Cache Organization

March 30, 2016   .   Ganesh Kumar
int dotproduct(int x[8], int y[8]) {
    int sum = 0;
    int i;
    for (i = 0; i < 8; i++)
        sum += x[i] * y[i];
    return sum;
}

Good spatial locality?
Set 0

| 0 | 16 Bytes |

Set 1

| 0 | 16 Bytes |

Assume a 6-bit address space with t=1, s=1 and b=4.

Say x[0] is stored at address 0.... and x[7] at address 28.
And y[0] starts immediately after at address 32 .... and y[7] at 60.
<table>
<thead>
<tr>
<th>Element</th>
<th>Address</th>
<th>Set index</th>
<th>Element</th>
<th>Address</th>
<th>Set index</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[0]</td>
<td>0</td>
<td>0</td>
<td>y[0]</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>x[1]</td>
<td>4</td>
<td>0</td>
<td>y[1]</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>x[2]</td>
<td>8</td>
<td>0</td>
<td>y[2]</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>x[3]</td>
<td>12</td>
<td>0</td>
<td>y[3]</td>
<td>44</td>
<td>0</td>
</tr>
<tr>
<td>x[7]</td>
<td>28</td>
<td>1</td>
<td>y[7]</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

x[0] - Address 0 = 0 0 0000 -> Maps to Set 0
x[3] - Address 12 = 0 0 1100 -> Maps to Set 0
The first four element addresses in X map to set 0.

y[0] - Address 32 = 1 0 0000 -> Maps to Set 0
y[3] - Address 44 = 1 0 1100 -> Maps to Set 0
The first four element addresses in Y map to set 0.
In 1st iteration,
Search for $x[0]$ in cache.... Valid bit not set... Cache miss!
Read the block containing $x[0]$ onto the cache.
Still in 1st iteration,
Search for $y[0]$ in cache... Address 32 - $10\,0000$... Set 0
Tag bits don’t match... Cache miss!
So, read the block containing $y[0]$ onto the cache.
Still in 1st iteration,
Search for y[0] in cache... Address 32 - 10 0000
Tag bits don’t match... Cache miss!
So, read the block containing y[0] onto the cache.
Replace existing line in Set 0
Solution to these problems?

Increase E!

Set Associate Cache

\[ 1 < E < C / B \]

\[ E = 2 \rightarrow \text{Two-way associative cache} \]
\[ E = 3 \rightarrow \text{Three-way associative cache} \]

E is also called associativity.

Set Matching

Line Matching

\[(\text{key, value}) \]

valid + tag bits (Associative memory)
10100          Conventional memory

Line Replacement (on miss)

Which line to replace?

1. Empty line
2. Policy → Random
   → Least Frequently Used
   → Least Recently Used

Need to keep track of how many times each line is used.
LRU.

Search for a line $\rightarrow 3$

Fully Associative Cache

$E = C / B$
E ≤ c \not \in \mathbb{B} \; \text{why?}

We have no set bits (s = 0)

We just have 1 set

T = S \times E \times S \times E
So far $\rightarrow$ reads!

Writes?

Say we write to a word $w$ in the cache (write hit)

When should we update the value of $w$ in MM or the Disk?

1) Write-through

   Do it immediately!

Drawback?

   i) Super slow!

   ii) Bus traffic!
2. Write-back

Write only if a line is going to be replaced.

Drawbacks:

Need an extra bit!

Dirty bit $\rightarrow$ If updated, set to 1;
If not, set to 0.
Write misses?

Need to write to W

1. Write-allocate
   Load w onto the cache and then update it (Assumes good spatial locality)

2. No-write allocate
   Bypass the cache and write directly to the net memory.

Write through / No-write-allocate
Write back / Write-allocate

Cache levels at lower memory hierarchy use this.
Cache diagram

Core 0

- Registers
- L1 d-cache
- L2 cache

Core 1, Core 2, Core 3

L1 i-cache

L3 cache

Main memory

Could be only read only