CS 354 - Machine Organization & Programming
Tuesday, October 24, 2017

Midterm Exam - THIS Thursday October 26th, 7:15 - 9:15 pm

- **Lec 1 (1 PM):** room 6210 of Social Sciences
- **Lec 2 (11 AM):** room B10 of Ingraham Hall
- UW ID required
- closed book, no notes, no calculators
- see “Exam Information and Policies” on course site

Project p3 (6%): DUE at 10 pm on Wednesday, November 1st

Homework hw4 (1.5%): DUE at 10 pm TODAY Tuesday, October 24th

Last Time

Basic Cache Practice
Improving Direct Mapped Cache
Set Associative Cache
Replacement Policies
Fully Associative Cache
Writing to Caches

Today

Writing to Caches (from last time)
Cache Performance Metrics
Cache Parameters and Performance
Impact of Stride
Coding for Caches
Memory Mountain

Next Time

Project p3 and gdb
Assembly Language Intro
Read: B&O 3 Intro, 3.1 - 3.3

Copyright © 2016-2017 Jim Skrentny

Cache Performance Metrics

Miss Rate

number of misses / number of memory references
lower if better

Hit Rate

1 - miss rate
number of hits / number of memory references
higher is better

Hit Time

latency - time to transfer data to caches
= set selection + line selection + word extraction
shorter is better

Miss Penalty

additional time to process a miss

L1 hit	4 cycles
L1 miss served from L2	10 cycles
L1 miss served from L3	35 cycles
L1 miss served from MM	180 cycles

35 cycles to L3 + 100 cycles for ram to serve 1st
16 bytes + 45 cycles for the remaining blocks.

Copyright © 2016-2017 Jim Skrentny

Cache Parameters and Performance

Bigger Block (fixed C and E)

→ If C and E are fixed in size, what's reduced as blocks get bigger? S fewer sets

hit rate larger block increase spatial locality
but fewer sets decrease temporal locality

hit time same

miss penalty worse - larger blocks take longer transfer over the same size bus

Bigger Cache (fixed E and B) more sets

hit rate better, more sets increases temporal locality

hit time worse, more sets slows the set selection

miss penalty same

Therefore faster caches L1 are smaller, and slower caches L3 are larger

More Lines E (fixed C and B) fewer sets

hit rate more lines decreases conflict misses

hit time worse, more lines and larger tags require more circuitry to simultaneously check

miss penalty worse, takes longer to choose the victim

Therefore faster caches L1 have a lower E and lower caches L3 have more E

Intel Quad Core i7

all: 64 byte block, use psuedo LRU, write back

L1: 32KB, 4-way Instruction & 32KB 8-way Data
L2: 256KB, 8-way
L3: 8MB, 16-way (shared by 4 cores)

Copyright © 2016-2017 Jim Skrentny

Impact of Stride

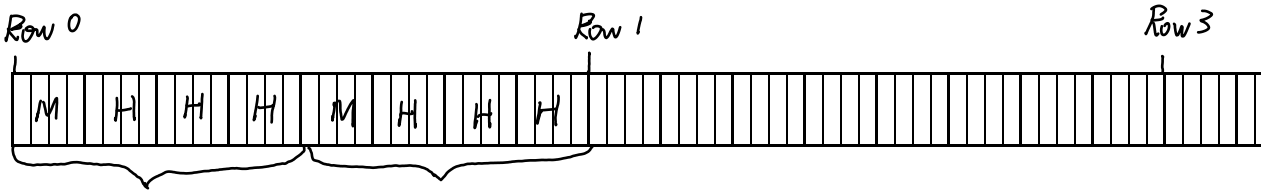
Stride Misses

% misses = min(1, (wordsize * k) / B)
k is stride in words
B blocks size in bytes

Example:

```
int initArray(int a[][8], int rows) {
   for (int i = 0; i < rows; i++)
      for(int j = 0; j < 8; j++)
         a[i][j] = i * j;
}
```

→ Draw a partial diagram of the memory layout of the first two rows of `a`:



Assume: `a` is block aligned and <u>is too big to fit entirely into the cache</u>
words are 4 bytes, block size is 16 bytes
direct-mapped cache is initially empty, write allocate used

→ Indicate the order elements are accessed in the table below and mark H for hit or M for miss:

a[i][j]	j = 0	1	2	3	4	5	6	7
i = 0	1 M	2 H	3 H	4 H	5 M	6 H	7 H	8 H
1	9 M	10 H	11 H	12 H				
...								

Miss rate = 2/8 = $\frac{1}{4}$ = 0.25
% Miss = Min (1, (4 x 1) / 16) = 25 %

→ Now exchange the `i` and `j` loops mark the table again:

a[i][j]	j = 0	1	2	3	4	5	6	7
i = 0	1 M							
1	2 M							
...	3 M							

⋮
Overwrite block 0

100 %

Copyright © 2016-2017 Jim Skrentny

Coding for Caches

- Make the common case go fast

  determine where your code typically spends
  most of its time and ensure that code rums fast

- Minimize cache misses
  - code your inner loops to minimize stride
  - keep your code's working set small enough to fit into the cache

Memory Blocking to Reduce Working Set Size

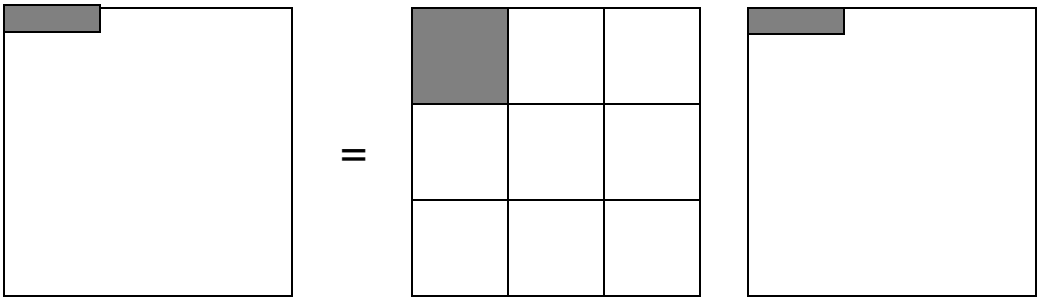
improve performance by dividing the application data into
appropriate sized blocks that maximize spatial locality

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & AB_{22} \end{bmatrix}$$

Matrix Multiply C = A B

$C_{11} = A_{11}B_{11} + A_{12}B_{21}$, $C_{12} = A_{11}B_{12} + A_{12}B_{22}$

$C_{21} = A_{21}B_{11} + A_{22}B_{21}$, $C_{22} = A_{21}B_{12} + A_{22}B_{22}$



- code is more difficult to understand so apply to most
critical data processing (math library functions)

Copyright © 2016-2017 Jim Skrentny

Memory Mountain

Independent Variables

stride - 1 to 16 double words step size used to scan through array
size - 2K to 64 MB arraysize

Dependent Variable

read throughput - 0 to 7000 MB/s

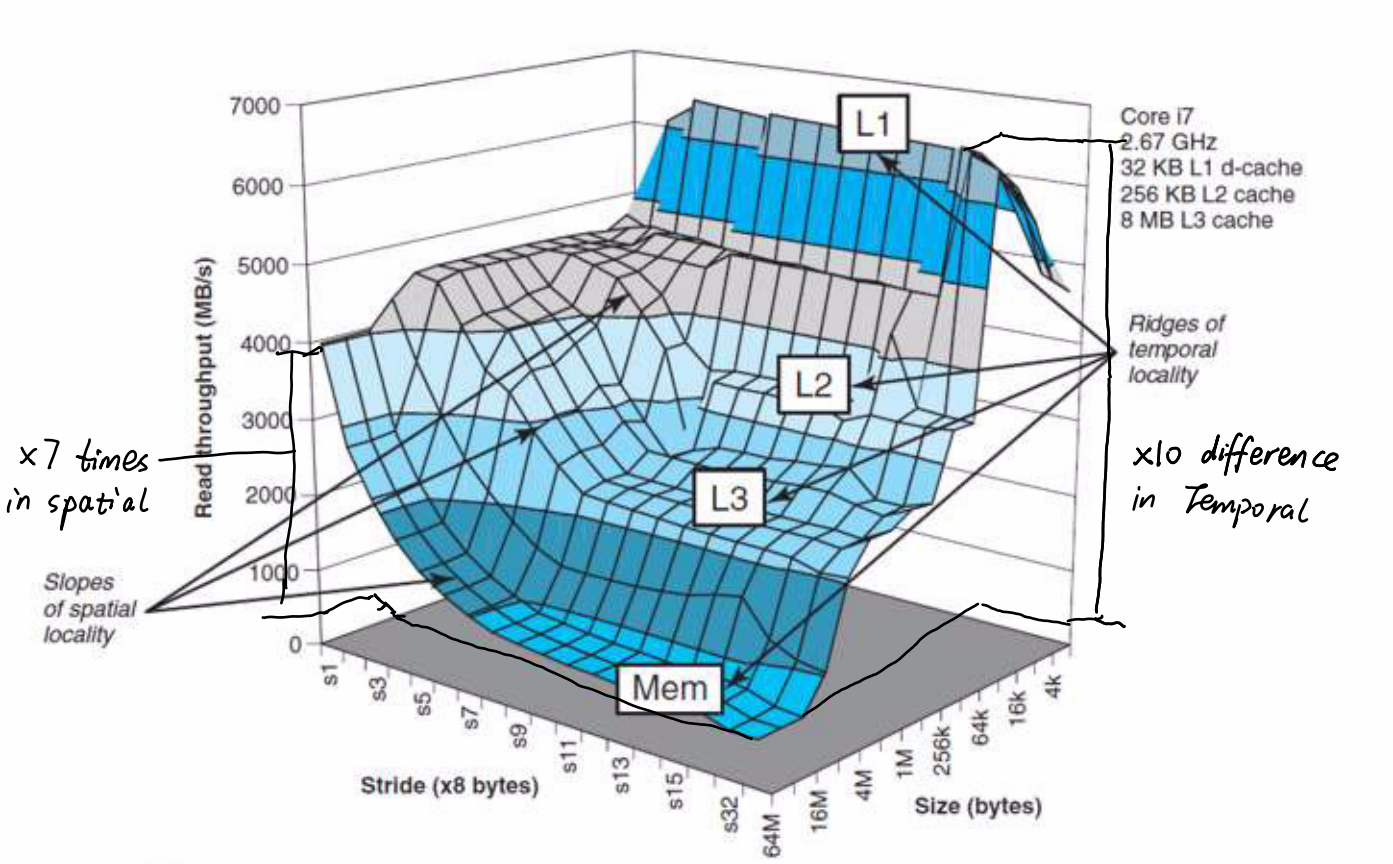


Figure 6.43 The memory mountain.

Computer Systems, A Programmer's Perspective
Second Edition, Bryant and O'Hallaron

Temporal Locality Impacts

Spatial Locality Impacts

- memory access I not characterized by a single value, it is a landscape that you can exploit through spatial an temporal locality.

Copyright © 2016-2017 Jim Skrentny