Recap: Improving Cache Performance

CPU time = (CPU execution clock cycles \* Memory stall clock cycles) \* clock cycle time
Memory stall clock cycles = (Reads \* Read miss rate \* Read miss penalty) + (Writes \* Write miss rate \* Write miss penalty)

• Above assumes 1-cycle to hit in cache
  – Hard to achieve in current-day processors (faster clocks, larger caches)
  – More reasonable to also include hit time in the performance equation

Average memory access time = Hit Time + Miss rate \* Miss penalty

Recap: Cache Optimization Summary

<table>
<thead>
<tr>
<th>Technique</th>
<th>MP</th>
<th>MR</th>
<th>HT</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multilevel caches</td>
<td>+</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Early Restart &amp; Critical Word 1st</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority to Read Misses</td>
<td>+</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merging write buffer</td>
<td>+</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victim Caches</td>
<td>+</td>
<td>+</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Larger Block Size</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Higher Associativity</td>
<td>+</td>
<td>-</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Pseudo-Associative Caches</td>
<td>+</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler Reduce Misses</td>
<td>+</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-Blocking Caches</td>
<td>+</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW Prefetching of Inst/Data</td>
<td>+</td>
<td>+</td>
<td>2/3</td>
<td></td>
</tr>
<tr>
<td>Compiler Controlled Prefetching</td>
<td>+</td>
<td>+</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Avoiding Address Translation</td>
<td>+</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Trace Cache</td>
<td>+</td>
<td>+</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Static Random Access Memory (SRAM)

• A type of semiconductor memory used for Caches
• Memory retains its contents as long as power remains applied
  ➔ Static unlike dynamic RAM (DRAM) that needs to be periodically refreshed
• Nevertheless it is volatile memory
• Six transistors for each memory cell (bit)
• Gate level access time

Dynamic RAM (DRAM)

• Uses smaller number of transistors (one transistor per cell)
• Uses a capacitor
• Changes in the charge are detected and amplified
• Cheaper than SRAM
• Slower than SRAM
**DRAM (Cont’d)**

- Reading is destructive
  - When a bit is read you destroy the stored value (if it was a zero)
  - After reading a bit, you must re-write it (if it was a zero - in practice it is easier to re-write the value in all cases)
- Stored values must be periodically refreshed
  - Bits are stored using capacitance, it is necessary to periodically re-write (refresh) the stored values because charge leaks away over time
  - Refresh is typically managed by the memory subsystem
- Reading is a relatively slow process
  - Because of need to re-write the contents of a bit that has been read, a stabilization period is required after reading
  - Process of detecting changes via sense amplifiers is quite slow

**Cache vs. Main Memory**

- Cache uses SRAM: Static Random Access Memory
  - No refresh (6 transistors/bit vs. 1 transistor/bit, more wires, area is 10X)
  - Address not divided
- Main Memory is DRAM: Dynamic Random Access Memory
  - Dynamic since needs to be refreshed periodically (~8 ms, ~5% time)
  - Addresses divided into 2 parts (Memory as a 2D matrix) sent one at a time to reduce the number of address pins:
    - RAS or Row Access Strobe, CAS or Column Access Strobe
- Performance of Main Memory:
  - Latency
    - Access Time: time between request and when word arrives
    - Cycle Time: time between requests
  - Bandwidth
- DRAM/SRAM size ratio of 4 – 8 for comparable technologies, SRAM/DRAM cost, cycle time ratio 8 – 16

**Internal Organization of a 256M bit DRAM**

- Internally, might use banks of memory arrays
  - E.g., 256 1024x1024 arrays, or 64 2048x2048 arrays
- Normally packaged as dual inline memory modules (DIMMs)
  - Typically 4-16 DRAM chips, 8 byte wide

**Improving Memory Performance in a DRAM**

- Increasingly important because fewer chips/system

  **Evolutionary**
  - Fast page mode
    - Allow multiple CAS accesses without need for intervening RAS
    - Optimizes sequential access, exploiting the row buffer (1024-2048 bits)
    - Extended Data Out (EDO): 30% faster in page mode
    - Overlays data out with CAS toggling
  - Synchronous DRAM (SDRAM)
    - Avoid need for handshaking between chip and memory controller
    - Chip also has a register with number of requested bytes: these are transmitted without explicit requests from controller
  - Double Data Rate (DDR) DRAM
    - Transmit data from chip on both the falling and rising edge of clock signal
    - DDR2 is the next-generation DDR memory technology which features faster speeds, higher data bandwidth, lower power consumption, and enhanced thermal performance

**DRAM History**

- DRAMs: capacity +60%/yr, cost -30%/yr
  - 2.5X cells/area, 1.5X die size in ~3 years
- Rely on increasing numbers of computers and memory per computer
  - SIMM or DIMM is replaceable unit
  - Computers can use any generation DRAM
  - Growth slowing because demand is coming down

- Commodity industry
  - High volume, low profit, conservative
  - Little organization innovation in 20 years

- Order of importance: (primary) Cost/bit, (secondary) Capacity
  - First RAMBUS: 10X BW, ~30% cost, but little impact

**Higher Bandwidths**
Error Correction

- Motivation:
  - Failures/time proportional to number of bits!
  - As DRAM cells shrink, more vulnerable
- Went through period in which failure rate was low enough without error correction that people didn’t do correction
  - DRAM banks too large now
  -Servers always corrected memory systems
- Basic idea: add redundancy through parity bits
  - Simple but wasteful version:
    - Keep three copies of everything, vote to find right value
    - 200% overhead
  - ECC (error correction code) SDRAM is memory that is able to detect and correct some SDRAM errors
    - Replaced parity memory which could only detect, but not correct errors
    - Most ECC SDRAMs can correct single bit errors and detect, but not correct larger errors
    - One example: 64 data bits + 8 parity bits (11% overhead)

Improving Main Memory Performance

- Making memory faster has been difficult
  - At least try to get it to transfer a lot of data to higher memory bandwidth
- 1) Wider Main Memory:
  - Timing model
    - 4 to send address
    - 56 access time
    - 4 to send data
  - Cache block = 4 words
  - Memory width of 4 words: 4 x (4 + 56 + 4) = 256
  - Memory width of 4 words: 4 + 56 + 4 = 64
- 2) Use interleaved memory
  - Assume four banks are interleaved at word level
  - Each bank 1-word wide
- 3) Generalization: Independent Memory Banks
  - Memory banks for independent accesses vs. faster sequential accesses
    - Multiprocessors
    - I/O
    - CPU with Hit under m Misses, Non-blocking Caches
  - Each bank needs separate address and possibly data lines
  - New terminology
    - Memory is organized as Superbanks of possibly word-interleaved banks
    - Each superbank has separate address and possibly data lines
  - How many banks?
    - Ensure that if memory is being accessed sequentially (e.g., when processing an array) then by the time you try to read a second word from a bank, the first access has finished
    - Unfortunately, larger memory chips imply fewer banks

Avoiding Bank Conflicts

- Even if we assume that there are lots of banks, run into conflicts
  - With 128 banks, conflict on word accesses (512 is a multiple of 128)
  - Software fixes: loop interchange, or padding array so that it is not 2^n
  - Hardware fix: Prime number of banks, b, each with n words
    - Property: No conflicts for any sequence of consecutive addresses, as long as stride is not a multiple of b
    - Property: Resolving the address to a bank number, address within bank
      - bank number = address mod b
      - address within bank = address / b
      - modulus and divide per memory access are easy if number of banks is 2^n
      - for prime number of banks, banker (particularly )
Address Computation w/ Prime Number of Banks

- Fast computation is possible by storing words in banks using modulo interleaving (b banks, n = 2^c words per bank) ...
  - bank number = address mod b (same as before)
  - address within bank = address mod 2^c

- Above result stems from the Chinese Remainder Theorem

<table>
<thead>
<tr>
<th>Bank Number</th>
<th>Seq. Interleaved</th>
<th>Modulo Interleaved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 2</td>
<td>0 16 8</td>
</tr>
<tr>
<td>1</td>
<td>3 4</td>
<td>3 1 17</td>
</tr>
<tr>
<td>2</td>
<td>6 7 8</td>
<td>18 10 2</td>
</tr>
<tr>
<td>3</td>
<td>9 10 11</td>
<td>1 19 11</td>
</tr>
<tr>
<td>4</td>
<td>12 13 14</td>
<td>12 4 23</td>
</tr>
<tr>
<td>5</td>
<td>15 16 17</td>
<td>21 13 6</td>
</tr>
<tr>
<td>6</td>
<td>18 19 20</td>
<td>6 22 18</td>
</tr>
<tr>
<td>7</td>
<td>21 22 23</td>
<td>15 7 23</td>
</tr>
</tbody>
</table>