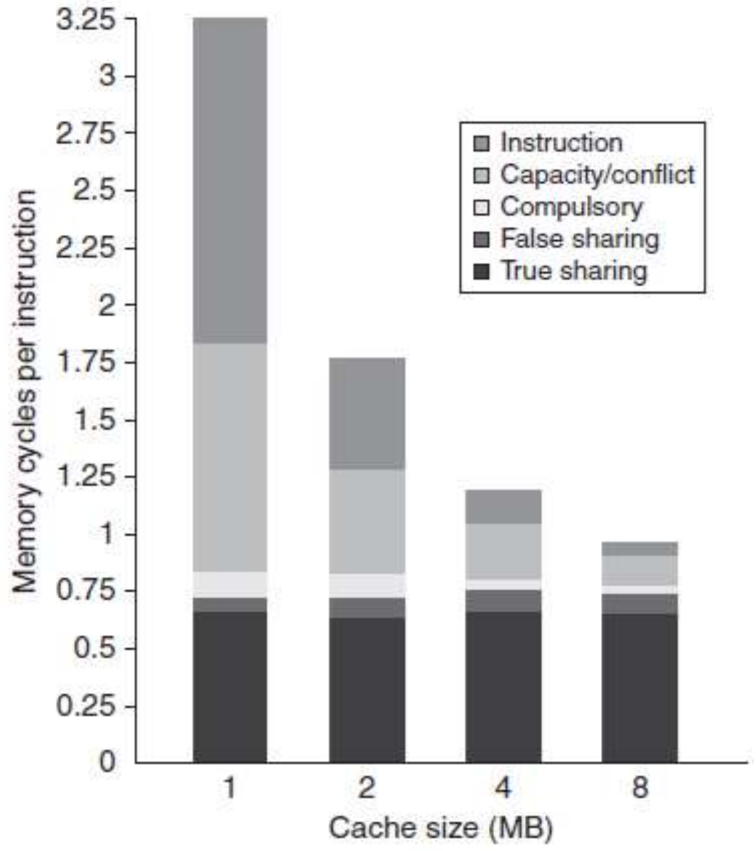


- CPI
  - 7.0 for OLTP
  - 1.6 for DSS
  - 1.3 for AltaVista
- The performance of the OLTP workload is very poor, due to a poor performance of the memory hierarchy
  → High L3 miss penalty

Impact of L3 Cache Size

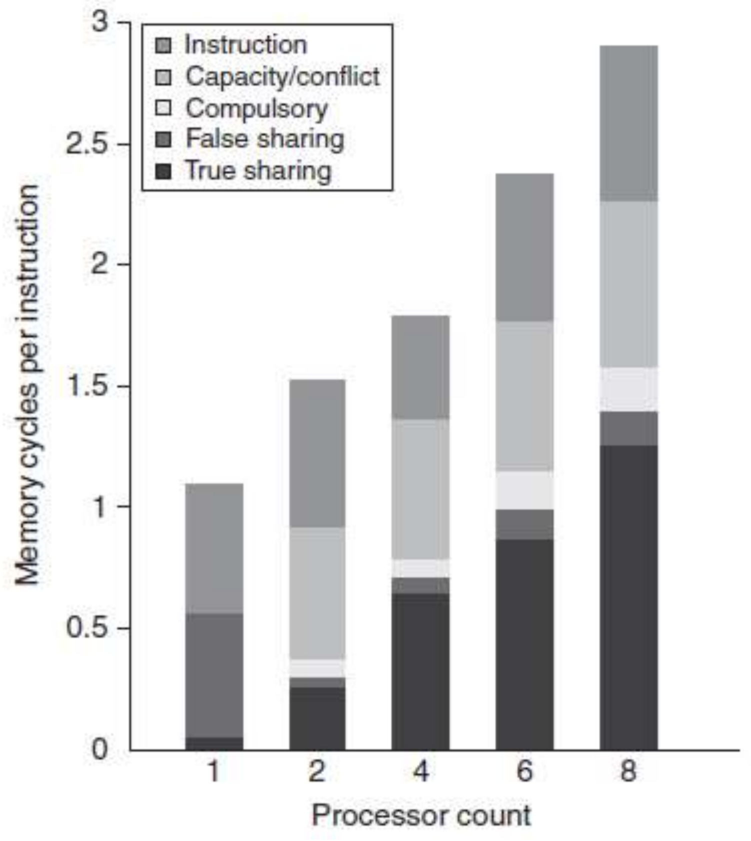


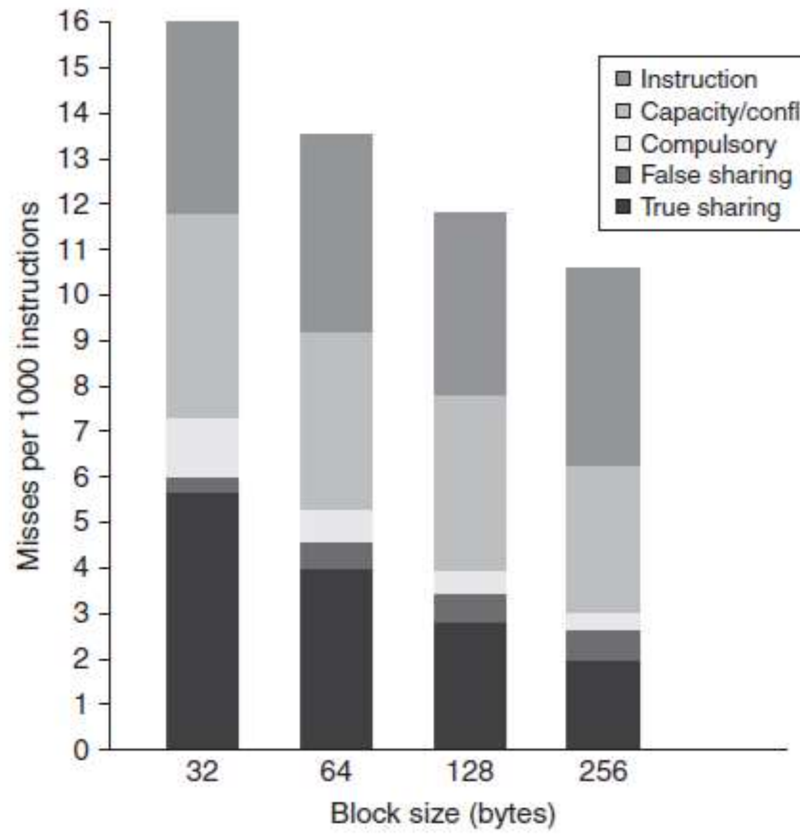
Most performance gain occurs going from 1MB to 2MB L3 cache, with diminishing returns beyond that point.



As L3 cache grows, instruction and capacity/conflict misses decrease significantly, but compulsory, false sharing, and true sharing misses remain largely unaffected.

Impact of Processor Count and Block Size





Increasing processor count leads to higher memory access cycles primarily due to increased true sharing misses.

Increasing block size from 32 to 256 bytes significantly reduces true sharing and compulsory misses, with modest impact on capacity/conflict misses.

Multiprogramming and OS Workload

- Multi-programmed workload
  - Two independent copies of the compile phases of the Andrew benchmark, a benchmark that emulates a software development environment
- Memory system

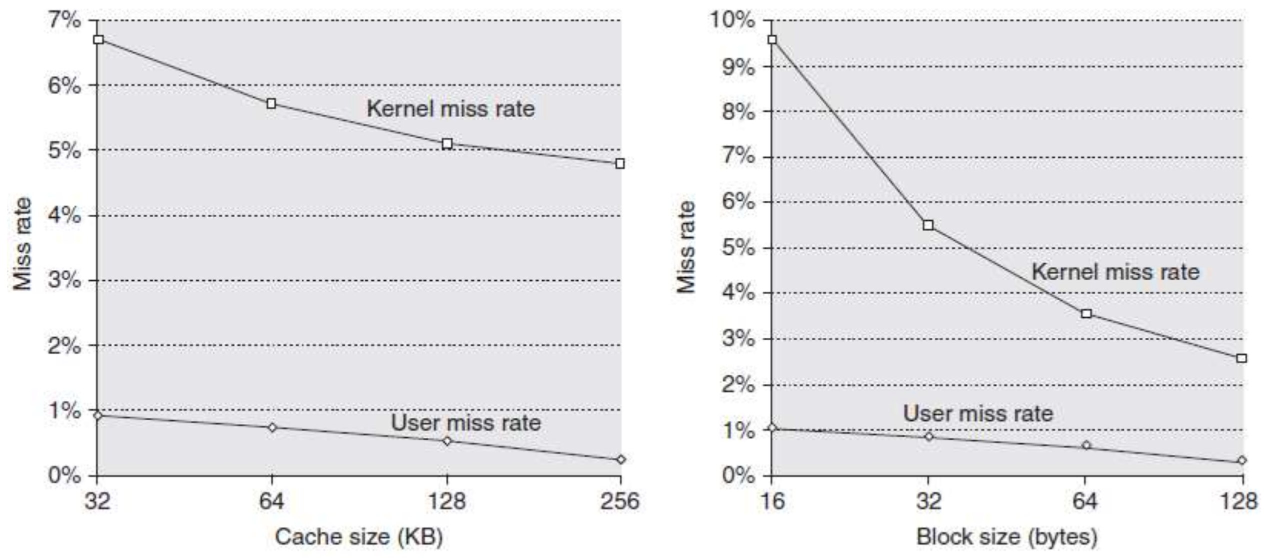
Component	Hit time (cycle)	Size
L1 instruction cache	1	32 KB, 2-way, 64-B block size
L1 data cache	1	32 KB, 2-way, 32-B block size
L2 cache	10	1 MB, unified, 2-way, 128-B block size
Main memory	100	
Disk	3 ms	

Workload Profile

	User execution	Kernel execution	Synchronization wait	Processor idle (waiting for I/O)
Instruction executed	27%	3%	1%	69%
Execution time	27%	7%	2%	64%

- This multiprogramming workload has a significant instruction cache performance loss, at least for the OS
- The instruction cache miss rate in the OS 1.7% to 0.2%
- User-level instruction cache misses are roughly one-sixth of the OS rate

L1 Data Cache Miss Rate



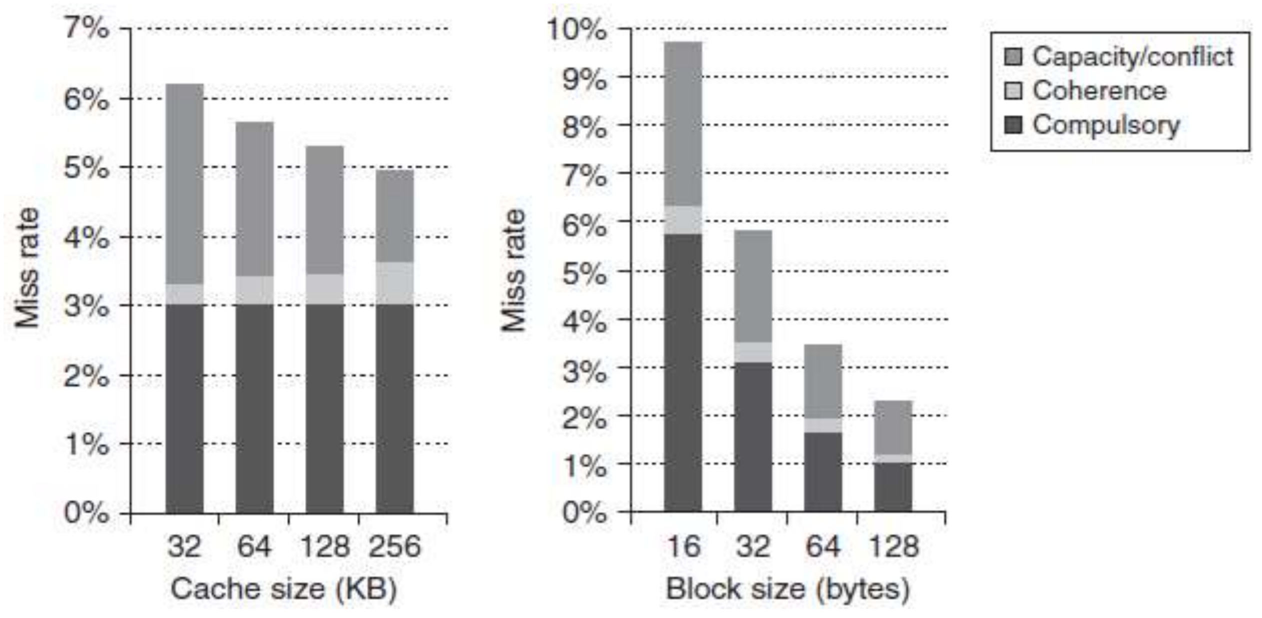
Increasing cache size affects user miss rate more than kernel miss rate. Increasing block size benefits both, but improves kernel miss rate more significantly.

Why OS Behaves Differently

- Kernel initializes all pages before allocation, increasing compulsory misses
- Kernel actually shares data, causing coherence misses
- User processes cause coherence misses only when scheduled on different processors

The OS is a much more demanding user of the memory system than user applications.

Kernel Data Miss Rate Breakdown



- For the kernel references, increasing the cache size reduces only the uniprocessor capacity/conflict miss rate
- In contrast, increasing the block size causes a reduction in the compulsory miss rate
- The absence of large increases in the coherence miss rate as block size is increased means that false sharing effects are probably insignificant, although such misses may be offsetting some of the gains from reducing the true sharing misses.

Challenges

- For the multi-programmed workload, the OS is a much more demanding user of the memory system
- It will become very difficult to build a sufficiently capable memory system
- One possible route to improving performance is to make the OS more cache aware, through either better programming environments or through programmer assistance
- OS and commercial workloads are less amenable to algorithmic or compiler restructuring
- As the number of cores increases predicting the behavior of such applications is likely to get more difficult