Improving Cache Performance

- Average memory-access time = Hit time + Miss rate x Miss penalty

- Improve performance by:
  1. Reduce the miss rate,
  2. Reduce the miss penalty, or
  3. Reduce the time to hit in the cache.
  4. Increase ILP

A Closer Look at Misses

- Classifying Misses: 3 Cs
  - Compulsory—The first access to a block is not in the cache, so the block must be brought into the cache. These are also called cold start misses or first reference misses (Misses in Infinite Cache)
  - Capacity—If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being discarded and later retrieved. (Misses in Size X Cache)
  - Conflict—If the block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory and capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. These are also called collision misses or interference misses. (Misses in N-way Associative, Size X Cache)

3Cs Absolute Miss Rate

How Can Reduce the Miss Rate?

- Block Size?
- Cache Size?
- Associativity?
- Prediction?
- Compiler?

Block Size vs. Cache Measures

- Increasing Block Size generally increases Miss Penalty and decreases Miss Rate

Reduce Misses via Larger Block Size

- Reduced compulsory misses
- Increased conflict/capacity
- Improved spatial locality
- Increased miss penalty
- For small cache sizes
Example: Optimal block size

- Memory system: 40 cycles overhead, hit time = 1 cycle
- 16 bytes / 2 clocks → 16 bytes in 42 cycles, 32 bytes in 44 cycles, ...
- For chart below, which block size has the minimum average memory access time for each cache size?

<table>
<thead>
<tr>
<th>Block size (bytes)</th>
<th>Miss rates by cache size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1K</td>
</tr>
<tr>
<td>16</td>
<td>10.0%</td>
</tr>
<tr>
<td>32</td>
<td>13.34%</td>
</tr>
<tr>
<td>64</td>
<td>13.76%</td>
</tr>
<tr>
<td>128</td>
<td>16.64%</td>
</tr>
<tr>
<td>256</td>
<td>22.01%</td>
</tr>
</tbody>
</table>

Example: Optimal block size

- Miss penalties: 40 cycles + 2 * (block size / 16)
- \( \text{AMAT} = \text{hit time} + \text{miss rate} \times \text{miss penalty} \)
- 16KB cache, 16 byte block: 1 + 0.0394 * 42 = 2.6548
- 16KB cache, 32 byte block: 1 + 0.0287 * 44 = 2.2628
- 16KB cache, 64 byte block: 1 + 0.0277 * 48 = 2.2672
- 16KB cache, 128 byte block: 1 + 0.0237 * 56 = 2.5512

Reduce Misses via Larger Caches

- Take advantage of technology (memory gap)
- Decrease capacity misses
- Increase hit time and cost

General Rules of thumb

<table>
<thead>
<tr>
<th>Cache Size (KB)</th>
<th>1-way</th>
<th>2-way</th>
<th>4-way</th>
<th>8-way</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>0.02</td>
<td>0.04</td>
<td>0.06</td>
<td>0.08</td>
</tr>
<tr>
<td>4K</td>
<td>0.04</td>
<td>0.08</td>
<td>0.10</td>
<td>0.12</td>
</tr>
<tr>
<td>16K</td>
<td>0.06</td>
<td>0.10</td>
<td>0.12</td>
<td>0.14</td>
</tr>
<tr>
<td>64K</td>
<td>0.08</td>
<td>0.12</td>
<td>0.14</td>
<td>0.16</td>
</tr>
<tr>
<td>256K</td>
<td>0.10</td>
<td>0.14</td>
<td>0.16</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Example: AMAT vs. miss rate

- Relative clock cycle hit time for different associativities
  - Cycle hit time for 2-way = 1.10 x 1-way cycle hit time
  - Cycle hit time for 4-way = 1.12 x 1-way cycle hit time
  - Cycle hit time for 8-way = 1.14 x 1-way cycle hit time

- Direct mapped (1-way) cycle hit time = 1 cycle

- Using data below, find cache sizes that make these statements true:
  - AMAT(8-way) < AMAT(4-way)
  - AMAT(4-way) < AMAT(2-way)
  - AMAT(2-way) < AMAT(1-way)

<table>
<thead>
<tr>
<th>Degree Associative</th>
<th>Miss rates by cache size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1K</td>
</tr>
<tr>
<td>1-way</td>
<td>133</td>
</tr>
<tr>
<td>2-way</td>
<td>105</td>
</tr>
<tr>
<td>4-way</td>
<td>99</td>
</tr>
<tr>
<td>8-way</td>
<td>89</td>
</tr>
</tbody>
</table>

Reduce Misses via Higher Associativity

- Remember tradeoff:
  - Increase block size → reduced miss rate, increased miss penalty
- New tradeoff:
  - Higher associativity → reduced miss rate, increased hit time

- Beware: Execution time is only final measure!
  - Will clock cycle time increase?
  - Hill [1988] suggested hit time external cache +10%, internal + 2% for 2-way vs. 1-way
Example: AMAT vs. miss rate

- AMAT\_n-way = hit time\_n-way + miss rate\_n-way \times miss penalty\_n-way
- 16KB cache, 1-way: 1.10 \times 0.02*50 = 2.2
- 16KB cache, 2-way: 1.12 \times 0.02*50 = 2.12
- 16KB cache, 4-way: 1.14 \times 0.02*50 = 2.04

- Results:
  - AMAT(8-way) < AMAT(4-way) for cache sizes < 32K
  - AMAT(4-way) < AMAT(2-way) for all cache sizes
  - AMAT(2-way) < AMAT(1-way) for all cache sizes

<table>
<thead>
<tr>
<th>Associative</th>
<th>AMAT by cache size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1K</td>
</tr>
<tr>
<td>1-way</td>
<td>1.79</td>
</tr>
<tr>
<td>2-way</td>
<td>1.60</td>
</tr>
<tr>
<td>4-way</td>
<td>1.45</td>
</tr>
<tr>
<td>8-way</td>
<td>1.39</td>
</tr>
</tbody>
</table>

Reducing Misses via Way prediction and Pseudo-Associativity

- Problem:
  - Direct mapped cache offers lowest hit times
  - 2-way set associative reduces conflict misses

- Solution:
  - Way-prediction: extra bits predict which block to try on next access
  - Pseudo-associativity:
    - Divide cache based on simple association (ex: inverse of MSB)
    - On miss check other half of cache (pseudo-hit)

- Complications:
  - Way prediction used on Alpha 21264, MIPS R4300
  - Pipelining implementation difficult for varying hit times
  - Better suited for caches not highly coupled to processor

<table>
<thead>
<tr>
<th>Hit Time</th>
<th>Pseudo Hit Time</th>
<th>Miss Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>